

Disambiguating Identity Web References using Social Data

Matthew Rowe

Submitted in partial fulfilment of the requirements
for the degree of Doctor of Philosophy at
the University of Sheffield
Department of Computer Science

April 2010

Abstract

The World Wide Web has evolved into an interactive information network, allowing web users to collaborate and share information on a massive scale. A large amount of information now available on the World Wide Web is personal information, which has either been disseminated voluntarily (i.e., a personal web page, profile page) or involuntarily (i.e., telephone directory, electoral register). The sensitive nature of personal information and its widespread visibility has led to a rise in malevolent web practices such as lateral surveillance and identity theft. Given this context, it is important to investigate the use of accurate and efficient automated techniques that would allow web users to detect those web resources containing their personal information without the need for manual processing of web content.

The process of detecting *identity web references* consists of searching the Web for possible web citations and then performing disambiguation to identify the true citations for a given person. In this thesis three distinct disambiguation techniques are presented each of which fuse Semantic Web technologies with existing state of the art techniques; Inference Rules built from seed data, Random Walks over a graph-space built from metadata models and Self-training machine learning classifiers combined with semantic graph matching techniques. A detailed evaluation of these techniques is presented using a real world dataset with respect to several baseline measures including human processing.

To function effectively automated disambiguation techniques require seed data, or background knowledge, describing a given person, however producing this data manually is expensive. Therefore this thesis explores how data leveraged from Social Web platforms can be used. To assess the suitability of such platforms a user study is presented assessing the similarity between the identity which users create online with their real world equivalent. The findings demonstrate a significant overlap and therefore suggest the suitability of data leveraged from the Social Web for use by automated techniques. Techniques are presented to leverage seed data from Social Web platforms such as Facebook, MySpace and Twitter. Semantic Web technologies such as knowledge representation formats and ontologies are used to describe the data in a machine-readable manner. Methodologies are also presented to construct metadata models from web resources which are to be analysed also using Semantic Web technologies.

The findings from this body of work demonstrate the feasibility of automated disambiguation techniques to replace the manual processing of web content and the facilitation of *identity web reference* discovery.

Acknowledgements

It only seems right to first and foremost thank my supervisor Prof. Fabio Ciravegna for his supervision and effort throughout my studies, not only when pursuing my Ph.D. but also during my study as an undergraduate under his supervision. Despite his hectic lifestyle and fully booked diary he always made time for me, for which I am extremely grateful. His lateral view to any topic area helped to clarify my work and view it from a completely unique perspective, something which I found to be invaluable when explaining my work to others. His guidance has provoked more questions and avenues of work for the future which I now wish to pursue.

The second level of support within my studies, particularly with regards to the machine learning aspect of my work, was Jose Iria. His explanations of how learning mechanisms worked helped me to understand how I could adapt and apply such techniques within the field of disambiguation. I would also like to thank the members of the Organisations, Information and Knowledge Group for their help with the experiments which created one of the baselines used for evaluation. Their time and effort was essential to providing an analysis of humans performing disambiguation tasks.

Insights into the Social Web were provided by the members of the W3C Social Web Incubator Group. Discussions of work and the views of digital identity on the Social Web within this forum have proved extremely useful in understanding the dynamics of the Social Web from several distinct perspectives: web developers, data evangelists and researchers. Similarly I acknowledge the support which I have had from members of the Semantic Web community and the help they have given me in understanding the power of semantics, this is something I have tried to convey within this thesis.

I would like to acknowledge the help and support my family have given me throughout my period of my study. Their guidance has provided a secure and stable platform which has allowed me to pursue my work and become established in a number of research communities. Lastly I would like to thank my cat Tinker for her listening capabilities. Whenever there was a hard problem to consider, explaining it to her seemed to help!

Publications

The work presented within this thesis is present within the following publications:

- Disambiguating Identity Web References using Web 2.0 Data and Semantics. M Rowe and F Ciravegna. In the Journal of Web Semantics. (2010)
- Inferring Web Citations using Social Data and SPARQL Rules. M Rowe. In proceedings of Linking of User Profiles and Applications in the Social Semantic Web, Extended Semantic Web Conference 2010. Heraklion, Crete. (2010)
- Harnessing the Social Web: The Science of Identity Disambiguation. M Rowe and F Ciravegna. In proceedings of Web Science Conference 2010. Raleigh, USA. (2010)
- Data.dcs: From Legacy Data to Linked Data. M Rowe. In proceedings of Linked Data on the Web Workshop, World Wide Web Conference 2010. Raleigh, USA. (2010)
- The Credibility of Digital Identity Information on the Social Web: A User Study. M Rowe. In proceedings of 4th Workshop on Information Credibility on the Web, World Wide Web Conference. 2010. Raleigh, USA. (2010)
- Improving Support for Web-based Visual Analysis of Social Graphs. A-S Dadzie, M Rowe and A.E Cano. IEEE VisWeek 2009 Workshop: Putting Visualization on the Web. (2009)
- Mapping between Digital Identity Ontologies through SISM. M Rowe. In proceedings of Social Data on the Web Workshop, International Semantic Web Conference 2009, Washington DC, USA. (2009)
- Learning to Classify Identity Web References Using RDF Graphs. M Rowe and J Iria. In poster track of International Semantic Web Conference 2009. Washington DC, USA. (2009)
- Applying Semantic Social Graphs to Disambiguate Identity References. M Rowe. In proceedings of European Semantic Web Conference 2009. Heraklion, Crete. (2009)
- Assessing Trust: Contextual Accountability. M Rowe and J Butters. In proceedings of Trust and Privacy on the Social and Semantic Web Workshop, European Semantic Web Conference 2009. Heraklion, Crete. (2009)

- Metasocial Wiki - Towards Interlinked Knowledge in a Decentralized Social Space. A E Cano, M Rowe and F Ciravegna. Semantic Wiki Workshop. European Semantic Web Conference 2009. Heraklion, Crete. (2009)
- Interlinking Distributed Social Graphs. M Rowe. In proceedings of Linked Data on the Web Workshop, World Wide Web Conference 2009. Madrid, Spain. (2009)
- The Interoperability of Lightweight Semantics for Social Networks. M Rowe. Position Paper for W3C Future of Social Networking Workshop 2009. Barcelona, Spain (2009)
- Getting to Me - Exporting Semantic Social Network Information from Facebook. M Rowe and F Ciravegna. In proceedings of Social Data on the Web Workshop, International Semantic Web Conference 2008. Karlsruhe, Germany. (2008)
- Disambiguating Identity through Social Circles and Social Data. M Rowe and F Ciravegna. In proceedings of Collective Intelligence Workshop, European Semantic Web Conference 2008. Tenerife, Spain. (2008)

Contents

1	Introduction	17
1.1	Motivations	17
1.2	Research Questions	19
1.3	Claims of this Thesis	19
1.4	Contributions	20
1.5	Thesis Structure	22
1.5.1	Part I - Background	22
1.5.2	Part II - Methodology	23
1.5.3	Part III - Conclusions	23
	 Part I - Background	 24
2	Digital Identity	25
2.1	Introduction	25
2.2	Web 2.0: The Emergence of a Social Web	25
2.3	Social Networking	26
2.4	Digital Identity	27
2.4.1	Fragmented Identity	29
2.4.2	Data Portability	30
2.5	Online Privacy	30
2.6	User Study: Online and Offline Identities	32
2.6.1	Evaluation Measures	32
2.6.1.1	Relevance	33
2.6.1.2	Coverage	33
2.6.2	Experimental Setup	34
2.6.2.1	Stage 1. Real World Social Network Creation	34
2.6.2.2	Stage 2. Digital Social Network Extraction	34
2.6.2.3	Stage 3. Comparing Real World and Digital Social Networks	35
2.6.3	Results	35
2.6.3.1	Relevance and Coverage	35
2.6.3.2	Demographics of Coverage	36
2.6.3.3	Behaviour	37
2.7	Summary	38
3	Digital Identity and the Semantic Web	40
3.1	Introduction	40
3.2	The Semantic Web	41
3.2.1	Origins	41
3.2.2	Representing Knowledge	41
3.2.2.1	XML	42
3.2.2.2	XML Schemas and Namespaces	42
3.2.2.3	RDF	42
3.2.2.4	RDFa	44

3.2.3	Ontologies	45
3.2.3.1	RDF-Schema	46
3.2.3.2	OWL	46
3.2.4	Rules	47
3.2.5	SPARQL	48
3.2.6	Linked Data	49
3.3	The Semantics of Digital Identity Information	50
3.3.1	Friend of a Friend (FOAF)	51
3.3.2	Semantically Interlinked Online Communities (SIOC)	52
3.3.3	Nepomuk Contact Ontology (NCO)	52
3.3.4	Microformats	53
3.3.4.1	hCard	53
3.3.4.2	XFN	54
3.4	Summary	54

Part II - Methodology 56

4	Requirements and Design: an Approach to Identify Identity Web References	57
4.1	Introduction	57
4.2	Related Work	58
4.2.1	Seeded Disambiguation Techniques	58
4.2.2	Unseeded Disambiguation Techniques	60
4.3	Requirements	61
4.3.1	Requirements for Disambiguation	62
4.3.2	Requirements for Seed Data	62
4.4	Approach Architecture	63
4.4.1	The Applicability of Social Data for Disambiguation	63
4.4.2	Seed Data Generation	64
4.4.3	Metadata Model Generation	65
4.4.4	Identity Disambiguation	65
4.4.4.1	Inference Rules	65
4.4.4.2	Random Walks	65
4.4.4.3	Self-training	66
4.5	Summary	66
5	Collecting Seed Data from the Social Web	67
5.1	Introduction	67
5.2	Related Work	67
5.2.1	Exporting Individual Social Graphs	67
5.2.1.1	Flickr Exporter	68
5.2.1.2	Twitter Exporter	68
5.2.2	Interlinking Social Graphs	68
5.2.2.1	Node/Edge Overlap	69
5.2.2.2	Node Mapping	69
5.2.2.3	Graph Smushing	70
5.2.2.4	Reference Reconciliation	70
5.3	Requirements for Collecting Seed Data from the Social Web	71
5.3.1	Requirements for the Exportation of Individual Social Graphs	71
5.3.2	Requirements for Interlinking Distributed Social Graphs	72
5.4	An Approach to Collect Seed Data from the Social Web	72
5.5	Exportation of Individual Social Graphs	74
5.5.1	Twitter Exporter	74
5.5.2	Facebook Exporter	75
5.5.3	OpenSocial Exporter	76

5.5.4	Graph Enrichment	76
5.5.5	The Validity of Exported Social Graphs	77
5.6	Interlinking Distributed Social Graphs	77
5.6.1	Graph Reasoning	78
5.6.1.1	Unique Identifiers	78
5.6.1.2	Geographical Reasoning	79
5.6.2	Producing Linked Data	81
5.7	Evaluation	82
5.7.1	Evaluation Measures	82
5.7.2	Dataset	83
5.7.3	Experimental Setup	83
5.7.4	Results	84
5.8	Summary	84
6	Generating Metadata Models	87
6.1	Introduction	87
6.2	Related Work	88
6.2.1	Handling XHTML Documents	88
6.2.2	Handling HTML Documents	88
6.3	Requirements for Generating Metadata Models	90
6.4	Generating Metadata Models from Web Resources	91
6.4.1	Generating RDF Models from XHTML Documents	91
6.4.2	Generating RDF Models from HTML Documents	93
6.4.2.1	Generating Context Windows	94
6.4.2.2	Extracting Person Information using Hidden Markov Models	97
6.4.2.3	Vocabulary Dimension Control	101
6.4.2.4	Deriving Transition Paths	102
6.4.3	Evaluation	103
6.4.3.1	Evaluation Measures	103
6.4.3.2	Dataset	103
6.4.3.3	Experimental Setup	104
6.4.3.4	Results	104
6.4.3.5	Deployment	105
6.5	Normalising Metadata Models to Consistent Semantics	106
6.5.1	State of the Art	106
6.5.1.1	Ontology Alignment	106
6.5.1.2	Semantics of Mappings	107
6.5.1.3	Transforming RDF Models	108
6.5.2	Social Identity Schema Mapping (SISM)	108
6.5.2.1	Mapping Identity Ontologies	109
6.5.2.2	Collections of Mappings	109
6.5.3	Generating Rules for Transforming RDF Models	111
6.5.3.1	owl:equivalentProperty/owl:equivalentClass/skos:broader	111
6.5.3.2	skos:related	112
6.5.3.3	skos:narrower	114
6.5.4	Applying Rules: Model Transformation	114
6.6	Summary	115
7	Rule-based Disambiguation	117
7.1	Introduction	117
7.2	Related Work	118
7.2.1	Rule Construction	118
7.2.2	Rule-based Classification Systems	119
7.3	Requirements for Rule-Based Disambiguation Techniques	119
7.4	Inference Rules: Rule-based Disambiguation	120
7.4.1	Constructing the Rule Base	120

7.4.1.1	Instance Extraction	121
7.4.1.2	Building Rules	123
7.4.2	Example Rules	124
7.4.2.1	Inverse Functional Property Rules	124
7.4.2.2	Person Rules	125
7.4.2.3	Location Rules	126
7.4.3	Applying Rules	126
7.5	Summary	127
8	Graph-based Disambiguation	129
8.1	Introduction	129
8.2	Related Work	130
8.3	Requirements for Graph-based Disambiguation Techniques	132
8.4	Random Walks: Graph-based Disambiguation	132
8.4.1	Building an Interlinked Graph Space	134
8.4.2	Identifying Strongly Connected Components	135
8.4.3	Traversing the Graph Space	136
8.4.3.1	Weighting Edges	136
8.4.3.2	Random Walks	138
8.4.4	Measuring distances within the Graph Space	139
8.4.4.1	Commute Time	139
8.4.4.2	Optimum Transitions	141
8.4.5	Clustering Web Resources	141
8.5	Summary	142
9	Semi-supervised Machine Learning for Disambiguation	144
9.1	Introduction	144
9.2	Related Work	145
9.2.1	Semi-Supervised Learning Strategies	145
9.2.1.1	Self-training	145
9.2.1.2	Expectation-Maximisation	147
9.2.1.3	Co-training	148
9.2.2	Classifiers	148
9.2.2.1	Perceptron	148
9.2.2.2	Support Vector Machines	150
9.2.2.3	Naive Bayes	153
9.2.3	Proximity Measures	154
9.2.3.1	Cosine-Angle Distance	154
9.2.3.2	Euclidean Distance	155
9.2.3.3	Mahalanobis Distance	155
9.3	Requirements for Semi-supervised Machine Learning Techniques	156
9.4	Self-training: Semi-supervised Machine Learning	156
9.4.1	Generating Binary Seed Data	158
9.4.2	Learning to Classify Identity Web References	161
9.4.2.1	Feature Extraction	161
9.4.2.2	Feature Similarity Measures	162
9.4.2.3	Mapping RDF models into a Machine Learning Dataset	164
9.4.2.4	Self-training	165
9.5	Summary	166
10	Evaluation	167
10.1	Introduction	167
10.2	Evaluation Measures	168
10.3	Dataset	169
10.4	Experimental Setup	170
10.4.1	Inference Rules	170

10.4.2	Random Walks	170
10.4.3	Self-training	171
10.5	Baseline Techniques	171
10.5.1	Baseline 1: Person Name as Positive Classification	171
10.5.2	Baseline 2: Hierarchical Clustering using Person Names	172
10.5.3	Baseline 3: Human Processing	172
10.5.4	Baseline 4: Outlier Detection	172
10.6	Constructing a Gold Standard	173
10.7	Results and Discussion	174
10.7.1	Inference Rules	174
10.7.2	Random Walks	176
10.7.3	Self-training	177
10.7.3.1	Perceptron	179
10.7.3.2	SVM	180
10.7.3.3	Naive Bayes	180
10.7.3.4	Effects of Self-Training	180
10.7.3.5	Effects of Negative Set Enlargement	185
10.8	Comparing Information Representations: Metadata vs Bag-of-words Models	187
10.8.1	Perceptron	190
10.8.2	SVM	190
10.8.3	Naive Bayes	190
10.9	Comparison of Disambiguation Techniques	192
10.10	Summary	193

Part III - Conclusions 194

11	Conclusions	195
11.1	Introduction	195
11.2	Research Conclusions	195
11.2.1	Automated Web Reference Identification	195
11.2.2	Analysis of Research Questions	197
11.2.3	Analysis of Claims	200
11.3	Analysis of Methodology Requirements	202
11.3.1	Requirements for Disambiguation	203
11.3.2	Requirements for Seed Data	204
11.3.3	Disambiguation Techniques	205
11.3.3.1	Rule-based	205
11.3.3.2	Graph-based	206
11.3.3.3	Semi-supervised Machine Learning	207
11.4	Summary	209
12	Future Work and Outlook	210
12.1	Introduction	210
12.2	A Proposed Approach to Risk Assessment	211
12.2.1	The Severity of Personal Information Dissemination	212
12.3	Future Methodologies	213
12.3.1	Random Walks	213
12.3.1.1	Spectral Clustering with Commute Time Kernel	213
12.3.1.2	Resolution of Random Walks	214
12.3.1.3	Edge Weights and their Effects on Random Walks	214
12.3.2	Semi-supervised Machine Learning	215
12.3.2.1	Discovering an Optimal Hypothesis during Self-training	215
12.3.2.2	Combining the Disambiguation Techniques	215
12.4	Identity and the Internet of Things	216
12.5	Data.dcs	217

12.6 HTML5: The next revision of the World Wide Web	218
12.7 Summary	218
Bibliography	219
Appendices	235
A Graph Reasoning Algorithm	235
B Linked Data Social Graph	237
C Tuning Levenshtein Measure	239
D Tuning Node/Edge Overlap	240
E Tuning Node Mapping	241
F Person Attribute Patterns	242
F.1 Name Patterns	242
F.2 Email Patterns	242
F.3 URL Patterns	242
G Experiments with Absolute Discounting	243
H Experiments using Interrater Gold Standard	244
I Tuning Baseline 2: Hierarchical Clustering	246
J Tuning Random Walks: Effects of Edge Weighting	248
J.1 Commute Time	249
J.2 Optimum Transitions	250
K Tuning Rocchio Classification for Self-Training	251
K.1 Strongest-Negative Enlargement	251
K.1.1 Perceptron	251
K.1.2 SVM	252
K.1.3 Naive Bayes	252
K.2 Full-Negative Enlargement	253
K.2.1 Perceptron	253
K.2.2 SVM	254
K.2.3 Naive Bayes	254
L Tuning Yarowsky Algorithm	258
M Self-training: Random Sampling for Negative Set Generation	260

List of Figures

2.1	The tiers of Digital Identity from [Win05]	28
2.2	Relevance and Coverage measures from social network comparisons	36
2.3	Distribution of relationship within the overlap between real world and digital social networks	36
2.4	Distribution of ages within the overlap between real world and digital social networks	37
2.5	Analysis of image sharing within digital social networks	38
2.6	Analysis of message sharing within digital social networks	38
3.1	RDF graph structure	43
3.2	RDF graph structure with additional information	44
3.3	Linked Data relating resources defining the author George Orwell in disparate datasets	50
4.1	Architecture of an approach to identify <i>identity web references</i>	63
5.1	Equivalent RDF instances modelled as graphs g and h , where the description of each instance differs in size.	69
5.2	RDF instances matched based on matching values defined by the inverse functional property <i>foaf:mbox</i>	71
5.3	The proposed approach to collect seed data from Social Web platforms	73
5.4	Portion of an example social graph from Twitter	74
5.5	Individual <i>foaf:Person</i> instances represented as subgraphs	77
6.1	Person information residing within a HTML document	93
6.2	Pipeline architecture of the technique for generating a metadata model from a HTML document	94
6.3	DOM Structure of the HTML document from Figure 6.1	95
6.4	Topology of a Hidden Markov Model for person information extraction	100
6.5	Hidden Markov Model providing labels for a given input sequence of tokens which are then transformed to symbols using mapping functions	102
6.6	Snippet of an RDF Model generated from a HTML document	105
6.7	Collection of mapped Person concepts	110
6.8	Collection of mapped Relationship concepts	111
6.9	Transforming an RDF model from using the VCard ontology to use the FOAF and GeoNames ontologies	111
7.1	A proposed technique to build and apply inference rules to disambiguate <i>identity web references</i>	120
7.2	Applying the Resource Leaves construct to extract RDF instance descriptions from a given metadata model.	122
7.3	Applying a rule to the RDF model of http://oak.dcs.shef.ac.uk/people matching a social network member's email address in the web resource	127
7.4	Applying a rule to the RDF model of http://oak.dcs.shef.ac.uk/people matching the social graph owner's homepage and name	128

8.1	The pipeline architecture of the proposed graph-based disambiguation technique.	133
8.2	The graph-space compilation process	134
8.3	Snippet of a compiled graph-space. A complete graph-space would be too large to display graphically in a legible manner.	135
8.4	Identifying Strongly Connected Components	136
8.5	A transformed graph-space connecting nodes by Commute Time Measures . .	140
9.1	Perceptron Architecture for an input vector x with 3 features	149
9.2	Learnt Perceptron with a separating hyperplane which is orthogonal to the weights vector	150
9.3	Learnt separator hyperplane with two supporting hyperplanes where the margin is maximised	151
9.4	Applying Self-training to disambiguate <i>identity web references</i>	157
9.5	Initial training sets iteratively enlarged using resources with the strongest classification confidences	158
9.6	Rocchio classification in a vector space used to generate negatives	160
9.7	Section of an example RDF model representing information found within a given web resource	162
9.8	RDF instances to be used as features	162
10.1	Precision/Recall curve for Inference Rules plotted against the baseline techniques	175
10.2	Precision/Recall curve for Random Walks using both Commute Time and Optimum Transitions to cluster, plotted against the baseline techniques . . .	177
10.3	Precision/Recall curves for permutations of Self-trained Perceptron with differing feature similarity measures plotted against Baseline Techniques	179
10.4	Precision/Recall curves for permutations of Self-trained SVM with differing feature similarity measures plotted against Baseline Techniques	180
10.5	Precision/Recall curves for permutations of Self-trained Naive Bayes with differing feature similarity measures plotted against Baseline Techniques . . .	181
10.6	Precision/recall curves showing pre-training classifier ($i = 0$) against converged classifier ($i = n$)	182
10.7	Effects of Self-training on different permutations of Perceptron and feature similarity measures	183
10.8	Effects of Self-training on different permutations of SVM and feature similarity measures	184
10.9	Effects of Self-training on different permutations of Naive Bayes and feature similarity measures	186
10.10	Precision/Recall curves for classifiers when enlarging the negative set with all negative samples after each training iteration	188
10.11	Pipeline architecture of the Yarowsky implementation	189
10.12	Precision/Recall curves for Classifiers using Metadata Models compared with a Bag-of-Words Model	191
10.13	Precision/Recall curves for Rules, Random Walks using Commute Time distances and Self-trained SVM with IFP	192
12.1	Architecture of an approach to assess the risk of identity theft/lateral surveillance posed to web users	211
C.1	Tuning Levenshtein distance for matching person names	239
D.1	False Positive Rate vs False Negative Rate when Tuning Node/Edge Overlap threshold	240
E.1	False Positive Rate vs False Negative Rate when Tuning Node Mapping threshold	241

G.1	Tuning Discount when smoothing using Absolute Discounting	243
I.1	Precision/Recall curve for various clustering thresholds for Hierarchical Clustering	247
J.1	Precision/Recall curves when tuning σ and η for Random Walks with Commute Time distances	249
J.2	Precision/Recall curves when tuning σ and η for Random Walks with Optimum Transitions	250
K.1	Precision/Recall curves when tuning ρ for SNE Self-trained Perceptron with different feature similarity measures	252
K.2	Precision/Recall curves when tuning ρ for SNE Self-trained SVM with different feature similarity measures	253
K.3	Precision/Recall curves when tuning ρ for SNE Self-trained Naive Bayes with different feature similarity measures	254
K.4	Precision/Recall curves when tuning ρ for FNE Self-trained Perceptron with different feature similarity measures	255
K.5	Precision/Recall curves when tuning ρ for FNE Self-trained SVM with different feature similarity measures	256
K.6	Precision/Recall curves when tuning ρ for FNE Self-trained Naive Bayes with different feature similarity measures	257
L.1	Precision/Recall curves for classifiers trained using the Yarowsky algorithm with different levels of ρ	259

List of Tables

5.1	Results from interlinking social graphs from Facebook and Twitter	84
6.1	Accuracy levels of Hidden Markov Models using different smoothing methods	104
10.1	Rater agreement	173
10.2	Strength of agreement when using Kappa (κ) statistic	174
10.3	Accuracy of Inference Rules compared to baseline techniques	174
10.4	Accuracy of Random Walks and clustering using Commute Time and Optimum Transitions compared to baseline techniques	176
10.5	Accuracy of different permutations of Self-trained classifiers and feature similarity measures compared to baseline techniques	178
10.6	Accuracy of different permutations of Self-trained classifiers using Full Negative Enlargement and feature similarity measures compared to baseline techniques	185
10.7	Accuracy of classifiers trained using Yarowky algorithm and a Bag-of-words model	190
12.1	Person attributes visible in web resources returned using Self-trained SVM with IFP	212
H.1	Accuracy of Inference Rules with respect to Baseline techniques using GS' . .	244
H.2	Accuracy of applying Random Walks with Commute Time and Optimum Transitions with respect to Baseline techniques using GS'	245
H.3	Accuracy of Different Permutations of Machine Learning Classifier and Feature Similarity Measure with respect to Baseline techniques using GS'	245
M.1	Accuracy levels using a random negative set generation strategy	260

List of Algorithms

1	<code>cwFind(<i>dom</i>)</code> : Given the DOM of a HTML document, returns a set of context windows	96
2	<code>domManip(<i>trigger,node</i>)</code> : Given a trigger string and the node which contains the trigger, derives the suitable DOM element to extract the window from . .	96
3	<code>extractWin(<i>trigger,content</i>)</code> : Given a trigger string and a DOM element's content, extracts the window from the trigger onwards	97
4	<code>intCheck(<i>S,T,c_a,c_b</i>)</code> : Check RDF model integrity prior to applying <i>skos:related</i> based rule. <i>S</i> is the mapping schema, <i>T</i> is the tripleset of the RDF model, <i>c_a</i> is the concept currently in <i>T</i> and <i>c_b</i> is the concept which <i>c_a</i> is mapped to in <i>S</i>	113
5	<code>buildRules(<i>r_p,G</i>)</code> : Induces rules from RDF instances. Input to this algorithm is the social graph (<i>G</i>) and the person whose web resources are to be disambiguated (<i>r_p</i>). Rules are induced and added to the rule base (<i>RB</i>). . . .	123
6	<code>optTrans(<i>P</i>)</code> : Derives the Optimum Transitions matrix <i>O</i> from incrementing the matrix power of <i>P</i>	141
7	<code>yarowsky(<i>X,Y⁰</i>)</code> : Takes as input a set of examples (<i>X</i>) and an initial labelling (<i>Y⁰</i>). A classifier is trained and applied to the examples repeatedly until its parameters converge. The algorithm outputs a final labelling (<i>Y</i>).	147
8	<code>rocchio(<i>P,U</i>)</code> : Takes as input the positive set (<i>P</i>) and the universal set (<i>U</i>) and compiles positive and negative prototype vectors which instances are then compared against to derive negative set membership. The new universal set (<i>U</i>) and the negative set (<i>N</i>) are returned.	159
9	<code>itEnl(<i>P,N,U</i>)</code> : Iterative self-training based on enlargement of training data according to classification rankings. <i>P</i> is the positive set, <i>N</i> is the negative set and <i>U</i> is the universal set	165
10	<code>sim(<i>g,h</i>)</code> : Compares two instances of <i>foaf:Person</i> represented as graphs (<i>g</i> and <i>h</i>) for similarity based on information within the graphs. Returns <i>true</i> if the instances are equivalent, otherwise <i>false</i>	235
11	<code>checkLocations(<i>loc_g,loc_h</i>)</code> : Takes two locations <i>loc_g</i> and <i>loc_h</i> and checks for a relation between them. If one exists then <i>true</i> is returned, otherwise <i>false</i> . .	236

Chapter 1

Introduction

1.1 Motivations

The World Wide Web now offers a variety of feature-rich interactive platforms that allow web users to become involved online by producing content and sharing information. Sites such as Wikipedia¹ and Yahoo Answers² now provide the functionality which allows web users to produce content and collaborate in an online environment. Media organisations such as the BBC³ now encourage the public to share information and interact as part of its News service. Web logging (blogging) platforms allow those web users who lack the technical expertise to construct web pages to publish their thoughts and musings on their own personal blog. Other sites, such as social networking platforms, give web users a base from which they are able to deploy, maintain and enrich their online web presence through bespoke profiles depicting their digital identity. In short the World Wide Web now provides a variety of rich functionalities and services which are inviting and easy to use, these services in turn allow web users to become involved in producing content on the Web, in doing so elevating their online presence.

The elevation of web users' online presence has produced several notable by-products: through standard Web search engines, such as Google⁴ and Yahoo,⁵ it is now possible to find information about the majority of people online, whether those people wish to be visible in such a way or not. The visibility of this information, and its sensitive nature, has lead to several malevolent web practices becoming increasingly popular. One such practice is lateral surveillance [And06], the action of finding and surveying a person without their knowledge. This has become a widespread practice among employers who are keen to vet potential employees prior to offering them a job and also among people vetting prospective dates. Another malevolent web practice on the rise is online identity theft, which currently costs the UK economy £1.2 billion per annum.⁶ In both of these cases the accessibility of personal information can have severe consequences if reused without a person's knowledge. As web users share more of their personal information within the public domain the risk of this

¹<http://www.wikipedia.org>

²<http://answers.yahoo.com/>

³<http://www.bbc.co.uk>

⁴<http://www.google.com>

⁵<http://www.yahoo.com>

⁶<http://www.identitytheft.org.uk/faqs.asp>

information being read by the wrong people or stolen and reused increases.

To assess whether a person is at risk from identity theft or susceptible to lateral surveillance, web resources (web pages, data feeds, etc) containing the person’s personal information are found and the information contained within those resources assessed for risk, thereby allowing the correct action to be taken (i.e., removing the information, or altering its visibility). For example, UK government guidelines stipulate that the visibility of the person’s name, their address and email address could lead to identity theft.⁷ The step of identifying the correct web resources which refer to a given person (*identity web references*) is essential to tackle malevolent web practices. This process is comprised of two steps; a filtering step where web references are retrieved, which are possible web citations, pointing to a specific person name (e.g., by retrieving web resources via standard search engines); and a vetting step where web resources are assessed and found to refer to a given person or to a person with an identical or similar name. This second vetting step - the decision process - is referred to within this thesis as *disambiguation*.

In recent years the practice of searching the Web and detecting web citations has risen. A recent study [MS10] conducted by the Pew Research Centre of 2,253 participants found that 57% of those surveyed searched for information about themselves online (an increase from 47% in 2006). At present those users who wish to identify their web citations must *manually* search the Web for all the web resources which may cite them, and then decide which are *identity web references*. Both the filtering and the vetting steps in this case are time consuming to web users, for instance searching the Web for a person name yields a large number of results returned for that specific name (e.g., Google returns 4,750,000 for John Smith only). The size of this result set in turn renders manual disambiguation largely unfeasible, given that the person must go through each web resource by hand and assess its contents. This presents a clear motivation for automated techniques to replace such manual processing both when searching the Web and performing disambiguation. For the filtering step, automation would save the user the time taken when querying the Web using separate search engines in order to leverage a broad range of possible web citations, while for disambiguation, automation would save the user the time required to analyse a web resource’s inherent features and decide whether the web resource is an *identity web reference* or not.

The scale of the Web and its tendency to change - given that new data is published, shared and discussed each day - presents a clear motivation for the application of automated disambiguation techniques, where such methods are able to save web users the time and effort required to detect web resources containing their personal information. Automated techniques require prior knowledge (seed data) describing the person whose *identity web references* are to be found. They need to be explicitly informed of the properties, features and attributes which make a given person unique, as the name of a person is insufficient - consider once again the example of querying Google using John Smith as the query input. However, producing such seed data manually for one person let alone for multiple people can be costly, time consuming and limited by access to resources.

⁷<http://www.getsafeonline.org/nqcontent.cfm?aid=1132>

1.2 Research Questions

The above problem setting motivates the work explored within this thesis. As mentioned the process of identifying *identity web references* is divided into two-stages: finding and disambiguating. This thesis focuses on the second stage of the identification process: the disambiguation of *identity web references* using automated techniques. Research is presented which explores the following research question:

How can identity web references be disambiguated automatically?

This question covers a broad spectrum and possible avenues of work. For clarity the above question is divided into several separate questions - based on the problem setting explained previously - which contribute to exploring this broad area of research. These are as follows:

1. *Is it possible to define automated techniques that alleviate the need for human processing?*
2. *Is it possible to define automated techniques that require minimal supervision and only the provision of initial seed data?*
3. *Can seed data be gathered or constructed inexpensively?*
4. *How can information distributed across the World Wide Web and found in heterogeneous sources be interpreted by automated techniques?*

1.3 Claims of this Thesis

The sheer scale of the World Wide Web and the rate at which information is published requires techniques capable of interpreting information from heterogeneous sources. Work within the Semantic Web community has explored methods to attribute information with meaning and structure in order to facilitate its interpretation by automated techniques and machines. For instance the KnowItAll system [ECD⁺05] extracted named entities from the Web which were then used to populate knowledge bases. The DBPedia project [ABK⁺08] explored techniques to leverage information from the online encyclopedia Wikipedia which was then enriched with explicit semantics to enable machine-readability. The use of semantics by such efforts provides the necessary means by which personal information from disparate sources can be interpreted effectively, thereby enabling automated disambiguation techniques to perform the task of humans and detect web resources which cite a given person. This claim is explicitly made as follows:

- *Automated disambiguation techniques are able to replace human processing*

Existing approaches [CDF⁺98, Coh02] to web document classification often use shallow approaches to represent document features. For instance, a common approach is to employ a bag-of-words model consisting of all keywords in a given document following lightweight data processing. This thesis makes the claim that a rich representation of information found within web documents is essential to ensure accurate disambiguation results. Furthermore,

by employing Semantic Web technologies to represent document features inferencing is possible over the provided knowledge structures, something which is limited with such shallow approaches to document representation:

- *Disambiguation techniques are aided by a semantic representation of web resources*

Applying automated techniques to disambiguate *identity web references* for a given person requires seed data to function effectively. This places constraints on the seed data: it must describe the identity of the person whose web citations are to be found and it must also be accurate (i.e. contain no errors that could lead to detrimental effects on disambiguation decisions). Furthermore, building seed data can be an expensive and time-consuming task, for instance by forcing a person to list all the people he/she knows together with their work details. Social data leveraged from the Social Web provides the background knowledge without the burden of seed data construction. Social Web platforms such as Facebook⁸ and MySpace⁹ provide a rich collection of functionalities that allow users to construct profiles which represent their identity in a digital space. This thesis claims that the digital identities which web users construct are simply representations of their real selves:

- *Data found on Social Web platforms is representative of real identity information*

Leveraging social data to support automated disambiguation techniques provides a novel solution to the problem of seed data generation. Such data describes both biographical and social network information of a given person, thereby providing the necessary features to enable accurate disambiguation of *identity web references*. State of the art work within the field of disambiguation [Mal05, KCMNT08, LBGT05, JWA⁺09] exploits similar features to facilitate disambiguation, such as the relationships and social network which a person maintains. This thesis claims that social data provides the necessary background knowledge from which the disambiguation of *identity web references* can be performed:

- *Social data provides the background knowledge required by automated disambiguation techniques*

1.4 Contributions

This thesis presents a body of work exploring automated techniques to disambiguate *identity web reference* through the use of social data. The main contribution of this thesis are three distinct techniques, each of which use seed data leveraged from Social Web platforms in a different manner:

1. *Inference Rules*: rules are built from the provided seed data and its inherent features. The constructed *Inference Rules* are then applied to the web resources which are to be disambiguated.
2. *Random Walks*: a graph-based disambiguation technique known as *Random Walks* is applied to measure distances within a graph-space and clusters nodes according to those distances. The novel contribution of this technique is its use of knowledge

⁸<http://www.facebook.com>

⁹<http://www.myspace.com>

structures from which a graph-space is compiled and an edge weighting method based on the semantics of the edges.

3. *Self-training*: a semi-supervised machine learning technique known as *Self-training* is used to train a machine learning classifier. The classifier is then used to label web resources as either citing a person or not. The effects of *Self-training* with different permutations of classifiers and feature similarity measures - used by the classifiers - are investigated to derive the optimum combination for disambiguation.

To enable the above techniques to be explored and seed data provided accordingly, this thesis also contains the following contributions:

- *An approach to identify identity web references*

The user-centric approach to identify *identity web references* provides the necessary framework to supply disambiguation techniques with seed data leveraged from the Social Web. The approach is divided into three stages: first social graphs are gathered from multiple Social Web platforms and linked together to produce a single social graph, second web resources which are possible web citations are gathered, and third the social graph is provided as seed data to the disambiguation techniques along with the web resources which are to be analysed.

- *An investigation of the relationship between data on Social Web platforms and real identity information*

Automated disambiguation techniques rely on seed data which is representative of a given person's real identity. To explore the viability of using data obtained from Social Web platforms this thesis provides a user study of identity information shared on the Social Web platform Facebook. This study analyses the extent to which the online identity constructed by web users on such a platform mirrors their real world identity. The results from the study provide empirical evidence which proves that Social Web platforms contain information which is representative of real identity information, and that web users employ such platforms as an extension of their offline lives.

- *Methodologies to generate and integrate seed data from Social Web platforms*

The identity of a Social Web user is fragmented into several pieces where each piece is found within a distinct platform (e.g., Facebook, MySpace). Several tools and methodologies are presented to leverage each of these pieces as social graphs from Social Web platforms. Semantic Web technologies are utilised to convert data into a machine-readable form using knowledge representation formats and ontologies. This provides consistent semantics describing data from distinct platforms. A technique known as Graph Reasoning is presented which reconciles identity fragments by exploiting contextual information and combining distributed social graphs into a single interlinked social graph for use as seed data.

- *Techniques to generate metadata models describing the information contained within web resources in a machine-processable form*

Automated disambiguation techniques benefit from a consistent interpretation of information with seed data collected from the Social Web. Therefore techniques are presented which construct metadata models describing information contained within heterogeneous web resources. An approach is presented to extract information from HTML documents via Document Object Model (DOM) manipulation and Hidden Markov Models.

- *A Methodology to normalise metadata models to a unary set of semantics*

Metadata models generated from heterogeneous web resources often suffer from the problem of homogeneous data defined using different ontologies - this is particularly evident when describing identity information given the varied use of differing digital identity ontologies. Therefore a mapping vocabulary called the Social Identity Schema Mapping vocabulary (SISM) is presented, which contains relations between concepts within distinct identity ontologies. Using this schema a methodology is presented to convert metadata models to use required ontologies by applying transformation rules automatically generated from SISM.

- *An extensive evaluation of disambiguation techniques using a real world dataset*

A detailed evaluation is presented of the explored disambiguation techniques using a dataset containing possible web citations referring to 50 people. The presented disambiguation techniques are compared against several baseline measures including a disambiguation technique which does not use seed data, and human processing of the same dataset. To demonstrate the effectiveness of using Semantic Web technologies for knowledge representation, disambiguation techniques are compared against shallow approaches which do not require rich markup, the results from this comparison demonstrate the improvement in precision achieved through the use of Semantic Web technologies. Furthermore, a key contribution of this evaluation is the comparison of disambiguation techniques against a baseline of humans performing the same task, where the findings empirically demonstrate a significant improvement in performance over humans.

1.5 Thesis Structure

This thesis is divided into three distinct parts, these parts are structured as follows:

1.5.1 Part I - Background

Chapter 2 introduces the notion of digital identity, describing its emergence and how identity information is fragmented across the Social Web. The motivational aspects behind the decentralisation of identity information are discussed, along with the challenges of compiling such identity fragments together. A detailed user study is presented investigating if the digital identity that web users build reflects their real world identity.

Chapter 3 discusses the relationship between digital identity and the Semantic Web by highlighting the importance of defining the semantics of digital identity information. This chapter presents an overview of the Semantic Web and presents key state of the art Semantic Web technologies such as knowledge representation formats, ontologies, logic and rules and current ontologies used to define the semantics of digital identity information.

1.5.2 Part II - Methodology

Chapter 4 presents the approach to identify *identity web references*. Related work to the field of disambiguation is introduced to provide an overview of the distinction between *seeded* techniques and *unseeded* techniques. Requirements are listed which disambiguation techniques must fulfill when used to detect *identity web references*. The architecture and design of the approach to identify web citations is explained, and an overview of the three stages of the approach is presented by describing the process of generating seed data and gathering possible web citations, and passing both onto a given disambiguation technique.

Chapter 5 investigates the reconciliation of digital identity fragments on the Social Web. Several tools and techniques are presented to reconcile identity information by exporting and integrating information from Social Web platforms and services, thus building a digital identity representation from available social data.

Chapter 6 presents current methods for generating metadata describing information within web resources and investigates techniques to lift metadata models from such resources. Differing techniques are explained for metadata model generation, each of which employ a different approach depending on the format of the web resource (i.e. HTML, XHTML). An approach to normalise metadata models to a given ontology set, and therefore a set of consistent semantics, is also described.

Chapters 7, 8 and 9 each investigate different approaches to disambiguate identity web references using social data. Chapter 7 investigates the use of rule-based disambiguation techniques, chapter 8 investigates graph-based disambiguation techniques and chapter 9 investigates semi-supervised machine learning techniques.

Chapter 10 presents a detailed empirical evaluation of the disambiguation approaches discussed in chapters 7, 8 and 9. The evaluation of each approach is performed using a large-scale dataset containing web pages referring to 50 participants. Information retrieval metrics are employed to measure the accuracy of each approach against several baseline measures.

1.5.3 Part III - Conclusions

Chapter 11 presents the conclusions drawn from this thesis, in particular commenting on the suitability of the explored techniques in helping to solve online problems such as identity theft and lateral surveillance, and how such techniques can be deployed in a wider context such as entity disambiguation. Chapter 11 also examines the requirements laid out in chapter 4 for the overall approach, the individual requirements for the techniques used throughout the various stages and the research questions from chapter 1.

Chapter 12 discusses what the future holds for digital identity by explaining the emergence of the Internet of Things and the predicted role that identity will play. The adaptation of the tools, techniques and approaches presented within this thesis are discussed in order to accurately monitor information about a given entity on the web.

Part I

Background

“Defend your privacy, the ads yammered on the hour, from all media. Is a stranger tuning in on you? Are you really alone?”

From “Ubik” by Philip K. Dick

Chapter 2

Digital Identity

2.1 Introduction

The growth of the Social Web has provided users with a wide range platforms and services covering distinct topics and areas of interest. Social Web platforms offer rich features and functionalities allowing users to construct a digital identity by creating a visible profile describing their information and interests. This thesis claims that users construct a digital identity on Social Web platforms which is representative of their real world identity. In order to investigate this claim this chapter discusses the background surrounding the Social Web and digital identity and presents a user study analysing the extent to which data found on the Social Web platform Facebook mirrors real identity information. The results from this user study provide empirical evidence to support the above claim and therefore suggest that disambiguation techniques would benefit from the use of such social data.

This chapter is structured as follows: section 2.2 describes the Social Web and how it emerged. Section 2.3 details the role of social networking on the Social Web, presenting several example Social Web platforms which provide social networking functionalities and describing information which is available from such platforms. Section 2.4 explains the concept of digital identity, discussing the fragmentation of digital identity information across the Social Web and the limited portability of this data. Section 2.5 describes the by-products of the increased presence of Web users including identity theft and lateral surveillance. Section 2.6 presents a study of Social Web users, measuring the extent to which real identity information is replicated on the Social Web. Section 2.7 finalises the chapter with the summary of the presented work and discussions.

2.2 Web 2.0: The Emergence of a Social Web

The original World Wide Web - which is now referred to as “Web 1.0” - was a web of anonymity for the majority of users with the exception of the very few who had a personal web page; mostly researchers and academics. The distinction between the real world and the digital world was well defined. The role of the majority of web users was as consumers, untrusted and untapped as sources of information. Web users could view information and purchase products but were unable to review or discuss them. Web 1.0 was a Web of documents tied together by hyperlinks where each document contained a separate and distinct

source of information. The information within each web document was only fit for human consumption, limiting the ability of machines to process such information.

During the first years of this millennium the Web experienced a paradigm shift: Tim O'Reilly coined the term "Web 2.0" [O'R05] as the user-centric Web where the Web acts as a platform on which applications can be built. In essence the shift is best paraphrased as the movement from a Document Web to a *Social Web*. Sites such as Wikipedia¹ proved the success of a "*wisdom of the crowd*" philosophy, by trusting an online community of contributors to audit created content and collaborate in order to share knowledge in a central space. Web logging, or "blogging", became a widely practiced action by allowing web users to publish their own content on the Web with very little technical know-how. Web users were no longer restricted by their lack of technical expertise, and published content was produced in abundance. Sites and services began to open up their data through APIs enabling web developers to combine data from multiple sources thereby producing new web applications known as "Mashups". This represented an interesting change in the role of the web user, by placing trust in their ability to use data and add content.

Web 2.0 has produced a rise in trust on the Web which is two-fold: firstly sites now trust web users as co-developers allowing them to use their data, and secondly users now trust web sites with their personal information. This latter development has been facilitated through the provision of rich functionalities which have allowed web users to replicate their real world lives in an online environment by communicating with friends, sharing information such as photos and videos, and generally interacting socially. Web 2.0 platforms which offer this functionality and provision of social networking capabilities are referred to as belonging to the *Social Web*.

2.3 Social Networking

The evolution of the World Wide Web into a web of social interaction has introduced web users to the notion of social networking in virtual spaces. By offering web users with the ability to establish and maintain relationships with other users, Social Web platforms support the construction of a social network which [FDZJ05] defines as follows:

"A social network consists of people or groups connected by a set of social relationships, such as friendship, co-working or information exchange."

Sociological studies have assessed the characteristics of the social networks formed on Social Web sites. For instance work by [HRT⁺08] states that such platforms are primarily used to maintain offline relationships. Similar work by [ESL07] also claims that Web users use such platforms to reinforce established offline relationships. These findings suggest that the offline social networks which a given person maintains are now being replicated within an online environment due to the availability of technology to support this replication. Three example Social Web platforms are now presented each containing social networking capabilities.

Facebook² is a well-established Social Web platform which allows users to build a profile consisting of their biographical information such as their name, their location and their

¹<http://www.wikipedia.org>

²<http://www.facebook.com>

interests and hobbies. Facebook facilitates the compilation of social networks by allowing users to add friends. The platform uses a “*walled garden*” principle where profile information is locked within the platform and can only be accessed by users of the site. Work by [LES06] found that the platform was primarily used for “*social searching*” where users would search for individuals they know offline and add them as friends, rather than establishing relationships with other users that they do not know offline.

The Social Web site MySpace³ primarily focuses on the sharing, discussion and participation in music. Users of MySpace are able to compile a profile containing the biographical information they wish to share and, like Facebook, social networks can be compiled by adding friends. MySpace offers an open service where the profile of a given user is accessible to web users who are not members of the site. According to [Cro07] MySpace contains profiles which are akin to alteregos of users and are not necessarily representative of the identity of a person, unlike profiles found on Facebook, therefore suggesting that users of MySpace are motivated by public perception due to the open nature of data access on the site.

Another example Social Web platform is the microblogging service Twitter.⁴ Twitter provides users with the functionality to publish microblogs which are commonly musings, thoughts or opinions, in a small message which the service defines as a “*tweet*”. Users are able to build a profile containing their biographical information, contact details and where they are located. Similar to the previous two Social Web platforms, users of Twitter are able to construct a social network within the service by “*following*” other users. This action establishes unilateral relationships indicating that the user is interested in seeing and reading another user’s tweets and interacting with them.

The aforementioned Social Web platforms give web users a presence which is visible in the public domain. How those users are perceived is influenced by the digital identity which they construct and what information they decide to publish. The following section discusses this notion of a digital identity and the related issues with its construction and dissemination.

2.4 Digital Identity

Before moving on to discuss digital identity, its role on the Web today and its various intrinsic properties, it is important to establish a precise and concrete definition of what identity is. Within this thesis identity is considered in the context of a person, that is the set of attributes and characteristics which collectively describe a given individual in a unique and distinct manner. The identity which a person pertains enables their identification and disambiguation from other similar individuals, for instance by observing additional information besides a person’s name. One can consider how an increase in the provision of identity attributes enhances the likelihood of identification, this is suggested in work by [Mar99] which describes a similar setting where *identifiability*, and thus a reduction in *anonymity* and *ambiguity*, is achievable through the provision of certain types of *identity knowledge* (e.g., legal name, locatability, pseudonym, social categorization, etc). Differing types of *identity knowledge* are more prone to ensuring identifiability over others, for example

³<http://www.myspace.com>

⁴<http://www.twitter.com>

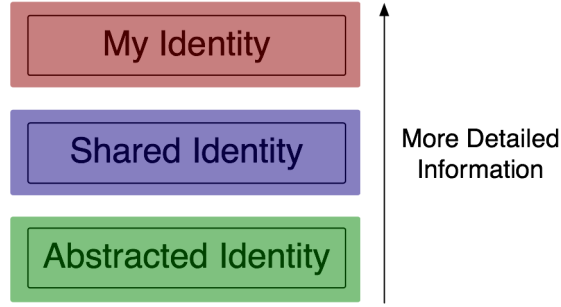


Figure 2.1: The tiers of Digital Identity from [Win05]

by observing a given person’s parents as a mechanism for circumventing name ambiguity in comparison with the legal name of the person.

In recent years the advancement of the Web and the provision of profile building functionalities on Social Web platforms such as Facebook, Bebo and Myspace, has enabled web users to create and shape an identity representation over which they are in complete control. Such an identity representation can be referred to as *intensional identity*, given that the dissemination of identity attributes is performed in a controlled manner and with purpose. The selection of what to show to others and how one’s identity should appear online is motivated by self-perception and how the individual wishes to be perceived by their peers. Work by [Bar07] observed that web users, when in control of their profile and how others will view them, tended to highlight their friendships with other popular individuals, thereby increasing the chances of the individual being accepted, or regarded as popular within a community or group. This suggests a disparity between *intensional identity*, constructed in a bespoke manner in a relatively short space of time, and *existential identity*: describing an individual in the real world, in an offline capacity. Managing this latter identity form is controlled only actions offline, and as such individuals have less control over how others may perceive them. Alterations in one’s existential identity take longer to filter through to peers, given that the changes are not observed instantaneously as they are online, where in some cases peers are notified directly of alterations via alerts and notifications (e.g., Facebook, Bebo, MySpace).

The variety of niche sites on the Web caters for a multitude of tastes and interests, allowing web users to assume alter-egos and in doing so maintain multiple identities, either by providing distinct pseudonyms in different platforms, or by discussing different topics in separate places. In the context of identity disambiguation, techniques are required which can cope with such a setting, given that web users wish to find those *nuggets* of information which are more valuable to them; those pieces of information which could cause harm to an individual if read and discussed by others (e.g., a person’s sexuality, state of health, sensitive identity information). Digital identity is defined in this thesis as the set of attributes that make a given person unique, in essence the collection of pieces of information which collated make up the profile of a particular individual.

In a similar manner to the types of *identity knowledge* set forth in [Mar99], work by [Win05] defined information relating to digital identity as being divisible into three tiers, shown in Figure 2.1. The first tier is called *My Identity*, which contains persistent identity information such as a person’s name, date of birth and genealogical relations, essentially

information that is constant and is unlikely to change. The second tier is called *Shared Identity*, which contains attributes assigned to an individual by other people such as the social network of a person. *Shared Identity* contains information which is susceptible to change as a person makes friends with different people and loses contact with others. The third tier is called *Abstracted Identity* and contains identity information derived from groupings and demographics. For example, identifying a person by a community of practice which they are involved in. Identity information within this tier is very likely to alter as a person's interests evolve over time.

Web users share identity information on Social Web platforms to enhance their experience when using the service. By sharing their personal information and building a profile which can be viewed by others, the chances of interaction with others increases. Sites such as Flickr, which are built around the sharing of photos, provides a richer user experience when the user of the site provides additional information about their interests and joins groups and participates with the community. This increases the network of the user and enables more information to be shared with them by the users of the site. In terms of Facebook, and similar Social Web platforms built upon interaction and communication paradigms, the motivation for sharing profile information is to interact with friends and ensure "*findability*". The recent Pew Report [MS10] found that 46% of web users searched for people from their past, coupling this figure with another finding from the same report that found 66% of internet users who had published their personal information had been contacted by someone from their past. Social web platforms provide a useful source for users to advertise themselves online, in effect replacing the telephone directory and becoming a new listings service.

2.4.1 Fragmented Identity

The growth of the Social Web has provided web users with a wide variety of Social Web sites and platforms to choose from. Each site offers different intrinsic functionalities and covers disparate topics of interest; therefore the profile a user maintains differs from service to service. In essence, the digital identity of a web user is fragmented between such platforms into different identity pieces. Identity information from the *Shared Identity* and *Abstracted Identity* tiers change the most between platforms, as users maintain different social networks and interact with a different community. For example, a web user may have a profile on the business-networking site LinkedIn⁵ and a profile on the social networking site MySpace. The person shares technical expertise and information on LinkedIn and shares music tips and discussions on MySpace. Each site contains a different facet of that person's digital identity, thereby providing fragmentation of his/her identity information across the Social Web.

One of the common characteristics of Social Web platforms is centralisation of data [mAYLL⁺09]. Centralisation occurs when a platform closes access to data so that it cannot be exported or reused by another platform, thereby ensuring that data is located in only one *central* place. As a result users must construct separate identity representations within different platforms, thereby fragmenting their identity across the Social Web. To gather a complete identity representation of a Social Web user, these identity pieces must be reconciled together.

⁵<http://www.linkedin.com>

2.4.2 Data Portability

Distinct Social Web platforms offer data portability using disparate methods and expose data using formats which are unique to the platform. Chapter 3 presents a discussion of the low-level characteristics of these formats and their limitations in greater detail. For now a high level explanation shall be given: a web user similar to the individual discussed in the previous example, may have a profile on two distinct Social Web platforms where each profile contains a different piece of his/her digital identity. Data can be returned by one platform in a structure which is tailored to the platform's data schema, whereas data exposed by another platform uses a different structure conforming to a different data schema. This limits the use of automated techniques to combine this data together in order to reconcile digital identity pieces, as automated processes are unable to interpret the different schemas and integrate the data effectively. This is comparable to differences in spoken languages and the need for translation into a mediator language to facilitate understanding.

The need for a common interpretation of social data is one of the central motivations behind the data portability project,⁶ allowing Web users to reuse *their* own data across platforms by using open standards and common agreed information formats. If automated disambiguation techniques are to make use of data obtained from Social Web platforms, then issues regarding data portability and identity fragmentation must be addressed. This requires methodologies to be investigated to reconcile fragmented identity pieces distributed across the Social Web by lifting information into a mediator format to support the integration process.

2.5 Online Privacy

Increased usage of Social Web platforms has influenced other web sites to offer similar interactive services and functionalities, allowing regular web users to share content and participate. As discussed previously, the rise in trust between the web user and web platforms has lead to such users now sharing an increasing amount of personal information within the public domain. It is also commonplace for previously offline services such as telephone directories, electoral registers and genealogical services to be made available online. Such platforms display biographical information about a range of individuals, such as their address and telephone number, and other personal information accessible from a user's profile page. These details can be readily exploited to commit identity theft, leading to serious implications for the unfortunate victim. The extent of the problem is reflected in worldwide government initiatives to educate the public against suitable practices online. For example the UK guidelines⁷ suggest against publishing “*...things like: phone numbers, pictures of your home, workplace or school, your address, birthday or full name*” in an online space due to the sensitive nature of such personal information.

The collation of certain identity attributes into a single profile provides sufficient details for the identity of an individual to be stolen and reused. A worrying by-product of the availability of personal information on the Web is the increase in the number of content farms which are now present. Such sites are deployed to aggregate information from distributed

⁶<http://www.dataportability.org/>

⁷http://www.getsafeonline.org/nqcontent.cfm?a_id=1132

web sites in to a centralised area. This is performed by sites such as ZoomInfo⁸ which has begun duplicating and aggregating personal information in a single location, thereby making it harder for web users to manually keep track of web resources containing their personal information, and more worryingly, providing a collection of web resources containing personal information.

Another malevolent web practice on the rise is lateral surveillance. The definition of lateral surveillance, according to [And05], is “*not the top-down monitoring of employees by employers, citizens by the state, but rather the peer-to-peer surveillance of spouses, friends, and relatives.*”, however this notion of surveillance being restricted to those who the subject knows directly is limited. In later work [And06] by the same author the definition of lateral surveillance is amended to incorporate the actions of employers vetted prospective employees, and describes the actions of partygoers who access information about possible dates. Within this thesis lateral surveillance is defined as the action of surveying and viewing a person’s profile or personal information which is visible on the Web, without the subject being aware. This notion of a lack of awareness leans towards the practice of online, or cyber-stalking, where the subject is obsessively viewed and/or followed on the Web. The opt-out policies implemented by various web sites allows information about a given person to be displayed until a request is made for its removal. From the perspective of the platforms which host such personal information, this policy decision is commercially just; by allowing people to view the service as interactive rather than restrictive, despite it reducing the privacy of users, and promoting lateral surveillance.

Those web users who choose to search the Web for their citations and then remove or hide any sensitive information perform a time-consuming and laborious task. As described in the introduction of this thesis, the cognitive process by which web users identify their web citations is to filter through all possible citations (i.e., by querying a web search engine) and then vet the returned results for citations (disambiguation). The application of automated techniques would alleviate this time consuming process and provide web users with a collection of web resources containing their personal information. Given the motivation behind performing this action: aversion of lateral surveillance, avoiding falling victim to identity theft, web resources which contain information describing a person’s real world identity pose a serious risk to the individual: particularly if they are unaware of their personal information being disseminated and therefore viewed without their permission. In order to automate the process of identifying such web resources, automated techniques require seed data which describes the real world identity of a given person. Producing such data manually is time consuming, for example requiring a form to be completed in order to capture the relevant details, instead such information could be leveraged from Social Web platforms. However as discussed previously, the identity which is constructed on a Social Web platform is controlled by motivational aspects such as self-perception which suggest a possible disparity between a person’s existential identity - their identity offline - and their identity on the Social Web. The utilisation of seed data from Social Web platforms is required to detect web resources which contain a person’s existential identity information - given the severe consequences should that information be collated and reused by others (e.g., identity theft). Therefore the following section compares these two identity representations via a user study, in essence looking for a correlation between the online and offline identities.

⁸<http://www.zoominfo.com>

2.6 User Study: Online and Offline Identities

The digital identity of a person defines how that individual is perceived within an online environment. [HRT⁺08] and [ESL07] stipulates that Social Web platforms contain social networks which are extensions of offline networks. Automated approaches to disambiguate *identity web references* require background knowledge, where the background knowledge is representative of a person’s real identity. Social Web platforms provide a source for such identity information based on the provision of features allowing web users to construct profiles and interact with their peers online. This section investigates the extent to which online identities constructed on a given Social Web platform mirror their offline/existential equivalents.

Referring back to the tiers of identity (§2.4), identity information falls into three tiers where the *shared identity* tier contains social network information. The social network which a given person maintains is an important aspect of disambiguating that person from others, based on the unique combination of friends and relationships. Therefore in order to compare the online identity that a Social Web user maintains with their real world identity, the overlap of offline and online social networks is analysed. This overlap provides a quantifiable measure of the extent to which online identities and real identities are correlated.

Online and offline social networks are compared in [SRWE08] through a similar user study. Participants of the study listed their top ten face-to-face friends - therefore resembling their real world social network - and their top 10 friends on instant messaging programs and Social Web platforms - thereby giving their digital social network. The authors then compared the overlap between these two networks. The results from the study found that 49% of participants’ real world social network was also present online and that 22% of the participants (16 people) had a online social network which contained all their listed real world relationships.

To provide the analysis of real and digital identity information a Social Web user study was conducted - similar to work by [SRWE08]. The results from [SRWE08] were not reused within this thesis due to the timeliness of the study and the sample used. In [SRWE08] the study was conducted in 2007, whereas the study presented hereafter was conducted a year later. The significant growth in usage of Social Web platforms during this period ensured that a sample taken a year earlier would not truly reflect an up-to-date analysis of the online identities maintained on Social Web platforms with their real world equivalent. Furthermore, the sample of analysis in [SRWE08] is taken from an urban university in Los Angeles in the United States, therefore it would be beneficial to the community to provide a sample from another country and therefore provide sufficient data for comparison of the two studies. The remainder of this section describes the experiment used for the study in detail, explaining the selection of participants, the experimental procedure and the results and empirical measures used for quantitative analysis.

2.6.1 Evaluation Measures

To measure the extent of the replication of real world social networks in a virtual environment, the information retrieval metrics precision and recall as described by [vR79] were used. Information retrieval metrics are designed for measuring the accuracy of information discovery systems and the classic definitions feature two sets of documents: A denotes the

set of *relevant documents* and B denotes the set of *retrieved documents*. Therefore precision and recall are defined as follows:

$$precision = \frac{|A \cap B|}{|B|} \quad (2.1)$$

$$recall = \frac{|A \cap B|}{|A|} \quad (2.2)$$

Such information retrieval metrics are well suited to the experimental setup used for this study. Imagine that the set of relevant documents, A , actually refers to the set of *relevant people*. When each participant defines his/her real world social network, this network characterises the most relevant people to that person. [vR79] defines relevance as being assumed to have a broader meaning of “aboutness” and “appropriateness” which given the context of the study suits the relevance of people to the participant. In a similar manner, one can consider the set of retrieved documents, B , to actually denote the set of *retrieved people*, which in the context of this study is the digital social network. Based on these adaptations of the document sets, the metrics are redefined as relevance and coverage.

2.6.1.1 Relevance

The definition of precision from [vR79] describes a measure which returns “...the proportion of retrieved information that is relevant...”. Given that in this study relevant information is the real world social network and retrieved information is the digital social network, precision can be redefined as a measure of the relevance of a participant’s digital social network as follows:

$$relevance = \frac{|real \cap digital|}{|digital|} \quad (2.3)$$

This metric of relevance therefore measures the ratio of strong-tied to weak-tied relationships within the digital social network, or rather, the proportion of the digital social network that is denoted by strong-tied relationships and therefore relevant. Strong-tied relationships are defined in [DB04] as “...the kind of ties that exist among close friends and families, the kinds of ties that connect dense clusters, are, in general, good sources for social support”, whereas weak ties are defined as “...the kinds of ties that exist among people one knows in a specific and limited context”. For instance, it is common for Social Web users to maintain a social network containing many weak-tied relationships. As each study participant lists his/her real world social network it is likely that they will list strong-tied relationships, given that these are the first that come to mind. Using the above metric derives a measure of relevance of the digital social network with respect to the real world social network.

2.6.1.2 Coverage

The definition of recall from [vR79] defines a measure which returns “...the proportion of relevant material that is actually retrieved...”. Given that relevant material is the real world social network and retrieved material is the digital social network, recall is redefined as a measure of coverage as follows:

$$coverage = \frac{|real \cap digital|}{|real|} \quad (2.4)$$

Coverage provides a measure of the extent of the replication of each participant's real world social network within a virtual environment. The intersection is derived between the set of people from the real world social network and the set of people from the virtual social network. The size of this intersection is then normalised by the size of the real world social network. The derived value returns a quantifiable measure of the proportion of the real world social network that is now replicated online, and thus to what extent the real world social network is covered by the digital social network.

2.6.2 Experimental Setup

The user study consisted of an experiment designed to provide an insight into the repetition offline social networks in an online environment. The behaviour of Social Web users was also analysed to investigate which members of their online social network participants interacted with the most. Participants for the user study were gathered by emailing all students and staff at the University of Sheffield requesting their participation in the experiment. The Social Web platform was chosen as a suitable platform from which to assess digital identity information - given that this is the most popular and widely used in the UK with 22 million users.⁹ Therefore when contacting participants it was specified that each participant must have an account with this platform. In total the user study consisted of 50 participants - 25 male and 25 female - covering a wide age range (18 - 45). The experiment was comprised of 3 separate stages as follows:

2.6.2.1 Stage 1. Real World Social Network Creation

The first stage involved each participant describing his/her social network in the real world. They were given an online form with 20 rows, one for each person in the participant's social network. Each row was completed with the name of a person, their age and the type of relationship (family, friend, colleague). More rows could be added should the participant wish to add more than 20 people. The intention of this step is for the participant to list his/her social network containing the strongest and most important (strong-tied) relationships, given that these will be the first that come to mind.

2.6.2.2 Stage 2. Digital Social Network Extraction

The second stage extracted the social network of each participant from Facebook using the Social Circular Facebook application,¹⁰ specifically developed for the study. Image and conversation data was analysed for behavioural trends: images were extracted that each participant featured in. For each participant images were analysed to find which other people appeared within the images. In order to analyse the conversation data, all messages each participant shared were extracted. In a similar manner to the image analysis, the messages were analysed to find which people had sent and received them.

⁹<http://www.clickymedia.co.uk/2009/10/uk-facebook-user-statistics-october-2009/>

¹⁰<http://apps.facebook.com/socialcircular>

2.6.2.3 Stage 3. Comparing Real World and Digital Social Networks

The final stage of the procedure involved the comparison of the two social networks. Each participant compared his/her real world social network from stage 1 with the digital social network from stage 2. If a person appeared in both the real world and digital networks then the participant flagged that person. In comparing the networks, the overlap in people present in both networks is derived and therefore gives a quantifiable measure of the replication of offline social networks in an online environment.

2.6.3 Results

2.6.3.1 Relevance and Coverage

The results from the user study (presented in Figure 2.2) show that, in general, the majority of each participant's real world social network is replicated in an online environment. Coverage provides a measurement of the extent of this replication, where the results show a range of coverage from 0.5 to 1 with an average coverage measure of 0.77. To place this in context, on average 77% of a given Social Web user's real world relationships are repeated in a virtual environment. This figure indicates that the digital identity representation constructed on a Social Web platform is now, to a large extent, representative of a web user's identity in real life. This differs from the results obtained from the user study in [SRWE08] which found that coverage - defined as *overlap* in their work - was only 49%. Several reasons could explain the difference in figures. The user study presented in this thesis was conducted in November 2008, whereas the user study in [SRWE08] was conducted a year earlier. During this time the uptake of Facebook, and Social Web platforms, has increased dramatically.¹¹ Also the user study in this thesis was conducted in the United Kingdom, whereas the study from [SRWE08] was conducted in the United States.

The results from the study also produced an average relevance measure of 0.23. To contextualise this measure, on average 23% of a given web user's online social network consists of important, strong-tied relationships and the remaining 77% are weak-tied relationships. One can imagine that online social networks, although intrinsically social in nature, do not contain a large number of important relationships. [GAH05] argues that relationships found on Social Web sites form part of much vaster social networks, and are mainly classed as weak-ties; where the two people are acquainted and nothing more.

As the results in Figure 2.2 demonstrate coverage tends towards 0.9 as the online social network increases in size. Although the adoption of Social Web platforms by web users has become a mainstream phenomena, there are still users who resist following their friends and peers in signing up to such services. The results show a logarithmic regression [Dar90] model best fit the trends of the data, where a nonlinear relationship exists between the size of the social network and the level of coverage: the larger the virtual social network becomes, the more relationships it contains, strong-tied and weak-tied, therefore the likelihood of the real world social network overlapping with the digital social network increases. Conversely, when the social network grows in size relevance is reduced: as the number of weak-tied relationships within the network gradually increases, the ratio of strong-tied relationships to weak-tied relationships grows.

¹¹<http://techcrunch.com/2009/09/15/facebook-crosses-300-million-users>

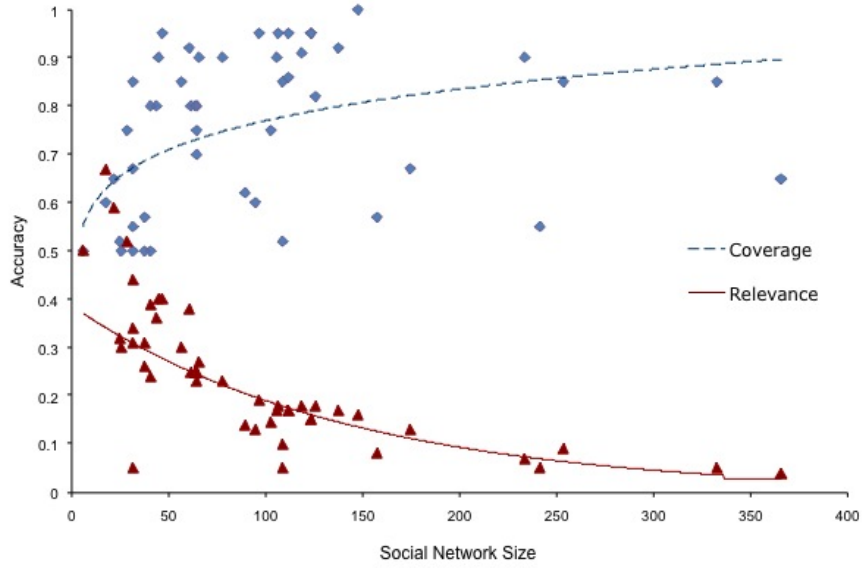


Figure 2.2: Relevance and Coverage measures from social network comparisons

2.6.3.2 Demographics of Coverage

Deeper exploration of the collected data also provides an analysis of the portion of the real world social network which is replicated online. This portion of the covered social networks is queried for the relationship types which it is comprised of. Recall from the description of the experiment that when compiling their real world social networks, participants were given a form in which they selected the type of relationship which they had with their social network member - choosing from either *coworker*, *friend* or *family*. Figure 2.3 shows the breakdown of the distribution of these three relationship types. The majority of real world relationships which are replicated online - and are therefore strong-tied - were with friends (62%). Family and coworkers both made up much smaller portions of the covered social network - found to be 24% and 14% respectively.

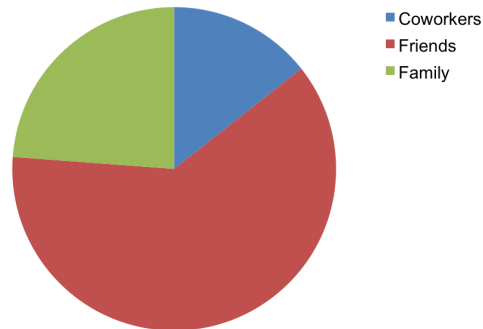


Figure 2.3: Distribution of relationship within the overlap between real world and digital social networks

Further analysis of the data found that for 68% of relationships from the real world social networks which were with coworkers were replicated online. For friends 89% of strong-tied

relationships were maintained in a digital social network and 73% of family relationships listed in the real world social network were replicated online. Such findings suggest that the demographic of strong-tied relationships within a digital environment are largely driven by interactions with friends rather than coworkers or family members. Using the collected data from the experiments the age demographic was also analysed within the portion of the real world social networks which were covered online. Figure 2.4 shows the breakdown of these ranges. Given that the participants within the study were largely students who used Facebook regularly on campus, it is unsurprising that the largest age group within the covered portion of the network consisted of the lowest age band tested (< 21). Recent statistics¹² published on the demographics of Facebook usage found the largest age group to use the platform were > 35 . Therefore the age demographic of the study sample, given that this is composed of individuals associated with a university, is not indicative of the age grouping of Facebook overall. It is worth noting also, that the results showed an increase in the portion of the real world social network replicated online as age decreased, thereby indicating a propensity of younger generations to maintain important offline relationships online.

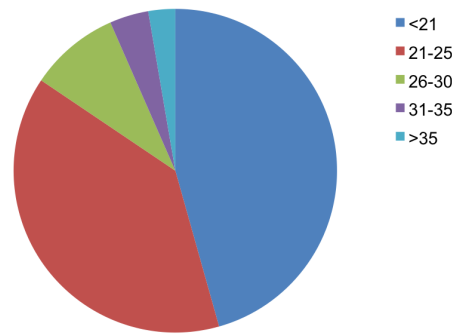


Figure 2.4: Distribution of ages within the overlap between real world and digital social networks

2.6.3.3 Behaviour

The behaviour of each participant within their social network was analysed by collecting image and message data. Figures 2.5 and 2.6 present the results of the analysis. Images featuring each participant were analysed to see which other people appear in those images, therefore deriving a cumulative count for each member of the participant's social network. A similar approach was used for analysing the messages sent and received by each participant to count how many times each social network member had shared a message with the participant. The results from this analysis demonstrate a trend in the behaviour of the participants to share a low number of messages and images with a large number of people. This interaction could be contextualised as a web user sharing a rare conversation with an acquaintance. The results also show how each participant shares a high number of images and messages with very few people - depicted in the tail of the graphs and its low frequency levels.

¹²<http://royal.pingdom.com/2010/02/16/study-ages-of-social-network-users/>

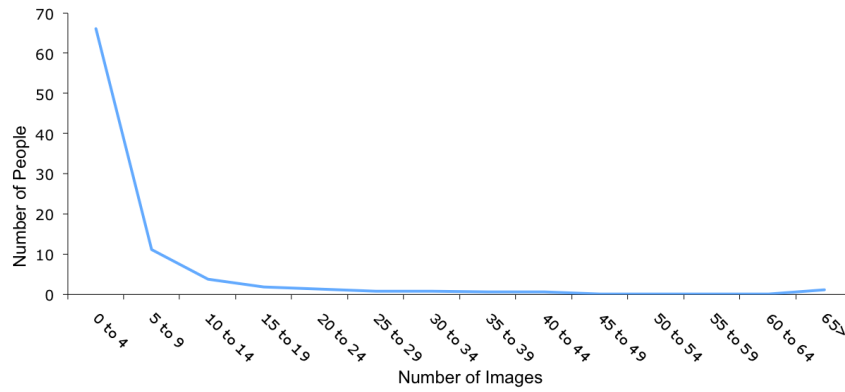


Figure 2.5: Analysis of image sharing within digital social networks

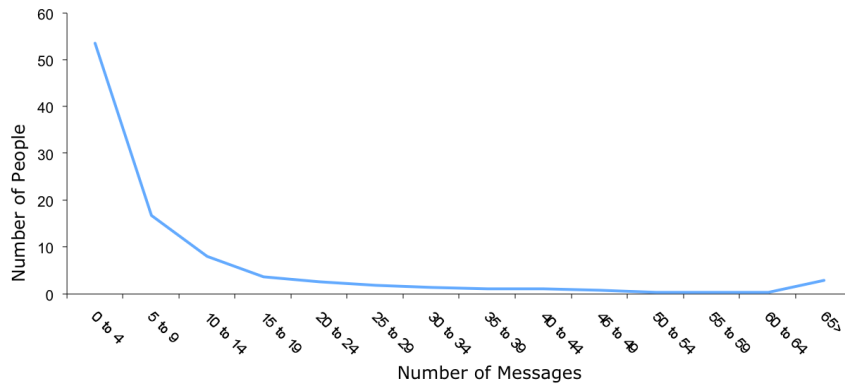


Figure 2.6: Analysis of message sharing within digital social networks

The tail of the graphs in Figures 2.5 and 2.6 represents those people within a person’s online social network with whom the person has a relationship incorporating frequent interactions. Those relationships are maintained by the need to communicate often in an online environment by sharing images and sending and receiving messages. When analysing the tail of each graph (> 9 for images and messages), on average, 92% of the people who feature in the tail are members of the real world social network. This indicates that the tail is representative of a given web user’s real world social network and contains strong-tied relationships. Moreover, given that the supplied real world network is strong tied relationships, the level of interaction offline is also likely to be repeated online, thereby indicating that the online environment in which those users interact is actually an extension of their offline environment. This suggests that Social Web platforms provide a useful source for data which is indicative of the real identity of users of such platforms.

2.7 Summary

This chapter has described the Social Web, explaining how it emerged and how it has affected the role of web users. Social Web platforms have allowed web users to construct an online identity, which as the results from the previous user study indicate, is synonymous with their real world identity. These results support one of the claims within this thesis that data

found on Social Web platforms is representative of real identity information.

This chapter has discussed the fragmentation of the digital identity across the Social Web and the issue of portability of this data. In order to address these issues chapter 5 presents work investigating how data can be exported from Social Web platforms and interlinked together for use as background knowledge, or seed data, to support automated disambiguation techniques. Semantics Web technologies are used to aid the aggregation of data from distinct sources by providing machine-readable metadata descriptions. The following chapter presents the Semantic Web and discusses how digital identity information can be defined with semantics.

Chapter 3

Digital Identity and the Semantic Web

3.1 Introduction

Using automated techniques to disambiguate *identity web references* on the Web requires information found within heterogeneous web resources to be parsed and a common interpretation achieved. Machines are limited in their ability to interpret data, instead explicit declarations must be provided which describes the meaning of the data. For instance, if a web page contains the name of a company then a machine is unable to know that a given sequence of characters or words within the page represents the company name without being informed of the semantics of the sequence. The Semantic Web provides technologies to make this declaration possible. This chapter presents an overview of the Semantic Web, describing how metadata models can be constructed to represent the underlying knowledge structures of web resources.

Metadata models reference concepts within an ontology, which provide a reusable conceptualisation of a domain of knowledge [BLHL01]. As the Semantic Web has grown so too has the number of ontologies produced, shared and combined together. Several widely used ontologies are now available for defining the semantics of digital identity information, therefore allowing machines to interpret data within a web resource as denoting a person's name or address. This chapter discusses the need for defining the semantics of digital identity information within the context of this thesis and presents several in-use digital identity ontologies.

This chapter is structured as follows: section 3.2 presents an overview of the Semantic Web describing knowledge representation technologies, ontology languages, rules and Linked Data. Section 3.3 describes how the semantics of digital identity information can be defined by presenting several ontologies which conceptualise digital identity. Section 3.4 provides a summary of the chapter.

3.2 The Semantic Web

3.2.1 Origins

The first mention of the Semantic Web was heard in 1994 when the founder of the World Wide Web, Tim Berners-Lee, gave a plenary talk¹ about future directions for the World Wide Web. Berners-Lee described problems with the current form of the Web as a flat document web linking together web pages via hyperlinks. Information within each web page was easy for humans to understand but difficult for machines. The World Wide Web, at this point, was emerging as a mass network of web pages linked together without meaning. By attaching semantics to web pages, machines would be able to *understand* information within those pages and do *intelligent* things with the data; the Web would become a *Semantic Web*.

Fast-forward seven years to 2001 and [BLHL01] described a concrete vision for the Semantic Web which would result in a decade of research towards pursuing the realisation of this vision. The vision presented a future scenario where software agents would be able to interact and exchange information in order to complete complex tasks. For example, booking appointments with a physiotherapist by liaising with other agents, checking available diary times for the appointments and sharing information in common formats. This vision encapsulated three key areas of work for realising a Semantic Web:

1. Express meaning
2. Ontologies
3. Agents

The intention of the Semantic Web is to allow the expression of meaning within web resources by encoding semantics within those resources. Web 1.0 was the document web: information could be parsed by humans but not machines. Around 2005 the evolution into Web 2.0 saw the migration of users from consumers to citizens and provided web users with the ability to interact socially. Web 2.0 sites and platforms began to allow access to data using APIs, which allowed web developers to combine data from multiple sites to create new and innovative web applications. One of the limitations of Web 2.0 is the lack of common semantics to define the meaning of data. For example, Web 2.0 platforms return data as structured responses, but with no common definition of what the structure means.

The Semantic Web is the next phase in this web revision cycle by providing an Intelligent Web: structured data is provided with a definition of what the structure means. Information from heterogeneous sources is encoded with common semantics thereby facilitating the *understanding* of such information by machines. For automated techniques to interpret web resources, the knowledge present within the resources must be represented in a machine-readable format. The following subsection presents several technologies to facilitate this representation.

3.2.2 Representing Knowledge

Before moving on to explain how knowledge can be represented, it is important to consider what knowledge actually is: the definition of knowledge was interpreted by [Ack89] in rela-

¹<http://www.w3.org/Talks/WWW94Tim/Overview.html>

tion to data, and information. Data can be regarded as raw letters and characters which lack context, while information is data enriched to have some significance and meaning through contextualisation. Knowledge is information and data, which is understood through semantics - declaring the explicit meaning of something through a widely understood means, this allows complex questions to be asked and answers provided. In the context of this thesis knowledge is defined as information combined with metadata, thereby allowing automated processes to understand what information means given its semantics. Several technologies exist for representing knowledge as follows.

3.2.2.1 XML

The Extensible Markup Language (XML) allows knowledge to be defined using metadata models through the use of user-defined markup [GP03]. Data is surrounded by bespoke tags, or elements, producing metadata which provides data with meaning. By using extensible tags the developer defining the XML can specify bespoke tags for a specific purpose. XML offers a structured format for information, therefore enabling information to be transported and reused more easily. However, exchanging information between heterogeneous sources using XML has limitations. For instance, XML is limited when the vocabulary of tags used by one party may differ from the vocabulary of tags used by another party. If an XML document is exchanged between the parties, the recipient may not be able to interpret metadata used by the sender. The meaning of tags may be obvious to humans, however a machine is limited in interpreting such metadata. Another limitation of XML is the ambiguity of XML element names: element names within one vocabulary may be identical to element names used in another vocabulary, yet refer to different concepts.

3.2.2.2 XML Schemas and Namespaces

To alleviate ambiguity in element names within XML data models, XML schemas and XML namespaces [BHLT06] can be used. An XML schema defines the structure that an XML model must adhere to, placing constraints on the hierarchy of elements, the values the elements may contain and the naming of the elements. An XML namespace provides a unique identification of an XML element through a Uniform Resource Identifier (URI). The URI points to the element definition in a given XML schema. When XML is transferred between sources, the recipient is able to interpret the elements given the namespace of the schema defining the structure of the model.

3.2.2.3 RDF

The resource description framework (RDF) [MM04] provides a framework for describing metadata models containing machine-processable semantics. RDF consists of three main concepts: resources, properties and statements [FHLW03]. Resources depict an object or thing (e.g., a person) identified by a URI. Properties describe the attributes and relations of a resource. Statements consist of a resource, a property and a value of that property which describes the resource. Statements denote a binary ground predicate (e.g., *mother(X,Y)* describes X as the mother of Y) and can be expressed as a triple consisting of a subject and object linked together by a predicate:

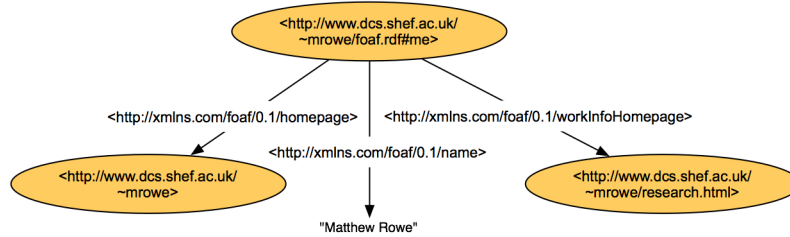


Figure 3.1: RDF graph structure

<subject, predicate, object>

Statements can be linked together such that the object of one triple is the subject of another. To this end RDF provides a graph structure which overcomes the limited hierarchical structure of XML to offer a more flexible and expressive representation. URIs are used in RDF, like XML, to provide unambiguous references to concepts within ontologies and available resources. For example, consider the following triple:

```
<http://www.dcs.shef.ac.uk/~mrowe/foaf.rdf#me>
  <http://xmlns.com/foaf/0.1/name>
    "Matthew Rowe"
```

A machine reading this triple would interpret the subject as a resource, or individual, identified by a URI which in this case defines a person. The resource has a property defined within the FOAF ontology identified by a URI, and this property has the value assigned within the object of the triple which in this case is a string literal (literals are primitive values such as a string or an integer). The reader must also understand that there are cases where a resource is not given a URI, in such a scenario a resource is referred to as being a *blank node* or an *anonymous node*. Triples can still be related to the resource, but machines can only parse and interpret such related information within a local context. Referring back to the previous triple, and the resource which the triple is related to, additional information can be assigned to the resource, or person in this case:

```
<http://www.dcs.shef.ac.uk/~mrowe/foaf.rdf#me>
  <http://xmlns.com/foaf/0.1/name>
    "Matthew Rowe"
<http://www.dcs.shef.ac.uk/~mrowe/foaf.rdf#me>
  <http://xmlns.com/foaf/0.1/homepage>
    <http://www.dcs.shef.ac.uk/~mrowe>
<http://www.dcs.shef.ac.uk/~mrowe/foaf.rdf#me>
  <http://xmlns.com/foaf/0.1/workInfoHomepage>
    <http://www.dcs.shef.ac.uk/~mrowe/research.html>
```

The three statements combined form a graph representing information about the person, or resource. As Figure 3.1 shows, the graph formed around the resource defines a person as having a homepage located at the defined URI, and a work information page defined by another URI. This structure can be extended to form a web of machine-readable information. Resources within web pages represent instantiations of objects with properties describing additional information about the resource. Relations can be specified between resources to capture the semantics of the relation. For example consider this example encoded using RDF/XML syntax, note the XML namespaces:

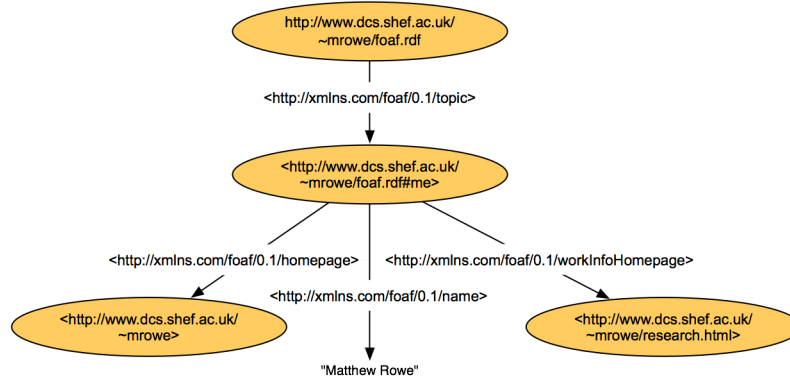


Figure 3.2: RDF graph structure with additional information

```

<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:foaf="http://xmlns.com/foaf/0.1/">
  <foaf:Person rdf:ID="me">
    <foaf:name>Matthew Rowe</foaf:name>
    <foaf:homepage rdf:resource="http://www.dcs.shef.ac.uk/~mrowe"/>
    <foaf:workInfoHomepage
      rdf:resource="http://www.dcs.shef.ac.uk/~mrowe/research.html"/>
  </foaf:Person>
</rdf:RDF>

```

There are several other syntaxes for RDF such as Turtle [BBL08], Notation 3 [BL06] and RDFa [Gro08]. When RDF is published on the web as a document, the document is assigned a URI. Resources within the document, and within the subsequent RDF model, are accessible through a hash URI [SCV07], by pointing to a specific resource within the document. For example, Figure 3.2 shows the graph structure of the above RDF/XML snippet published as a document on the web and can be found at <http://www.dcs.shef.ac.uk/~mrowe/foaf.rdf>. A resource exists within the document describing a person, identified by the internal identifier `me`. This identifier is concatenated to the URI of the document to provide a referenceable hash URI, referring to the resource: `<http://www.dcs.shef.ac.uk/~mrowe/foaf.rdf#me>`. A machine provided with the URI of the person could access the RDF/XML document on the Web and retrieve the relevant information about that person from within the document, gathering the graph connected to the resource (such as the graph from Figure 3.2).

3.2.2.4 RDFa

Web documents encoded as RDF/XML provide a metadata model with explicit semantic descriptions. Information within the document, although in some cases human-readable, is primarily intended for machine-processing. One of the intentions of expressing meaning within web documents as defined in [BLHL01] was such that “...*semantics were encoded into the Web page ... using off-the-shelf software for writing semantic web pages*”. To this end web pages were intended to be both human-readable and machine-readable. Web pages encoded using the Hypertext Markup Language (HTML) allow web designers to style information within a page so that information is displayed to the user in a legible manner. The often poorly-formed representation of HTML makes it hard for machines to parse and glean

knowledge from such web pages. The Extensible Hypertext Markup Language (XHTML) extends HTML by making web pages well formed and therefore able to parse using automated processes. Using the Resource Description Framework in Attributes (RDFa) [Gro08] semantics can then be added to XHTML elements within a given web page. Just as in the definition of RDF models in RDF/XML notation, RDFa uses namespaces to link to ontology concepts through a URI. Concepts from the ontology can then be added to an element within an XHTML document:

```
<div xmlns:foaf="http://xmlns.com/foaf/0.1/" typeof="foaf:Person">
  <p><span property="foaf:name">Matthew Rowe</span></p>
  <p><a rel="foaf:homepage" href="http://www.dcs.shef.ac.uk/~mrowe">Homepage</a></p>
</div>
```

As the example shows, ontology concepts are added to the attributes of XHTML elements within the page without altering the page's look or feel. RDFa is an example of *lowercase* semantics [Kha06], where existing information in the page is raised to knowledge through the addition of simple concept definitions. By adding semantics, explicit knowledge structures can be embedded within an XHTML document, relating information together and allowing both humans and machines to process the resource.

3.2.3 Ontologies

Knowledge can be represented through the use of metadata models, where metadata attributes meaning to data. As shown in the previous subsection, the definitions used for metadata correspond to concepts from an ontology. An ontology provides a formal conceptualisation of a domain of knowledge [BLHL01]. The most basic definition of an ontology is a vocabulary of terms used in a particular domain and the definition of relationships between those terms. Original definitions of an ontology date back to philosophers who considered an ontology to be a metaphysical representation of reality. In essence this representation encapsulates a perspective or view of the world in a formal way. This encapsulation according to [FHLW03] is vital in sharing a common understanding of a domain between people and heterogeneous information sources.

Within the computer science domain, ontologies consist of machine-processable information defined using formal constructs. A machine does not *understand* the ontology as humans might, it is merely able to *interpret* the terms within the ontology and the relationships between those terms. Formal constructs within the ontology place restrictions on relationships and how terms can be related to one another. Schemas differ from ontologies by defining the syntax and structure of documents through placing constraints on how a document may be constructed. Ontologies however, describe domains at a conceptual level by describing the theory of the domain, and not merely the structural and syntactic information.

According to a recent survey featured in [Car07] of people using ontologies in their work or research, 69% used ontologies to share a common understanding of information, 56% used ontologies for the reuse of domain knowledge and 36% used ontologies to make assumptions about a domain explicit. By defining a domain formally, ontologies have a vital role in sharing a common understanding of the domain and what a given piece of information means. In order to formally represent the concepts of a given domain, several common

ontology languages are in use on the Semantic Web. Two of the most popular languages (according to [Car07]) will now be discussed detailing their advantages and limitations with respect to one another.

3.2.3.1 RDF-Schema

The Resource Description Framework Schema (RDFS) [BG04b] provides an ontology language built on RDF/XML notation for describing resources and relationships between resources. RDFS forms the basic building block for ontology languages used within the Semantic Web community. Classes and properties of those classes are defined by *rdfs:Class* and *rdfs:Property* respectively. A property within an ontology has a domain (the subject) and a range (the object). In RDFS the domain is defined using *rdfs:domain* and the range is defined using *rdfs:range*. Class hierarchy is defined using the *rdfs:subClassOf* property to relate a parent class to a child class. Property hierarchy is also defined in a similar way using *rdfs:subProperty* to relate a parent property to a sub property. An instance, or individual, is an instantiation of a *rdfs:Class* class.

Constructing an ontology using RDFS has several notable limitations. Describing relations between properties is limited to a basic hierarchy. This is sufficient for basic ontologies describing a simple nested domain such as a taxonomical structure, but limited when more complex conceptualisations are required. In RDFS, classes cannot be defined as disjoint with one another. For example, modelling two classes within an ontology such as *ex:Man* and *ex:Woman* using RDFS would not allow the expression of disjointness between those classes. Cardinality constraints cannot be placed on properties within an RDFS ontology, therefore limiting quantification of property values, and the characteristics of properties cannot be explicitly defined (such as denoting intrinsic behaviour of properties). In short RDFS provides a useful ontology modelling language, yet lacks the expressivity needed to model more complex domains of knowledge.

3.2.3.2 OWL

The Web Ontology Language (OWL) [SWM04] was created from the need for a more expressive ontology language [AvH04]. OWL provides reasoning power over ontological constructs by using formal semantics. OWL is similar to RDFS in several ways; it uses the same RDF/XML notation form, defines classes and properties of classes and borrows hierarchical class and property definitions from RDFS.

One advantage of OWL is the ability to import external ontologies into an OWL ontology, therefore allowing concepts defined in one ontology to be reused in another. OWL defines two types of properties: a property of type *owl:ObjectProperty* defines a relation between two classes and a property of type *owl:DatatypeProperty* defines a relation between a class and a literal value (string, integer). OWL allows cardinality restrictions to be placed on properties by setting the quantification of the range. Another advantage of using OWL is the definition of characteristics of properties: properties can be set to be functional, inverse functional, transitive and symmetric. In comparison with RDFS, modelling using OWL allows a much more expressive encapsulation of a domain than simple hierarchical relations. OWL also allows axioms to be defined within an ontology. An axiom places restrictions on the use of concepts through chains of statements which must hold for a given concept to be

used. For example, for *mother*(*X*, *Y*) to hold and be true, *female*(*X*) and *parent*(*X*, *Y*) must also hold and be true. The foundations of axioms lie with rules and first order logic which are discussed in the following subsection (§3.2.4).

Equivalent resources referenced by different URIs can be identified as being equivalent using the *owl:sameAs* predicate, thus providing an identity link between two resources, or individuals. An example of this would be where the same person appears in separate metadata models. Conversely distinctly non-equivalent resources can be expressed using *owl:differentFrom* thereby defining disjoint resources. The use of *owl:sameAs* has been an essential linking predicate for the Linked Open Data community to enable equivalent resources in disparate datasets to be linked together.

3.2.4 Rules

Rules provide a means of logically inferring statements based on the presence of knowledge. Richard Jeffrey [Jef89] states that:

“Logic is the science of deduction. It aims to provide systematic means for telling whether given conclusions do or not follow from given premises.”

Ontologies are ground in the use of first-order logic (FOL) or predicate logic. This provides logic based around unambiguous statements and the ability to derive a conclusion from a set of observed premises [AvH04]. This observed set provides the reasoning behind the derivation of a conclusion, otherwise known as a proof. Rules provide a systematic means to analyse a set of premises, and if they hold and are true, logically derive a conclusion which must therefore also hold and be true. Rules follow a standard construct comprised of the antecedent (tail) and the consequence (head) as follows:

$$P_1, P_2, \dots, P_n \rightarrow Q$$

Rules form an important component of the Semantic Web by allowing inferences to be made based on an existing knowledge model. This forms the basis from which reasoning over data can function. Imagine an ontology models the genealogical relationships between people. If a person is male, and the person has a parent and the parent has a sibling, then one can infer that the person is the nephew of the parent’s sibling. In predicate logic a rule is defined as:

$$parent(Y,X), male(X), sibling(Y,P) \rightarrow nephew(X,P)$$

Premises in the antecedent of the rule can be defined using triples containing concepts from an ontology. This allows rules to be applied to a metadata model to order to make inferences from knowledge in the model. For instance, the above rule could be applied to a metadata model containing information about a given person’s family tree:

```

<foaf:Person rdf:ID="matthew">
  <foaf:name>Matthew</foaf:name>
  <gen:parent rdf:resource="#derek"/>
  <foaf:gender>male</foaf:gender>
</foaf:Person>
<foaf:Person rdf:ID="derek">
  <foaf:name>Derek</foaf:name>
</foaf:Person>
<foaf:Person rdf:ID="peter">
  <foaf:name>Peter</foaf:name>
  <gen:sibling rdf:resource="#derek"/>
</foaf:Person>

```

The inferred statement from the rule, given that the premises are represented as triples, relates *#matthew* as being the nephew of *#peter*. Various rule syntaxes exist including RuleML,² SWRL [HPB⁺04], Jena Rules³ and SPARQL rules [PS06].

3.2.5 SPARQL

RDF provides the back bone for knowledge representation on the Semantic Web. In order to query such formats a language was created called SPARQL [PS06]. SPARQL functions by matching patterns within the graph-structure of RDF. When querying a given RDF graph SPARQL allows the inclusion of variables for elements in a given triple. A SPARQL query consists of two parts which are denoted as *clauses*. The **SELECT** clause contains the output of the graph query, in the same manner as SQL *selects* elements from records within databases, variables are included within this clause which are bound to the patterns specified within the **WHERE** clause. This clause contains the triples which match elements within RDF graphs, allowing multiple graphs to be queried at one time.

The application of SPARQL is best illustrated through an example. Referring back to the RDF model of a given person's family tree in the previous section, a SPARQL query can be constructed to retrieve all the person names from the graph as follows:

```

PREFIX foaf:<http://xmlns.com/foaf/0.1/>
SELECT { ?name }
WHERE {
    ?x foaf:name ?name
}

```

SPARQL queries can also contain literals and resources within the **WHERE** clause to match information within the graph structure. This allows URIs to be included within a query - given that URIs denote unique resources within the Semantic Web - therefore allowing information to be retrieved. SPARQL can also be used as a rule language by altering the **SELECT** clause and replacing it with a **CONSTRUCT** clause, allowing the variables bound within the **WHERE** clause to be bound to variables within the **CONSTRUCT** clause. Using the general architecture of rules - as described previously - one can regard the **WHERE** clause as denoting the antecedent of a rule - containing the necessary triples which must be matched - and the **CONSTRUCT** clause as denoting the consequence of a rule - containing the inferred premise from the graph pattern.

²<http://ruleml.org/>

³<http://jena.sourceforge.net/inference/>

Referring back to the rule within the previous section which infers a *nephew* relationship between two people within a given metadata model, this rule can be written as a SPARQL CONSTRUCT query as follows:

```
PREFIX foaf:<http://xmlns.com/foaf/0.1/>
PREFIX gen:<http://example.com/gen>
CONSTRUCT { ?x gen:nephew ?p }
WHERE {
    ?y gen:parent ?x .
    ?x foaf:gender "male" .
    ?p gen:sibling ?y
}
```

At the time of writing this thesis a proposal had been put forward to enhance SPARQL to include *update* functions [SMB⁺08]. SPARQL was primarily intended as a language for reading RDF graphs, however its widespread adoption and flexibility has lead to a growing belief that it should also be used to modify RDF graphs - in the same way as SQL modifies records in relational databases. The two *update* clauses will include **INSERT** and **DELETE** which allows triples to be written into a given graph structure and triples removed.

3.2.6 Linked Data

The Semantic Web was devised to offer a web of data interlinked by relations where the meaning of those relations is defined through machine-readable semantics [BHAR07]. By providing a web of data defined using machine-processable semantics [HRH08] software agents are able to employ a “*follow-your-nose*” (FYN)⁴ strategy to traverse this web by moving between datasets and sources, and discover additional knowledge about particular resources. For instance, work by [RSS08] has concentrated on the interlinking of music datasets. By interlinking such information from heterogeneous sources, resource definitions (artists, albums, etc) become enriched with additional knowledge provided within separate locations. An agent finding an artist within one dataset would be referred to additional knowledge about the same artist in another dataset.

The Linked Open Data⁵ project has investigated the linking of structured content on the Web by interlinking datasets published as RDF from distributed sources. Linked datasets are lifted from relational databases and XML, and converted to RDF models, thus providing information with semantics. Work by [Hau09] explains steps to leverage existing data stored within databases, or as basic XML, to RDF through Linked Data principles and the selection of the correct ontologies to describe the data conceptually. Linked Data follows four key principles:⁶

1. **Resources must be described using URIs:** in order to provide a unique identity for the resource.
2. **URIs must be dereferenceable:** following the URI will return some information about the resource it describes.
3. **Provide useful information when following a URI:** the returned information must be in a machine-readable format (i.e., RDF).

⁴<http://inkdroid.org/journal/2008/01/04/following-your-nose-to-the-web-of-data/>

⁵<http://linkeddata.org/>

⁶<http://www.w3.org/DesignIssues/LinkedData.html>

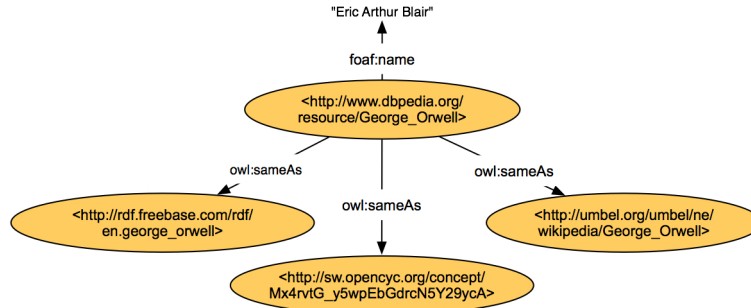


Figure 3.3: Linked Data relating resources defining the author George Orwell in disparate datasets

4. **Provide links to additional URIs:** relate resources to additional descriptions or related content.

Applications are able to *consume* Linked Data and combine it with additional external datasets to offer novel and innovative functionalities, clearly influenced by the ability of web developers to ‘ *mashup*’ data from disparate Web 2.0 sites. As more Linked Datasets are produced, the need for the detection of related resources in separate datasets grows. Interlinking between datasets involves discovering related resources and providing a relation between those resources defined using the relevant predicate; e.g., *owl:sameAs* for equivalence. Figure 3.3 demonstrates how a given resource representing the author George Orwell from the DBPedia dataset is linked to additional resources which are defined as representing the same entity. Accurately discovering these related resources is an ongoing research challenge and is investigated within this thesis in chapter 5 when reconciling digital identity fragments from the Social Web.

Producing Linked Data on a large-scale has been investigated by the DBPedia project⁷ by extracting structured information from Wikipedia and converting this information into RDF in order to support machine-readability [ABK⁺08]. The exposure of large-scale Linked Datasets has allowed applications to be built that provide services based on the knowledge within such datasets. For instance, DBPedia Mobile as presented in [BB08], offers a mobile device application capable of combining the location of the device user with knowledge from DBPedia to suggest nearby points of interest. Descriptions of those points of interest also harvest additional knowledge through the provision of links to information in additional datasets.

3.3 The Semantics of Digital Identity Information

As the Semantic Web has grown so too have the number of ontologies and vocabularies used to capture knowledge domains. Several ontologies and vocabularies are in-use to define digital identity information and the remainder of this section presents examples of these.

⁷<http://www.dbpedia.org>

3.3.1 Friend of a Friend (FOAF)

The Friend of a Friend project⁸ investigates how Semantic Web technologies can be used to describe people and the relationships between people. As a result of the project's work an ontology [BM07], modelled using OWL, was produced that conceptualises people and their relationships on the Web. The project began in 2000 and the first ontology was released not longer after the project's commencement. To date the FOAF ontology has become one of the most widely used stable ontologies on the Semantic Web. Particularly since the emergence of the Social Web, FOAF has provided a useful means for formalising the semantics of digital identity information and has allowed machine-readable metadata to be exported and reused from Social Web platforms. Current techniques and research investigating these practices are explained in chapter 5. The FOAF ontology describes two areas of digital identity information: biographical information and social network information.

A person concept in the FOAF ontology represents a resource on the Web and is expressed as an instance of the *foaf:Person* class. Biographical information is assigned to the person using properties for the name, gender and date of birth of the person. Resources such as the homepage, weblog, OpenID and email address of the person are also related to the *foaf:Person* instance using *owl:ObjectProperty* properties. As mentioned previously, OWL allows characteristics to be assigned to properties within an ontology, and in this instance the properties are typed as *owl:inverseFunctionalProperty*. This characteristic denotes a one-to-one relationship between the person, and the resource describing him/her. The resource therefore denotes a unique identifier of the person such that no other person can have the same email address, homepage, openid or weblog.

Relationships between people are expressed using the *foaf:knows* property. This property defines one person knowing another person in the most general sense of the term. For instance, *Alice* knows *Bob* doesn't express that *Alice* and *Bob* are friends, but that they are merely acquainted in some way. Digital identity information can be expressed using FOAF semantics as follows:

```
<foaf:Person rdf:ID="me">
  <foaf:name>Matthew Rowe</foaf:name>
  <foaf:givenname>Matthew</foaf:givenname>
  <foaf:family_name>Rowe</foaf:family_name>
  <foaf:gender>male</foaf:gender>
  <foaf:img rdf:resource="http://profile.ak.facebook.com/s61109237.jpg"/>
  <foaf:holdsAccount>
    <foaf:OnlineAccount>
      <foaf:accountServiceHomepage rdf:resource="http://www.facebook.com/" />
      <foaf:accountName>61109237</foaf:accountName>
    </foaf:OnlineAccount>
  </foaf:holdsAccount>
  <foaf:interest>Football</foaf:interest>
  <foaf:interest>music</foaf:interest>
  <foaf:knows>
    <foaf:Person>
      <foaf:name>Sam Chapman</foaf:name>
    </foaf:Person>
  </foaf:knows>
</foaf:Person>
```

⁸<http://www.foaf-project.org>

Work by [Dow05] predicted that social networking sites would offer the functionality to create FOAF files easily and automatically from the site's metadata, however this has not yet come to fruition. FOAF has become a widely used ontology and a standard which other similar ontologies are compared against. The blogging web site LiveJournal⁹ offers the facility for each user to export their information from the site as RDF using FOAF. However, Social Web sites have not used FOAF to describe exportable metadata, instead such sites expose XML using XML schemas that are specific to the platform, thereby restricting a common interpretation of information from heterogeneous sources.

3.3.2 Semantically Interlinked Online Communities (SIOC)

The Semantically Interlinked Online Communities (SIOC) project¹⁰ investigates how online communities can be conceptualised. The goal of the project is to produce links between communities in disparate locations on the web. To this end Semantic Web technologies allow the definitions of ontologies to model such links. As described in chapter 2, the changing role of the web user has lead to users participating in web communities in some form such as forums, chat rooms, newsgroups and blogs. Each community is interpreted as a single island of rich knowledge, unlinked and unique. Knowledge is shared within the community of each island, allowing web users to gain feedback, critique and support. The SIOC project investigates how these islands can be interlinked together through the provision of bridges between knowledge sources.

One of the main products of the SIOC project has been the creation of the SIOC ontology [BHBD05]. This ontology is built on top of the FOAF ontology, by reusing the conceptualisation of people and relationships between people, to define how users interact and share information within communities. Whereas FOAF encapsulates the digital identity of a person including their biographical and social network information (from the *my identity* and *shared identity* tiers - §2.4), SIOC conceptualises a given person's community information (from the *abstract identity* tier). The SIOC ontology contains six main classes of knowledge associated with a community: Site, forum, post, role, usergroup and user. These abstract class definitions offer a flexible range for capturing knowledge which exists in online communities. Properties are expressed between the classes, for example describing the role of a user within a forum (e.g., moderator) and relating a user with a post.

Work by [BPCB08] demonstrates how Social Web platforms such as blogging services and social bookmarking platforms can express their content using the SIOC specification, providing semantic formalisations for authors, and their generated content. In doing so metadata models describing information within each island can be interlinked together through Linked Data principles, therefore linking islands of knowledge with information sources from the web of Linked Data (§3.2.6).

3.3.3 Nepomuk Contact Ontology (NCO)

The Nepomuk project¹¹ investigates the creation and use of social semantic desktops, investigating how knowledge can be modelled and reused across desktop environments. Projects

⁹<http://www.livejournal.com>

¹⁰<http://sioc-project.org/>

¹¹<http://nepomuk.semanticdesktop.org>

are modelled to support collaborations between workspaces by describing project contacts, roles of people within projects and work produced within projects. One of the key products from the creation of a social semantic desktop was the Personal Information Model Ontology (PIMO) [SvED07]. A given person can create his or her own personal information model using PIMO thereby formally representing their own personal knowledge space, this knowledge space can then be reused by other people in different settings.

In order to describe identity information the Nepomuk Contact Ontology (NCO) was created. NCO follows the same principles as FOAF by modelling a resource (denoting a person) as an instance of *nco:Contact*. Information is then attributed to the contact using properties to describe the person's name, webpage, telephone number and other biographical information. The Nepomuk Contact Ontology describes the digital identity of a person including their biographical information (from the *my identity* tier). The ontology does not define relationships between people or the communities within which people interact. PIMO and NCO are motivated by capturing personal information about users of social semantic desktops to leverage reuse across domains.

3.3.4 Microformats

Microformats¹², as the name suggests, are simple lightweight formats embedded within XHTML element attributes in a web page, much in the same way as RDFa (§3.2.2.4). Using a small and limited vocabulary, Microformats contain useful representations of knowledge commonly found in web pages (e.g., people, places, organisations). The web designer responsible for a given web page is able to add additional machine-readable markup to a web page's content, invisible to the human reader as it is rendered within the browser. In terms of digital identity information, in a similar manner to FOAF, Microformats provide formats for describing people and the relationships between people using the hCard and XFN Microformats respectively:

3.3.4.1 hCard

The hCard Microformat is a semantic representation of the vCard business card file format developed for attaching contact details to emails. The hCard Microformat describes people and organisations within web pages. Much in the same way as digital identity ontologies, an instance of a person is created and assigned biographical properties such as a name, a date of birth and an email address as follows:

```
<span class="vcard">
  <span class="fn">Matthew Rowe</span>
  <span class="url">http://www.dcs.shef.ac.uk/~mrowe</span>
  <span class="email">mrowe@dc.s.shef.ac.uk</span>
</span>
```

The hCard Microformat offers a useful way to provide XHTML markup for defining people and organisations. A machine reading the page is able to recognise the markup and extract the relevant data from the elements in the XHTML page. In order to allow metadata models to be gleaned from web pages containing the hCard Microformat an ontology was

¹²<http://microformats.org/>

created [HSW07]. The ontology defines the terms used by the hCard Microformat and the relationships between those terms.

3.3.4.2 XFN

Another useful and widely used Microformat is the XHTML Friends Network (XFN), which allows hyperlinks to be typed according to the relationship they denote. Hyperlinks used within web pages express a link between the page and another resource on the Web, the majority of hyperlinks lack context and meaning, and simply denote a relation of some kind. Even humans are, in some cases, unable to interpret the meaning of hyperlinks between resources. XFN offers a limited vocabulary of relationship types encapsulating family, geographical, professional and friendship. Terms from this vocabulary are encoded within the *rel* attribute of a given hyperlink as follows:

```
<a href="http://matthewroweshow.blogspot.com/" rel="me">Blog</a>
<a href="http://www.dcs.shef.ac.uk/~fabio/" rel="colleague">Fabio Ciravegna</a>
<a href="http://www.dcs.shef.ac.uk/~paul/" rel="friend">Paul Richmond</a>
<a href="http://www.dcs.shef.ac.uk/~jdb/" rel="colleague friend">Jonathan Butters</a>
<a href="http://www.paulgoss.net/" rel="contact">Paul Goss</a>
```

A software agent is able to parse hyperlinks containing the XFN Microformat and derive relationships based on the XFN types. An ontology has been created by [Cyg08] to conceptualise XFN relationships and explicitly type them as subproperties of *foaf:knows* from the FOAF ontology. Metadata models gleaned from web pages containing XFN Microformats are able to use this ontology in order to provide semantic definitions of relationships. This provides a specialisation of relationships, thereby overcoming the generality of using *foaf:knows*.

3.4 Summary

This chapter has presented an overview of the Semantic Web and various Semantic Web technologies which are used throughout this thesis. The Semantic Web is continually growing as more machine-readable information is published. In 2008 work by [HHRH08] provided an interesting insight into the current size of the Semantic Web, stating that “...*just as the surface of a sphere is bounded but unlimited, the Semantic Web is*”. The intonation of this statement reinforces the position of the Semantic Web as an interlinked web providing paths through a multitude of Linked Datasets.

Disambiguating web resources which cite a given person requires handling data within web resources using automated means. As discussed within this chapter, the use of semantics aids this automation in several ways: ontologies provide automated processes with a common understanding of information thereby aiding its integration, and using metadata models to represent the underlying knowledge structure of web resources exposes the information of the resource in a machine-processable manner, thereby enabling its comparison against seed data by disambiguation techniques. In later chapters of this thesis (chapters 7-9) several techniques are presented which provide novel disambiguation methods utilising the semantics of information in a distinct manner.

The range of different digital identity ontologies and vocabularies exposes the issue of homogeneous data being defined using distinct ontologies. To overcome this limitation tech-

niques are required to handle metadata models containing concepts from disparate ontologies. These ontologies must be mapped together to enable automated processes to discover the relations between related concepts in disparate ontologies.

This chapter concludes the background discussion of this thesis. Part II, which follows, presents the methodology investigating the disambiguation of *identity web references*. The following chapter presents an approach to identify *identity web references* throughout which Semantic Web technologies play a crucial role.

Part II

Methodology

“The Encyclopaedia Galactica defines a robot as a mechanical apparatus designed to do the work of man..”

From “The Hitch-Hikers Guide to the Galaxy” by Douglas Adams

Chapter 4

Requirements and Design: an Approach to Identify Identity Web References

4.1 Introduction

The discussion of digital identity within chapter 2 outlines the growing problems associated with the increased online presence of web users and the resultant reduction in online privacy. This increased presence has led to a rise in malevolent web practices; namely identity theft and lateral surveillance (see §2.5). In order to avoid falling victim to such practices an individual must be informed of the web resources containing his/her personal information - so that the correct action may be taken (i.e., removing the information or altering its visibility). The process of identifying web citations involves two stages: first possible web citations are gathered - by searching using the person's name - and second the web resources which cite the person are *disambiguated*. Disambiguation aims to reduce ambiguity by compiling a set of web resources all of which refer to the same person. Performing the task of identifying *identity web references* manually is time consuming and expensive, thereby presenting a clear motivation for the use of automated techniques.

This chapter proposes a user-centric approach to identify *identity web references* using social data leveraged from the Social Web to support the automated disambiguation techniques. Requirements are presented which disambiguation techniques must fulfill. These requirements are gleaned from an analysis of current disambiguation techniques within the state of the art. In essence the approach presented within this chapter provides the necessary architecture by which distinct disambiguation techniques can be investigated by using each disambiguation technique in a modular fashion - inputting seed data and web resources and outputting the set of web citations.

This chapter is structured as follows: section 4.2 presents an analysis of existing disambiguation techniques, discussing their limitations and methods. Section 4.3 describes the requirements that disambiguation techniques must fulfill and the requirements imposed on seed data. Section 4.4 presents an approach for identifying *identity web references* and an overview of the various stages of the approach - all of which are explained extensively in the

following chapters.

4.2 Related Work

As mentioned previously the process by which web citations are identified is divided into two stages: first web resources are gathered which *may* cite a given person, and second the web resources are analysed to discover which resources *do* cite the person. This thesis focuses on the latter of these stages: the *disambiguation* process. Disambiguation techniques can be divided into two types: *seeded* and *unseeded*. In a similar manner to machine learning strategies, *seeded* techniques (equivalent of supervised/semi-supervised machine learning approaches) rely on labelled data in order to support the disambiguation process - this can be some prior knowledge about a given person. *Unseeded* techniques (equivalent of unsupervised machine learning approaches) do not use any labeled data - instead consistencies are detected between web resources to facilitate grouping. An overview of state of the art *seeded* and *unseeded* techniques is now presented.

4.2.1 Seeded Disambiguation Techniques

The seeded technique used by [BM05] relies on background knowledge provided as a list of names representing the social network of a person. The Web is then searched using the person names within the social network and then link structures of the returned web pages are used as features to enable the clustering of web pages into two clusters; *relevant* which cite the given person and *non-relevant* which do not. The intuition behind using link structures is that web pages which cite a specific person are likely to be connected, maybe not directly but via transitive paths.

Disambiguation can be regarded as a classification problem (i.e., deciding on the label to give a web resource based on its content). In the context of seeded disambiguation techniques, web resources must be labelled as either citing a given person or not. Classifying web resources in such a manner has been investigated in work by [YHcC04] by gathering and labelling web pages. Seed data is provided as labelled training examples from which a machine learning classifier is learnt and applied to unlabelled web pages. Similar work is presented in [FL06] which, by using labelled seed data, is able to learn a classifier and apply this to a text-based dataset.

Similar seeded disambiguation approaches have been applied in the text classification and word-sense disambiguation domains. The approach described in [RJ99], known as “*meta-bootstrapping*”, attempts to compile a semantic lexicon by classifying words with their semantic classes: comparable to labelling a person’s name with the person it describes. Input is an unannotated text corpus and a collection of seed words, candidate words are then extracted and the strongest words are added to the lexicon. A heuristic is used to assess the suitability of the word to the semantic class, where the heuristic uses the existing lexicon to make the assessment, thus ensuring that previously added words are incorporated into future assessments. A similar approach to disambiguation is discussed in [TR02], which also takes as input an unlabelled corpus and collection of seed words. Words are then extracted using patterns and placed in a *pattern pool*, ranking is then performed and the strongest words are added to the lexicon. The rankings performed in both [RJ99] and [TR02] enable

disambiguation by ensuring that the correct label (semantic class) is assigned to the word.

Disambiguation through such bootstrapping is also performed in work by [Yar95] to disambiguate word senses. The approach is based on two key assumptions: one sense per collocation (surrounding words have the same sense), and one sense per discourse (a given word has a consistent sense in a single document). In relation to identity disambiguation in web documents, the former assumption is true, however the latter does not hold given that a web document could be a person listings page, or online telephone directory in which many identical person names appear that refer to distinct namesakes. The approach begins using several seed examples which denote differing senses (or classes), from which a classifier (or decision list as used in [Yar95]) is trained. The classifier is then applied to unlabelled examples from the corpus thus generating a new example set. A analysis of this approach to bootstrapping disambiguation decisions is presented in chapter 9.

An extension of the above bootstrapping paradigm is presented in [BM98], in the context of web page classification: a form of web page class disambiguation. Rather than using a single classification model, [BM98] apply a strategy known as “*Co-training*” to bootstrap two classification models, each with differing perspectives of the web pages. Disambiguation is performed by providing some initial seed urls, and building the classification models using different features from the seed documents (e.g., bag of words for one classifier, and hyperlinked words for another classifier). Each classifier is then applied to the unlabelled corpus of web documents and the strongest classifications are added to the training data of the other classifier.

The need to identify *identity web references*, given that web users are now susceptible to identity theft and lateral surveillance, is reflected in the number of various commercial services now available to tackle such practices. Such services, for a nominal fee, offer web users with the ability to track their identity online and refer users to web pages which contain their personal information. For example Garlik’s Data Patrol¹ service searches across various data sources for personal information about the individual and uses information when the user signs up to the service to aid the process (includes biographical information such as the name and address of the person). The Trackur² service monitors social media on the Web for references and citations with the intended goal of monitoring the reputation of an individual. Similar to Data Patrol, Trackur requires background information about the person whose reputation is to be monitored. Identity Guard³ is one of the most widely used personal information monitoring services. It functions in a similar vein to Data Patrol and Trackur by using identity information disclosed by the web user to monitor that person’s personal information online.

Background knowledge, or seed data, forms a vital component of seeded disambiguation techniques, however obtaining this seed data is a difficult process and is limited by access to resources, as highlighted in [BM05]. Manually building seed data is expensive and time consuming. In several pieces of work [YHC04], [WYZ07] and [FL06] background knowledge is previously known and supplied to techniques with no explanation of how it was obtained and from what sources. This lack of explanation presents a gap within the literature which must be addressed. Moreover relying on seed data, can limit the functionality of the ap-

¹<http://www.garlik.com/dpindividuals.php>

²<http://www.trackur.com/>

³<http://www.identityguard.com>

proach: if the features of the seed data do not cover features found within web resources (which are *identity web references*), then the approach may fail to detect those resources. Therefore seeded techniques must cope with this lack of coverage, and use a suitable strategy to overcome this limitation.

4.2.2 Unseeded Disambiguation Techniques

A large portion of literature related to the area of disambiguation focuses on partitioning a set of web resources into distinct groups, or clusters, where each cluster contains web resources referring to a given person. Such *unseeded* techniques require no supervision or preliminary background knowledge, as common features are observed between the web resources to enable partitioning. For instance, the unseeded disambiguation technique presented in [Mal05] searches for web pages using a person’s name, and then clusters web pages based on their common features, which in this case are person names within the pages. Clustering is performed using latent semantics analysis combined with hierarchical clustering.

An unsupervised technique from [FH04] gathers a set of web pages from searching the Web and creates a feature of each page by pairing a person’s name within the page together with a relevant concept from that page, for example *Matthew Rowe/PhD Student* or *Matthew Rowe/Student*. Using these pairings a maximum entropy model is learnt which predicts the likelihood that two pairings refer to the same person. Web pages are then clustered based on the likelihood measures of pairings within separate web pages. This approach requires not only for all person names to be found within the page, but also complicates the process by requiring the derivation of a concept that describes the person. Comparable work in [SHC⁺07] provides a similar unsupervised technique by aligning topics with person names within a web page. Generative models are used to generate the topic to be paired with the person name and web pages are then clustered by common name and topic pairs similar to [FH04]. Features within web pages are represented by named entities in the approach described in [KNTM08]. Using these named entities a term frequency-inverse document frequency (TF-IDF) score for each named entity within the document is calculated. Web pages are then clustered together by common named entities and TF-IDF scores for those entities above a certain threshold.

Several disambiguation techniques have been described in the literature to group publications and citations by specific authors: the technique described by [HZG05] uses spectral clustering to split a set of citations gathered from searching citation databases into disparate clusters where each cluster contains citations of a distinct author. The features used for each citation include the co-author names, the title of the paper and the place of publication. An alternative approach is described in [BG04a] by creating citation features from co-authors of the publication. Citations are clustered based on their link distances between one another, where the distance denotes the similarity of the citation co-author list.

Citation disambiguation, also termed *citation matching*, is addressed in [MNU00] over a large citation dataset. The goal of the approach is partition the corpus into clusters such that a single cluster contains citations to a single, distinct paper. The approach uses a two-step disambiguation strategy: first a blocking step is performed which generates *canopies* thereby pooling together *similar* citations using a computationally light metric (e.g., an inverted index of citation features), and second greedy agglomerative clustering is applied using an

expensive similarity metric (e.g., short document similarity via TF/IDF cosine-similarity) to group together citations which appear within the same canopy, thus producing k cluster where each cluster contains citations to a single paper. *Citation matching* is also investigated in [WMPH04] using graph partitioning within Markov random fields: a graphical model is created of citations and their inherent features. Using this model, cliques are then inferred based on common entity attributes (where the entity corresponds to a given paper).

An approach described in [WGLD05] extracts features from web pages as lexical information, linguistic information and personal information. A basic agglomerative (bottom-up) clustering strategy is then applied to a set of gathered web pages, resulting in several clusters where each cluster contains web pages citing a given person. By varying the combinations of features (lexical, lexical+linguistic, lexical+linguistic+personal) [WGLD05] demonstrates the variation in clustering accuracy when applied to the same set of web pages. The motivation behind this approach is to provide improved results for search engines when queried using a person's name by grouping web pages together that refer to a distinct person. Comparisons can be drawn between the unseeded techniques described so far and approaches documented in work published at the Web People Search Evaluation Workshop.⁴ The focus of this workshop is to address the highly ambiguous nature of person names within the context of person search results on the Web. The majority of the approaches featured at the workshop combine a feature extraction technique with a clustering strategy (e.g., agglomerative, conglomerative). Such approaches must detect how many different people, and therefore clusters, exist within the set of web pages.

One of major limitations of unseeded techniques is their unfocussed nature and their need to learn, or approximate, the size of k - where k represents the number of namesakes (different people sharing the same name) in a given set of web resources. If the task is to find all the web resources for a given person then such techniques are not as suitable as *seeded* techniques, due to the requirement for the correct cluster to be selected which contains the web resources citing the person. Furthermore when using unseeded techniques to disambiguate *identity web references* for a given person then a large portion of irrelevant information is processed and compared, detracting from the intended focus of the person. One of the advantages of unseeded techniques, however is their lack of a requirement for background knowledge describing a given person, this alleviates the need to build such seed data which can be an expensive process.

4.3 Requirements

Seeded and *unseeded* disambiguation techniques have both advantages and disadvantages: *seeded* techniques are suited to the disambiguation of web resources for a single person, yet rely on seed data to support the disambiguation process. *Unseeded* techniques are suited to the disambiguation of web resources for multiple people, yet are limited when disambiguating *identity web references* for a single person. The research focus of this thesis is the investigation of automated techniques which are able to alleviate the manual disambiguation process performed by web users who wish to find their web citations. An approach is therefore required which concentrates on the identification of *identity web references* for a single person, in doing so incorporating disambiguation techniques which are best suited to this

⁴<http://nlp.uned.es/weps/>

setting. To this end, state of the art work indicates that *seeded* disambiguation techniques are better suited to the task, as *unseeded* techniques lead to the production of k clusters thereby requiring the correct cluster to be selected for the individual whose *identity web references* are to be found. However, in order to utilise such techniques several requirements must be fulfilled both by a chosen disambiguation technique, and also by the provided seed data to the technique.

4.3.1 Requirements for Disambiguation

The requirements for a seeded disambiguation technique are as follows:

- *Perform the disambiguation process with minimal supervision*
The decision process must be able to disambiguate web resources effectively with no intervention following the provision of seed data. In doing so the technique must be fully automated and require no human involvement. This is consistent with several state of the art approaches such as Garlik, Trackur and [FL06].
- *Achieve disambiguation accuracy comparable to human processing of data*
Automated approaches are required to alleviate the need for humans to monitor their web citations. Therefore techniques are required which are efficient, low-costing and more importantly effective.
- *Cope with web resources which do not contain any features present within the seed data*
Although seed data provides background knowledge for disambiguating *identity web references* of a given person, it may not contain the maximum knowledge coverage. The disambiguation process must be able to cope with this lack of coverage by either learning from, or adapting to, features not present within the provided seed data in order to improve their performance.

4.3.2 Requirements for Seed Data

The reliance on background knowledge by seeded disambiguation techniques places emphasis on both the acquisition and quality of this data. Feature coverage of the data is important, it must describe the identity of a given person such that information can be used to correctly aid disambiguation decisions (i.e., contain features which are common with the features of web resources which cite the person). Therefore the provided seed data must fulfil the following requirements:

- *Produce seed data with minimal cost*
Manually constructing background knowledge is a time consuming and costly process. Therefore techniques are needed which overcome these limitations by producing seed data inexpensively.
- *Generate reliable seed data*
The seed data used by the techniques must contain accurate information about the person whose web citations are to be found. Inaccuracies in the data risk affecting the overall accuracy of the techniques.

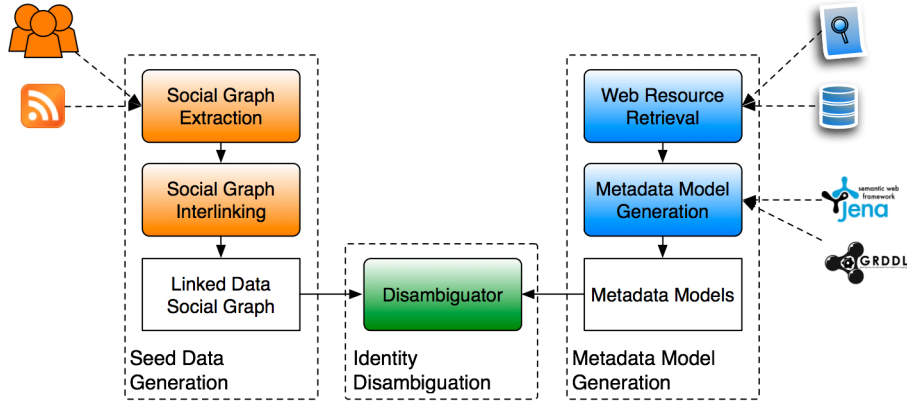


Figure 4.1: Architecture of an approach to identify *identity web references*

4.4 Approach Architecture

In order to investigate the use of *seeded* disambiguation techniques, seed data must be gathered and supplied to such methods. This section now describes a user-centric approach to identifying *identity web references*, which provides the necessary architecture to facilitate the provision of seed data to support automated disambiguation techniques. The architecture of the approach is divided into three stages:

1. Seed Data Generation
2. Metadata Model Generation
3. Identity Disambiguation

The first stage *Seed Data Generation* gathers seed data about the person whose web citations are to be found by exporting social graphs, modelled using RDF, from distinct Social Web platforms and interlinking these graphs together. The second stage *Metadata Model Generation* gathers web resources which are possible *identity web references* and generates metadata models describing the underlying knowledge structure of the resources using Semantic Web technologies. The third stage *Identity Disambiguation* applies the seed data to disambiguate *identity web references*.

Figure 4.1 shows the architecture of the approach where each stage consists of various steps. The following subsection explains the applicability of social data to aid disambiguation, before moving on to describe the various stages of the approach in more detail.

4.4.1 The Applicability of Social Data for Disambiguation

In chapter 2 of this thesis the correlation between data found on Social Web platforms and real identity information was discussed. Sociological studies and the empirical evidence provided as a result of the user study (in §2.7) indicate that data found on an example Social Web platform, to a large extent (77% coverage), mirrors real identity information. Deeper analysis of the real world social network which users maintain on Social Web platforms also indicated that the covered portion contained a range of relationship types (colleague, family, friends) not merely those relationships which are driven by social interaction. The provision

of such findings suggests that social data provides seed data to support disambiguation techniques, given its similarity with real world identity information. However, before discussing how social data is leveraged for such techniques, it is important to assess whether the features of this data would actually be beneficial to the disambiguation process. In essence, one must consider whether the features found within web resources are consistent with the available social data.

Prior to evaluating the disambiguation techniques presented within later chapters (7-9) a dataset of web resources was compiled by querying several search engines (Google, Yahoo and Sindice) using a list of researchers within the Semantic Web community (e.g., *Sam Chapman*, *Matthew Rowe*). By manually analysing a portion of the dataset it was found that on average 5 distinct people appeared within each web resource and that in the majority of cases these people knew one another (i.e., worked together, part of the same sports club). The presence of these features suggests that automated disambiguation techniques would benefit from seed data which describes individuals that a given person knows, in essence their social network. A large body of existing work within the field of person disambiguation [Mal05, KCMNT08, LBGT05, JWA⁺09] utilises this same assumption: that a person will co-occur in web pages with those people he/she knows.

As discussed in chapter 2, Social Web platforms offer rich functionalities which allow users to construct a digital identity consisting of biographical and social network information. Leveraging this information would provide features that are consistent with those present within web resources to be disambiguated. The Semantic Web technologies presented and discussed within chapter 3 allow this information to be leveraged in a machine-readable form as RDF, by using suitable ontologies to describe the biographical and social network information of a given person. Using the same technologies to represent information present within web resources which are to be analysed, would provide a consistent interpretation of information with the social data, thereby supporting the disambiguation process.

4.4.2 Seed Data Generation

Given the suitability of social data to support disambiguation techniques, the first stage of the approach, as shown in Figure 4.1, is to gather and prepare this seed data. Seed data is generated by reconciling digital identity fragments distributed across multiple Social Web platforms. This generation procedure consists of two steps: first individual social graphs are exported from several Social Web platforms using user submitted details (e.g., providing profile access, or submitting the URL/username of their account), where each graph describes the digital identity of the individual within a given platform (a single digital identity fragment). Social graphs are leveraged from individual platforms as RDF using the FOAF and GeoNames⁵ (describing geographical locations) ontologies to define biographical, social network and geographical information. Second these social graphs are combined together by interlinking associated information within separate graphs, thereby reconciling digital identity fragments.

Individual social graphs are represented using RDF, therefore a low-level inference technique known as *Graph Reasoning* is applied to interlink the distinct graphs together. This step produces a single Linked Data social graph containing a more complete digital identity

⁵<http://www.geonames.org/ontology/>

profile describing a given person for use as seed data by the disambiguation techniques. The social graph owner (the person describable by the graph) is therefore accessible via a dereferenceable URI, this URI is looked-up by the later disambiguation techniques and the relevant associated seed data returned.

4.4.3 Metadata Model Generation

The second stage of the approach generates metadata models describing knowledge present within web resources. This stage is also divided into two steps: first the World Wide Web and the Semantic Web are queried using the name of the person whose *identity web references* are to be detected. Second the returned web resources are prepared for analysis by the disambiguation techniques. Metadata models are produced, as RDF, describing the underlying knowledge structures of the web resources. Web resources contain information which is defined in a heterogeneous format: some resources contain well structured content which can be parsed easily (i.e., RDF, XHTML), while other resources contain badly formed content (i.e., HTML) which limits automated processing. This step handles differing formats in order to produce metadata models describing the web resource. The semantics of these models are normalised to be consistent with the ontologies used within the provided seed data, thereby providing a consistent interpretation of information. This consistent interpretation facilitates feature analysis by the disambiguation techniques.

4.4.4 Identity Disambiguation

The third stage of the approach uses the seed data from reconciling digital identity fragments on the Social Web and the web resources, represented as metadata models, to disambiguate *identity web references*. One of the key claims of this thesis is that disambiguation techniques achieve accuracy levels which are comparable to human processing. Therefore to contrast different approaches to the disambiguation problem, three complimentary *seeded* disambiguation techniques are explored. Each technique uses either a *supervised* approach by using only *labeled* data (seed data) or a *semi-supervised* approach by using both *labeled* (seed data) and *unlabeled* (set of web resources) data. The differences between these approaches are discussed following the evaluation of the techniques. The three explored techniques are briefly described as follows:

4.4.4.1 Inference Rules

Inference Rules are used to perform disambiguation by building SPARQL rules from the provided seed data - based on the semantics of the data - and applying the constructed rules to the metadata models of the web resources. Information within the models is matched by the rules in order to infer the web resource as being an *identity web reference*. This technique employs a *supervised* approach to disambiguation by using only the initial *labelled* data (seed data) from which to make disambiguation decisions.

4.4.4.2 Random Walks

A graph-based disambiguation technique known as *Random Walks* [Lov93] is utilised by constructing a graph-space from the seed data, which is modelled as RDF, and the set of web

resources to be disambiguated, the underlying knowledge structures of which are modelled using RDF. Triples within the RDF models represent an edge (the predicate) linking two nodes (the subject and object), therefore permitting edges to be weighted according to the ontology concept which they represent. Distances are measured through the graph space and clustering is performed using these proximity distances. Random Walks use both *labelled* and *unlabelled* data to make disambiguation decisions and is therefore deemed as providing a *semi-supervised* approach.

4.4.4.3 Self-training

A form of semi-supervised machine learning called *Self-training* [CSZ06] is employed to learn a classifier from an initial set of seed data and then gradually retrain the classifier, thus improving its accuracy. Machine learning instances are modelled using RDF graphs and the features of those instances are represented using RDF instances from the graphs. This therefore permits the novel variation of feature similarity measures between three different RDF instance matching techniques from the literature. Combining these different feature similarity measures with distinct machine learning classifiers provides unique permutations to be tested when Self-training. Unlike Inference Rules, Self-training incorporates both *labelled* and *unlabelled* data into the disambiguation process, by doing so a *semi-supervised* approach is employed.

4.5 Summary

This chapter has described a user-centric approach to identify *identity web references*. The three stages of the approach provide a modular architecture to enable distinct disambiguation techniques to be explored. Social data is collected from multiple Social Web platforms and used as seed data for automated disambiguation techniques. Possible web citations are gathered by querying the World Wide Web and the Semantic Web, returning a set of web resources from the queries and converting the resources into metadata models using knowledge representations formats and ontologies. The approach uses as input the seed data (modelled as a social graph) and the set of web resources into a given disambiguation technique.

Several requirements are outlined within this chapter which a disambiguation technique must fulfill. These requirements were created based on the use of *seeded* and *unseeded* techniques reviewed within the state of the art. The disambiguation techniques which are presented in chapters 7, 8 and 9 of this thesis each use a *seeded* approach, therefore relying on the provision of prior knowledge about a given person to facilitate disambiguation. The review of current work when applying *seeded* disambiguation techniques has exposed a gap within the literature where a coherent explanation of how seed data is acquired is not provided. Requirements are placed on the seed data within this chapter to ensure that it is built inexpensively and is reliable - i.e., contains features which support disambiguation.

This chapter has provided a brief overview of the work which is presented within the following five chapters. The next chapter presents research exploring the generation of seed data by reconciling digital identity fragments distributed across the Social Web.

Chapter 5

Collecting Seed Data from the Social Web

5.1 Introduction

Chapter 2 discusses the issues of *identity fragmentation* and *data portability* associated with Social Web platforms. This chapter investigates methodologies to overcome these limitations, and in doing so leverage seed data to be used by the later presented disambiguation techniques in chapters 7, 8 and 9. An approach is presented which creates machine-readable metadata models describing social data found within distinct Social Web platforms. Semantic Web technologies from chapter 3; RDF and ontologies, are used to provide consistent semantic interpretations of information from these disparate sources. The created metadata models are then interlinked together to provide a single social graph to be used as seed data.

This chapter is structured as follows: section 5.2 presents state of art techniques to leverage social graphs from Social Web platforms and methods to links these graphs together. Section 5.3 describes the requirements for social graph generation and interlinking. Section 5.4 presents the technical architecture of social graph generation. Section 5.5 details the various techniques and applications created to generate social graphs from heterogeneous data sources. Section 5.6 presents the methodology for linking distributed social graphs together. Section 5.7 evaluates the accuracy of the interlinking methodology and presents the results and discussion. Section 5.8 presents the conclusions drawn from the work conducted within this chapter, paying attention to the fulfilment of the necessary requirements.

5.2 Related Work

5.2.1 Exporting Individual Social Graphs

Generating social graphs from Social Web platforms describes digital identity information in a machine-readable format. Several existing tools have been developed to facilitate the process of exporting machine-readable information from such platforms, these are now discussed.

5.2.1.1 Flickr Exporter

Work by [Pas08] describes the creation of a web application designed to export a user's personal and community information from the photo sharing, and social networking, site Flickr.¹ The Flickr Exporter² functions through the use of a simple user interface requesting the username of the user whose information is to be exported. The exporter generates RDF using both the FOAF and SIOC ontologies as follows: the user's personal information is described using FOAF, detailing their name, and location, along with their friends on Flickr. The *foaf:knows* relation is used to depict those relationships. User data is interlinked by describing the geographical location of the user using Geonames,³ giving a unique resolvable URI to the location. The SIOC specification is used to describe additional properties of the user detailing the location of the user's photo gallery and their profile image. The exporter also details any communities that the user has joined within Flickr, as this forms an integral part of the site allowing users to share and discuss specific photo topics. The SIOC specification is well suited to this task, as it facilitates the expression of forums and groups, and whether a given user is a member, and if so what their role is.

5.2.1.2 Twitter Exporter

Similar to the Flickr Exporter, an exporter of RDF⁴ has also been written for the micro-blogging service Twitter. This exporter uses both the FOAF and SIOC ontologies to describe the friends a user is 'following' on Twitter, i.e., their social network. Each friend is described as an instance of *foaf:Agent* containing FOAF semantics such as *foaf:name*, *foaf:nick* and *foaf:homepage*. A blog post made by a user is expressed as an instance of *rdf:Description* containing *sioc:content* to express the content of the post, with *foaf:maker* linking back to the URI of the poster. However, one of the limitations of the Twitter Exporter, unlike the Flickr Exporter, is the lack of functionality to export geographical information despite Twitter storing this data.

Each of the above exportation tools utilises information which users generate and share within Social Web platforms. By leveraging this information as machine-readable RDF, *data portability* can be achieved, thus allowing information which was once closed and restricted to be collected and reused.

5.2.2 Interlinking Social Graphs

The reconciliation of distributed identity fragments requires exporting individual social graphs as RDF models and then linking these models together. In essence this linking process involves identifying related RDF instances in disparate models which describe the same resource. One of the advantages of using RDF to represent social data is its graph-like structure: an instance within a given RDF model is represented as a subgraph containing nodes (resources and literals) and edges (properties and relations). Detecting equivalent instances in disparate graphs can therefore be regarded as a graph-matching problem of which several state of the art techniques exist. Three example techniques are now explained

¹<http://www.flickr.com>

²<http://apassant.net/home/2007/12/flickrdf/>

³<http://www.geonames.org/>

⁴<http://tools.opiumfield.com/twitter/mattroweshow/rdf>

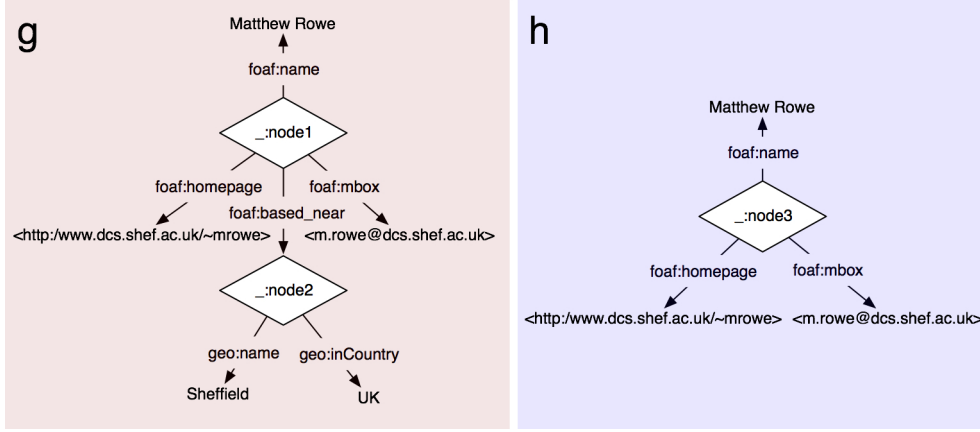


Figure 5.1: Equivalent RDF instances modelled as graphs g and h , where the description of each instance differs in size.

in detail. Within each technique the following set definitions are used: V_g denotes the set of nodes/vertexes within graph g and E_g denotes the set of edges within graph g . To ground the described techniques the RDF instances as shown in Figure 5.1 are used as an example.

5.2.2.1 Node/Edge Overlap

Graph matching using the overlap of nodes and edges as described in [PDGM08] utilises the Jaccard similarity [BDKW06] between two graphs to derive a similarity measure. The intuition behind this method is that the fewer edits required to transform one graph into another, then the greater the similarity between the graphs. This measure counts the number of edit operations to perform the transform, which are then normalised by summing the node and edge counts from each graph. Therefore the edit distance can be used to define the Jaccard similarity between two graphs as follows:

$$jaccsim(g, h) = 2 \frac{|V_g \cap V_h| + |E_g \cap E_h|}{|V_g| + |V_h| + |E_g| + |E_h|} \quad (5.1)$$

This measure returns a value ranging between 0 - indicating no match - and 1 - indicating an exact match. One of the disadvantages of this technique is that a threshold σ must be learnt, such that when this threshold is exceeded then two graphs are declared as equivalent. This reduces the generalisation of the technique between domains of application: as σ , once tuned to match RDF instances representing people in a given context, would not be applicable to match RDF instances representing publications in another context. In relation to the example from Figure 5.1, the two graphs are found to have 3 nodes which are the same and 3 edges, based on the presented formula for Jaccard similarity this yields a similarity measure of 0.705 (ignoring blank nodes).

5.2.2.2 Node Mapping

Work by [RSS08] presents a method to match graphs by providing possible mappings between nodes in separate graphs. This technique considers two sets of nodes: the set of object nodes (nodes with an incoming edge) and the set of subject nodes (nodes with an outgoing edge) denoted by O_g and S_g respectively. The similarity is derived for each possible mapping

between object nodes, and each possible mapping between subject nodes within separate graphs. This produces a list of possible object-to-object mappings, and subject-to-subject mappings together with a similarity score. Equivalence between graphs is found by summing the similarity scores for possible mapping combinations and normalising this by the number of pairs mapped. The similarity between two graphs is formally defined as:

$$sim(g, h) = \frac{\sum_{i=1}^{|O_g|} max(stringsim(s_g^i, s_h^i)) + \sum_{j=1}^{|S_g|} max(stringsim(o_g^j, o_h^j))}{|mappings|} \quad (5.2)$$

When processing this measure over Figure 5.1 one possible graph mapping is $\{("Matthew Rowe", "Matthew Rowe"), (<m.rowe@dcs.shef.ac.uk>, <m.rowe@dcs.shef.ac.uk>), (<http://www.dcs.shef.ac.uk/~mrowe>, <http://www.dcs.shef.ac.uk/~mrowe>)\}$, there are 3 pairs where each pair has a similarity score of 1 (given that the literals and resources are identical), normalising this cumulative count by the number of pairs (defined in Equation 5.2 as the number of mappings) would then give a score of 1. Mapping scores are calculated between many different instances, and the top ranked mapping is chosen as being equivalent. The application of the approach in [RSS08] is to detect links between music datasets for artists and records, however the context of application is not restricted to the music domain, it is clear that such a technique would be applicable to comparing instances in separate RDF models in order to detect equivalence. In a similar vein to Node/Edge Overlap, a threshold must be learnt which when exceeded declares the two graphs as being equivalent.

5.2.2.3 Graph Smushing

The term *smushing* describes the process by which RDF instances are aggregated together. Work by [SBF⁺08] explains how inverse functional properties of RDF instances can be used to facilitate *smushing* as follows: the decidability of such properties allows RDF instances which contain a value defined by an inverse functional property to have a one-to-one relationship with that value i.e., an email address of a person. No other entity may be identified by the same property value. In the context of Figure 5.1, the graphs would be matched as g and h each contain matching values defined by the inverse functional properties *foaf:homepage* and *foaf:mbox* (as shown in Figure 5.2). Although this technique has been widely used for matching RDF instances to construct interlinks between RDF datasets containing instances of *foaf:Person*, it is redundant when instances contain no value defined by inverse functional properties.

5.2.2.4 Reference Reconciliation

The process of matching instances is associated to the field of reference reconciliation. Existing work within the related field of object identification has explored the effects of training a *profiler* to detect object matches [DLLH03]. The profiler contains attributes and values which make up the description of the concept. This is then used to match other concepts which are deemed to refer to the same *thing* or object. The profiler runs using *If, Then* rules, such that objects are matched if they satisfy certain criteria. References are reconciled in [DHM05] by using contextual information and past reconciliation decisions in the form of a dependency graph. By using information which is external to a given entity reconciliation

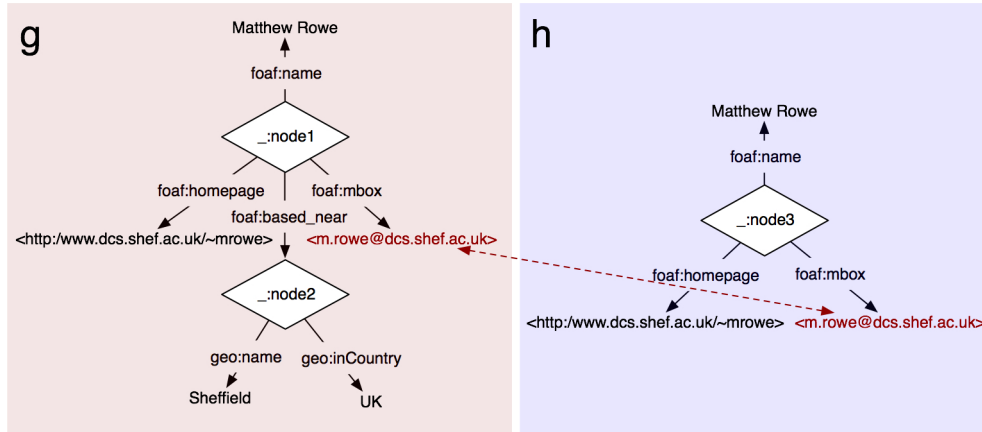


Figure 5.2: RDF instances matched based on matching values defined by the inverse functional property *foaf:mbox*

decisions can be performed, for instance when reconciling authors considering the individuals they have coauthored papers with. Techniques for the reconciliation of references are presented in [SPR09] which combine both numerical and logical (based on the semantics of distinct data items) approaches. In such work matching instances using only the *foaf:name* assigned to an instance of *foaf:Person* is unreliable due to person name ambiguity, therefore surrounding contextual information is utilised.

5.3 Requirements for Collecting Seed Data from the Social Web

The collection of seed data from the Social Web describing the digital identity of a given person is hindered by *data portability* and *identity fragmentation*. To address these issues techniques are required which are first able to export available digital identity information from a range of Social Web platforms, and then second combine this exported information together for use as seed data. In this instance the machine readability of information plays an important role to facilitate a common interpretation of information from disparate Social Web platforms. Digital identity fragments can be represented as metadata models using knowledge representation technologies described in §3.2.3 and digital identity ontologies explained in §3.3.2. This eases the discovery of equivalent information in separate models, as the semantics used to describe information in the models is the same. In order to collect seed data from the Social Web and overcome the limitations of *data portability* and *identity fragmentation* the following requirements must be addressed.

5.3.1 Requirements for the Exportation of Individual Social Graphs

- *Export social data contained within data silos*

Access to information contained within Social Web platforms is often restricted from being exported from the hosting service. To leverage information from such 'data silos' methods are required which can cope with such restrictions - this is essential where information is stored within a *walled garden* Social Web platform.

- *Capture the maximum amount of information*

Profiles on Social Web platforms contain both biographical and social network information. It is essential that information is not lost when exporting social graphs, thereby ensuring a sufficient amount of seed data is collected to aid disambiguation.

- *Provide information in a machine-readable form*

Exported information must be converted into a machine-readable representation. In the majority of cases, heterogeneous data schemas must be dealt with (i.e., XML models containing bespoke tags) and the correct ontologies applied.

- *Alleviate information ambiguity and anonymity*

When constructing a given metadata model from available information it is common for the instances to be created which are not identifiable by a URI, known as *blank nodes* or *anonymous nodes*. Such nodes must be identifiable, and therefore be given a URI in order to reduce ambiguity and allow dereferencing using Linked Data principles (see §3.2.6).

5.3.2 Requirements for Interlinking Distributed Social Graphs

- *Detect instances from separate social graphs referring to the same real world entity*

Instances in separate social graphs that refer to the same entity, or resource, must be linked together. This equivalence must be declaratively specified using the relevant machine-readable construct.

- *Maximise accuracy when interlinking social graphs*

The technique employed to derive links between distributed social graphs must achieve optimum results by correctly creating links between equivalent instances, whilst minimising the number of incorrect links created.

- *Generate a reusable aggregation of Social Web data*

When interlinking and aggregating social graphs into a single complete social graph it is essential to follow Linked Data principles and minimise the amount of information which is duplicated. The produced complete social graph must contain instances with links to additional information about those resources. The links should be represented as URIs pointing to external instances descriptions in distributed social graphs.

5.4 An Approach to Collect Seed Data from the Social Web

An overview of the approach architecture for reconciling digital identity fragments is shown in Figure 5.3. The aim is to create an interlinked social graph describing a given person's digital identity on the Social Web spanning multiple platforms, thereby addressing the following research question:

3. *Can seed data be gathered or constructed inexpensively?*

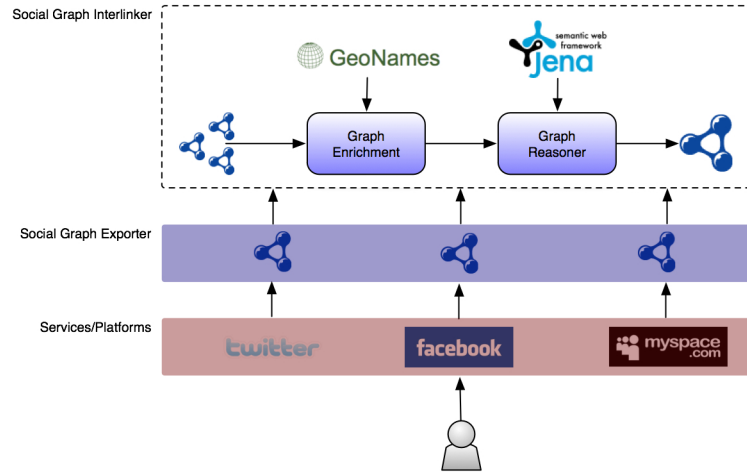


Figure 5.3: The proposed approach to collect seed data from Social Web platforms

Data portability is achieved from Social Web platforms by employing an exporter for each platform, this produces an RDF model containing concepts from existing digital identity ontologies. Each exporter functions by taking a given person's profile URI - linking to their account on the platform - and exporting the identity information from the site in its proprietary format. The exported information is then enriched with machine-readable metadata, thereby ensuring that it is no longer constrained to a single domain. The exporters can be easily adapted to capture new available information should the platform expose it. The modular separation of the approach offers greater flexibility which is important when considering the rate of growth of the Social Web and its susceptibility to change.

Individual social graphs are passed into the social graph interlinker, this component is responsible for linking the social graphs together. First the social graphs are enriched by providing resolvable URIs to identify geographical locations within the social graphs - these are locations of individuals in the social graph. Location details contain attributes such as the place name, however such names are often abbreviated, misspelt or one of several synonyms referring to the same place. Assigning a URI to this instance provides easier matching of locations which aids interlinking of graphs as will be demonstrated. The creation of an interlinked social graph requires the detection of equivalent RDF instances in separate social graphs which refer to the same entity, in this case a person (therefore instances of *foaf:Person*). The goal is to produce as rich a representation of information as possible: by detecting duplicate instances in separate social graphs and linking this information together a more complete description is created. When one considers that this complete social graph is later used as seed data for automated disambiguation techniques, then information completeness is vital in order to aid such techniques.

The following section (§5.5) details the techniques investigated and deployed in order to export individual social graphs from distributed Social Web platforms as a machine-readable format. The subsequent section (§5.6) presents a technique to link these distributed social graphs together and produce a Linked Data social graph.



This section discusses tools and methodologies to export social graphs from distinct Social Web platforms. Information present within each platform is discussed together with the methodology used to construct a knowledge model as RDF.

Although an exporter of social information exists for the microblogging site Twitter, this tool is limited in its exportation capabilities by not capturing a complete identity representation of the user. Based on this limitation, an additional exporter of social graphs from Twitter was created.⁵ The exporter functions by requesting information from Twitter for a given twitter user (using the username as input to the request), and an XML response is then returned. The conversion of this response into a semantic format involves mapping elements in the XML schema used by Twitter to concepts from the necessary ontologies. The response contains identity information describing the Twitter user: biographical information, social network information and latest microblogs. Biographical information and social network information is modelled using the FOAF ontology,⁶ thus enabling the depiction of relationships between the user and his/her friends in the generated social graph.

⁷The namespace *geo* is used hereafter.

5.5.2 Facebook Exporter

The Facebook Exporter⁸ allows digital identity information to be exported from the *walled garden* Social Web platform Facebook. First the tool authenticates the Facebook user whose social graph is to be exported. Following successful authentication, the service is queried to retrieve the identity information of the user which is returned as an XML response. In a similar manner to the Twitter exporter, biographical information is defined using the FOAF ontology; for each person, an instance of *foaf:Person* is created and the person's name is related to the instance using the *foaf:name* property.

A useful property which uniquely identifies a person is an email address. This can be defined using the *foaf:mbox* or *foaf:mbox_sha1sum* properties where each property is defined as *owl:inverseFunctionalProperty*. However Facebook and other similar Social Web platforms do not permit the exportation of email addresses (due to spamming). Instead to provide a URI for each person the user identification number employed within Facebook is utilised as a hash URI and is assigned to each person instance thereby providing a reference to the resource within the exported social graph. The user identification number is a simple incremental integer used by the site.

Social network information is captured by defining relationships between the user and his friends on the platform using the *foaf:knows* relation. Geographical information including the city and country of where a person lives is formalised as an instance of *geo:Feature*, defining the person's city and country using the *geo:name* and *geo:inCountry* properties respectively. The following is a snippet of the RDF exported from Facebook - displayed using RDF/XML syntax:

```
<foaf:Person rdf:ID="me">
  <foaf:name>Matthew Rowe</foaf:name>
  <foaf:givenname>Matthew</foaf:givenname>
  <foaf:family_name>Rowe</foaf:family_name>
  <foaf:knows>
    <foaf:Person rdf:about="#617555567">
      <foaf:name>Sam Chapman</foaf:name>
      <foaf:givenname>Sam</foaf:givenname>
      <foaf:family_name>Chapman</foaf:family_name>
      <foaf:img rdf:resource="http://profile.ak.facebook.com/617555567.jpg"/>
      <foaf:holdsAccount>
        <foaf:OnlineAccount>
          <foaf:accountServiceHomepage rdf:resource="http://www.facebook.com/" />
          <foaf:accountName>617555567</foaf:accountName>
        </foaf:OnlineAccount>
      </foaf:holdsAccount>
    <foaf:based_near>
      <geo:Feature>
        <geo:name>Sheffield</geo:name>
        <geo:inCountry>United Kingdom</geo:inCountry>
      </geo:Feature>
    </foaf:based_near>
  </foaf:Person>
</foaf:knows>
</foaf:Person>
```

In the above example the social graph owner - the person whose *identity web references* are to be disambiguated - is identified as the primary subject of the social graph using the

⁸<http://ext.dcs.shef.ac.uk/~u0057/FoafGenerator>

#me hash URI. A social network member is related to the person, and the biographical and geographical information is attributed to the person using the described concepts.

5.5.3 OpenSocial Exporter

Several Social Web platforms now use the OpenSocial API specification⁹ to define the APIs which allow third party access to data from such platforms. By developing an exportation technique for the OpenSocial API specification, exporters created for Social Web platforms supporting the same specification (Bebo,¹⁰ Orkut,¹¹ Hi5¹²) can be easily adapted to export social graphs from the other platforms.

Similar to the technique for exporting social graphs from Facebook, authentication is performed using OAuth¹³ prior to granting access to data. The response is returned as XML which is converted to RDF, describing biographical and social network information using the FOAF ontology and location information using the Geonames ontology. Each social network member is expressed as an instance of *foaf:Person* and the name of the person is described using the *foaf:name* property. Similar to social graph exportation from Twitter and Facebook, the user identification number of each social network member from the site is employed as a URI assigned to the relevant instance of *foaf:Person*. Geographical information is expressed using an instance of *geo:Feature* with *geo:name* describing the city or town, and *geo:inCountry* expressing the country. This location instance is then related to the person using the *foaf:based_near* property.

5.5.4 Graph Enrichment

Following the exportation of social graphs from Social Web platforms, it is essential to enrich the graph where possible. Enrichment is a process where instances within the graph which currently do not have a URI are assigned one. In terms of the Semantic Web vision (§3.2.1) resources on the web must have a URI, therefore reducing ambiguity and supporting common interpretation and integration of information attributed to resources.

The exportation of individual social graphs creates instances of geographical locations and assigns the location names and countries to these instances. This instance does not have a URI, it is an *anonymous node*. Therefore a URI is created for the location by querying the Geonames Web Service¹⁴ (which accesses the Geonames dataset) using the *geo:name* and *geo:inCountry* properties assigned to the *geo:Feature*. The service returns a list of possible URIs for the location ranked by relevance. The top URI is assumed to be the most relevant URI given the query. This is assigned to the instance, along with the latitude and longitude of the location which are assigned using the *geol:lat* and *geol:long* properties from the Geolocations vocabulary [Bri03]. The advantage of this enrichment procedure is evident when detecting equivalent instances of *foaf:Person* in separate social graphs as it simplifies the decision process to comparing URIs without the need for to compare the properties of the location. However if the location URIs are distinct then the properties of the locations are compared as will be discussed shortly.

⁹<http://www.google.com/opensocial>

¹⁰<http://www.bebo.com>

¹¹<http://www.orkut.com>

¹²<http://hi5.com>

¹³<http://www.oauth.net>

¹⁴<http://www.geonames.org/export/>

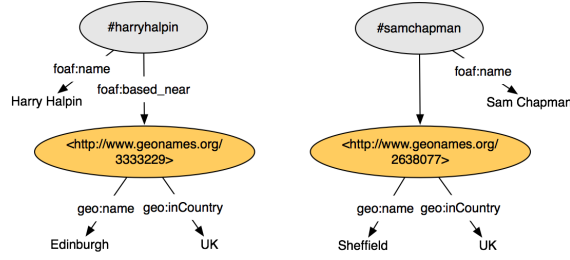


Figure 5.5: Individual *foaf:Person* instances represented as subgraphs

5.5.5 The Validity of Exported Social Graphs

The produced social graphs must be provided as valid RDF to ensure machine-readability. Prior to deploying each of the previously described social graph exporters as usable tools, validation was performed of the social graphs produced by each exporter: 100 randomly selected social graphs generated by each exporter were tested on two available RDF validators. Each graph was parsed using the RDF:About Validator,¹⁵ and the W3C RDF Validation Service.¹⁶ All the social graphs were validated as being correct, and no errors were found, therefore demonstrating that the produced RDF is valid, can be read by machines and more importantly, can be analysed for interlinking.

5.6 Interlinking Distributed Social Graphs

Interlinking social graphs identifies *foaf:Person* instances in separate graphs which refer to the same person and provides links between these instances using the *owl:sameAs* relation. Simply matching person names is unreliable due to person name ambiguity (several people in the same social network may share the same name), therefore additional information describing a person must be used to support the process of detecting person instances which refer to the same real world person. Work by [DHM05] uses contextual information about a given entity to resolve references. The same strategy is followed in this approach by using location information attributed to a given instance of *foaf:Person*. Location, or geographical information, provides consistent contextual information across multiple Social Web platforms. In an ideal scenario additional information would be used such as the company which the person works for, or their date of birth. However such information is not present, or is only present in a single platform, thereby making its applicability redundant when interlinking graphs. To support this decision process a subgraph is formed from each instance of *foaf:Person* in separate social graphs. The subgraph contains biographical and location information extracted from the social graph, an example of two subgraphs is shown in Figure 5.5. The formal definition for subgraphs is the same one used previously in this chapter when discussing the state of the art (§5.2.2).

Current graph matching techniques rely on comparing the syntactic properties of nodes and edges within separate graphs in order to return a measure of similarity. One of the advantages of using knowledge representation technologies, such as RDF, combined with ontologies is to support the decision process when comparing graphs, and therefore instances.

¹⁵<http://www.rdfabout.com/demo/validator/>

¹⁶<http://www.w3.org/RDF/Validator/>

Inference rules (see §3.2.5) provide a mechanism by which decisions can be made based on satisfying certain criteria stipulated within the antecedent of a rule, where the satisfied criteria provides the reasoning behind a derived conclusion or consequent. The following section investigates the use of such a technique named *Graph Reasoning* to detect equivalent person instances in separate social graphs by utilising the semantics of the information within instance descriptions. Graph reasoning is presented as an algorithm in Appendix A.

5.6.1 Graph Reasoning

Graph reasoning begins by verifying that the names of the two instances of *foaf:Person* are similar. This acts as a *blocking step* by only considering people with similar names as matches, thereby reducing the computations required if all instances of *foaf:Person* from separate graphs were compared. String literals related to the instances described by the *foaf:name* property are compared using a string similarity metric (e.g., Levenstein [CNC05]) to derive a match. If the names match then additional information is compared in order to confirm that the instances do refer to the same person and are therefore equivalent.

5.6.1.1 Unique Identifiers

Properties typed as *owl:inverseFunctionalProperty* such as *foaf:homepage*, *foaf:mbox*, and *foaf:phone* provide a one-to-one mapping between a person and a resource such that the resource is only ever associated with that person instance. As mentioned previously - in graph smushing §5.2.2.3 - matching values expressed using such properties provides a precise technique for matching equivalent *foaf:Person* instances. Such is the strength of inverse functional properties that should two instances of *foaf:Person* have matching properties, then the instances refer to the same person. A rule is now presented which detects such equivalence - using URIs of the social graph in two example exported social graphs to identify social network members - and representing the rule using SPARQL Rule notation:

```
PREFIX foaf:<http://xmlns.com/foaf/0.1/>
PREFIX owl:<http://www.w3.org/2002/07/owl/>
CONSTRUCT { ?x owl:sameAs ?y }
WHERE {
    ?person1 foaf:knows ?x .
    ?person2 foaf:knows ?y .
    ?x foaf:homepage ?h .
    ?y foaf:homepage ?h .
    FILTER (?person1 != ?person2)
}
```

The above rule infers an *owl:sameAs* relation between two resources, instances of people, if the *foaf:homepage* values assigned to those people match. The variables *?person1* and *?person2* query two distinct graph owners - controlled by the operator within the **FILTER** clause, thereby enabling two exported social graphs belonging to a given person to be linked together.¹⁷

¹⁷In the exported social graphs *?person1* and *?person2* are actually the same person, however the URIs will differ based on the exporters. They must be defined as unique to avoid matching instances in the same social graph.

5.6.1.2 Geographical Reasoning

Several Social Web platforms do not allow email addresses and homepage information to be exported. Therefore additional information must be used to find equivalent *foaf:Person* instances such as location information. Such information is used under the intuition that several social graphs describing the same person - i.e., the person whose web citations are to be found - will not contain different people in separate graphs who have the same name and the live in the same place. For example, a person named *Matthew Rowe* is friends with *Sam Chapman* on Facebook, and follows *Sam Chapman* on Twitter. On Facebook *Sam Chapman* is described as living in *Sheffield* and on Twitter *Sam Chapman* is also described as living in *Sheffield*. Therefore this suggests that both instances of *Sam Chapman* refer to the same person and should therefore be linked.

As Figure 5.5 shows, the subgraph representing *foaf:Person* is assigned location information defined as instances of *geo:Feature*. The graph enrichment process performed on exported social graphs assigns URIs obtained from the Geonames web service to anonymous nodes. Therefore when comparing instances of *foaf:Person* the URIs of each person's location are compared, should the URIs match then the instances of *foaf:Person* are deemed to refer to the same person. The following rule is only applied to people who have similar names, these people are filtered out using the previously described blocking step using a string similarity metric, thereby ensuring that person instances are only detected as equivalent if they have similar names and matching contextual information:

```
PREFIX foaf:<http://xmlns.com/foaf/0.1/>
PREFIX owl:<http://www.w3.org/2002/07/owl/>
CONSTRUCT { ?x owl:sameAs ?y }
WHERE {
    ?person1 foaf:knows ?x .
    ?person2 foaf:knows ?y .
    ?x foaf:based_near ?geo .
    ?y foaf:based_near ?geo .
    FILTER (?person1 != ?person2)
}
```

However, if the URIs are different then the geographical proximity of the locations is derived. Social web users will divulge more sensitive information in different platforms, depending on the visibility and accessibility of such information. For example in a *walled garden* Social Web platform the user feels that his/her information is more secure and would therefore state what suburb he/she resides in. Conversely, on a micro-blogging platform the user may only define their city of residence. Therefore the place names are used to derive a relation between the locations and discover the semantics of that relation should one exist. For example, a person named *Matthew Rowe* may reside in *Crookes* whereas another person, also named *Matthew Rowe* may reside in *Sheffield*. The locality of *Crookes* is derived to analyse if there exists a relation to *Sheffield*. In order to derive this relation knowledge bases are used that contain structured information expressing semantic relations between locations. In this instance DBPedia¹⁸ is queried through an available SPARQL endpoint¹⁹ using the following query:

¹⁸<http://dbpedia.org>

¹⁹<http://dbpedia.org/sparql>

```

SELECT ?p
WHERE {
    <http://dbpedia.org/resource/Crookes>
        ?p <http://dbpedia.org/resource/City_of_Sheffield>
}

```

This query returns the relation *dbprop:metropolitanBorough*²⁰ which describes *Crookes* as having a metropolitan borough which is *Sheffield*. Therefore, given that the names of the *foaf:Person* instances match, and the locations of each instance are geographically located, the instances are deemed as equivalent. The rule for matching locations by their suburban relation is as follows:

```

PREFIX foaf:<http://xmlns.com/foaf/0.1/>
PREFIX owl:<http://www.w3.org/2002/07/owl/>
PREFIX geo:<http://www.geonames.org/ontology/>
PREFIX dbprop:<http://dbpedia.org/property/>
CONSTRUCT { ?x owl:sameAs ?y }
WHERE {
    ?person1 foaf:knows ?x .
    ?person2 foaf:knows ?y .
    ?x foaf:based_near ?l .
    ?l geo:name ?g .
    ?y foaf:based_near ?m .
    ?m geo:name ?h .
    ?g dbprop:metropolitanBorough ?h .
    FILTER (?person1 != ?person2)
}

```

Several other location constraints are used including *dbprop:district* - defined as being a symmetric property - associating a given location with another location which is a district of the former, and *dbprop:postTown* which associates a given location with the postal town which is included in the former's postal address. It is common for alternative place names or abbreviations to be used for locations, for example *New York City* is commonly abbreviated to *NYC*. Such abbreviations should be covered when performing graph enrichment to resolve the locations to single URIs, however there are cases where this is missed or no URI is returned. When querying DBPedia, the relation *dbprop:redirect* defines two resources as being equivalent, therefore this can be translated into a rule as follows:

```

PREFIX foaf:<http://xmlns.com/foaf/0.1/>
PREFIX owl:<http://www.w3.org/2002/07/owl/>
PREFIX geo:<http://www.geonames.org/ontology/>
PREFIX dbprop:<http://dbpedia.org/property/>
CONSTRUCT { ?x owl:sameAs ?y }
WHERE {
    ?person1 foaf:knows ?x .
    ?person2 foaf:knows ?y .
    ?x foaf:based_near ?l .
    ?l geo:name ?g .
    ?y foaf:based_near ?m .
    ?m geo:name ?h .
    ?g dbprop:redirect ?h .
    FILTER (?person1 != ?person2)
}

```

²⁰Where the namespace *dbprop* refers to <http://dbpedia.org/property/>

For the above rules the place name literals are resolved to DBPedia URIs using the lookup service.²¹ this enables SPARQL queries to then be performed. Rules provide a solid mechanism for deriving a conclusion given the presence of certain criteria, which in this instance are triples. For a rule to hold and be satisfied, each of the triples, which forms the antecedent of the rule, must be present within the descriptions of the compared instances - thereby matching the graph patterns specified within the **WHERE** clause of the SPARQL rules.

The above rules could be generalised further, for instance by querying for a relation between locations and then verifying the semantics of the relation further to identify either equivalence - in the case of *dbprop:redirect* - or location subsumption - in the cases of *dbprop:metropolitanBorough*, *dbprop:district* and *dbprop:postTown*. This would involve dereferencing the ontology concepts and inspecting their semantics against pre-specified rules, thereby ensuring that no two locations are matched if the relation stipulates that the locations are geographically proximal (e.g., defining a location as being *dbprop:east* of another) or a similar relation.

5.6.2 Producing Linked Data

Following Graph Reasoning a set of matched instances of *foaf:Person* is produced. An inter-linked social graph is generated expressing links between these instances in separate graphs using Linked Data principles. The social graph owner - the individual whose digital identity fragments have been exported - is created as the first instance of *foaf:Person*. Interlinking has not been performed for this instance as he/she is expressed as the graph owner in each of the exported social graphs. For the set of matched *foaf:Person* instances a new instance of *foaf:Person* containing links to additional instance information as follows:

```
<foaf:knows>
  <foaf:Person rdf:about="#samchapman">
    <foaf:name>Sam Chapman</foaf:name>
    <owl:sameAs rdf:resource="http://namespace.com/facebook.rdf#617555567"/>
    <owl:sameAs rdf:resource="http://namespace.com/twitter.rdf#samchapman"/>
  </foaf:Person>
</foaf:knows>
```

Only the *foaf:name* property and a URI are included in the instance description. Hash values are used for identifiers according to guidelines described in [SCV07] due to the relatively small size of the datasets being considered for linkage. Where possible the identifiers are reused from the individual social graphs as such an identifier, if user selected, often resembles the handle of the user used across the Web - and is therefore familiar to human readers. For instance in the above *foaf:Person* instance information the identifier which *Sam Chapman* has used on twitter is reused: *#samchapman*. For *foaf:Person* instances which are not matched and only appear in one social graph then the identifier from that graph is reused.

In each of the previously described inference rules the consequent is the same; an *owl:sameAs* relation between two matched instances of *foaf:Person*. As the above RDF snippet shows, a new instance of *foaf:Person* is created representing a given person, the

²¹<http://lookup.dbpedia.org/>

owl:sameAs relation is then used to refer an agent viewing the instance information to additional descriptions via URIs (i.e., <http://namespace.com/twitter.rdf#samchapman>). The social graph owner is then associated with each of the new *foaf:Person* instances using the *foaf:knows* relation. In doing so when the Linked Data social graph is parsed the agent, or automated process, is able to find the social network of the social graph owner which spans many Social Web platforms. Appendix B contains a snippet from an example Linked Data social graph explaining the structure of the RDF model in more detail.

By providing Linked Data representations for each instance of *foaf:Person* in the inter-linked social graph, the duplication of personal information is minimised. This minimisation is an essential component of the reconciliation of distributed identity fragments as it passes responsibility for data access to separate locations and therefore separate access policies. For example, if a person wishes for their Facebook social graph to remain separate and only accessible by people they know and trust then access responsibility is delegated to the hosting service. Recent work by [mAYLL⁺09] presents a case for the control of social data using trusted parties, and recent advancements in access to social data now allows authentication to be controlled by such services such as OAuth and recently FOAF+SSL²².

Another motivation behind this design decision was to allow for extensibility and addition of new triples in the distributed social graphs. Imagine that a user has created his/her social graph containing exported social data from Facebook and Twitter. They have also recently built up a rich social graph of contacts and information on the professional networking site LinkedIn. Linking this new social graph to the existing Linked Data social graph only requires the addition of new triples to the existing structure. In essence this would include either the addition of a new instance of *foaf:Person* or the creation of a new *owl:sameAs* relation linking existing content in the aggregated social graph to the *foaf:Person* instance in the exported social graph from LinkedIn.

5.7 Evaluation

The success of reconciling distributed identity fragments, and thereby producing an inter-linked social graph for use as seed data, depends on the accuracy of the described Graph Reasoning methodology. The technique must correctly match the maximum number of equivalent instances whilst minimising the number of incorrect matches. In order to evaluate the accuracy of the graph reasoning technique, a comparison is made against two alternative graph matching techniques from the literature which were discussed at the start of this chapter: Node/Edge Overlap [BDKW06] and Node Mapping [RSS08].

5.7.1 Evaluation Measures

To perform the evaluation and gauge the accuracy of each of the techniques suitable measures must be used. Each technique is evaluated on its ability to minimise the number of Type I and Type II errors [SS83]: Type I errors are false positives, these are graphs, or instances, that are found to be equivalent which are not. Type II errors are false negatives, these are graphs, or instances, that are not found to be equivalent when they in fact are. To complete the sets: true positives are defined as the graphs which are correctly matched and

²²<http://esw.w3.org/topic/foaf+ssl>

therefore have links created between them. True negatives are the graphs which are correctly identified as not matching and therefore should not have links created between them. Such errors (Type I and Type II) are used to produce proportional error rates including the false positive rate - the proportion of false positives which should have been negative - and the false negative rate - the proportion of false negatives which should have been positive. These error rates range between 0 - indicating no errors - and 1 - indicating a large amount of errors - and are defined as follows:

$$false_pos_rate = \frac{|false^+|}{|false^+| + |true^-|} \quad (5.3)$$

$$false_neg_rate = \frac{|false^-|}{|false^-| + |true^+|} \quad (5.4)$$

5.7.2 Dataset

The dataset used for the experiment consisted of social graphs exported from two different Social Web platforms: Facebook and Twitter. Social graphs describing 50 people from the Semantic Web and Web 2.0 communities were obtained using the exportation methodologies described in this chapter. Therefore the dataset contained 100 social graphs, 2 for each participant. For each participant, the 2 social graphs describing that person were manually analysed to derive how many *foaf:Person* instances should be matched - and therefore links created - and how many should not be matched - no links created. This created a gold standard consisting of true positives and true negatives allowing the above error rates to be derived.

Every participant had social graphs where the same people appeared within separate graphs and should therefore be linked together. This is due to the nature of the platforms from which the data was exported: Facebook provides a platform for social networking based around sharing content such as photos and videos, and Twitter provides a platforms for sharing thoughts and musings with the world (§2.3). Both platforms are similar in their provision of functionality which allows relationships to be established and maintained, and content to be shared, yet offer slightly different environments for interaction.

5.7.3 Experimental Setup

Thresholds were required for both the node/edge overlap and node mapping techniques, which when exceeded would match the compared graphs. The thresholds were set to 0.7 for Node/Edge overlap (see Appendix D) and 0.75 for Node Mapping (see Appendix E). These values were found from tuning using 10% of the dataset. The Levenshtein string similarity measure [CNC05] was used by each technique for matching string literals and the threshold for a match was set to 0.75 following tuning using 10% of the dataset (see Appendix C). The remaining 90% of the dataset was held out for testing, the results of which are discussed below.

5.7.4 Results

Table 5.1 presents the results obtained from the evaluation. The Graph Reasoning technique presented within this chapter outperforms both Node/Edge and Node Mapping in terms of both the false positive and false negative rates. When tuning the thresholds for both Node/Edge overlap and Node Mapping the thresholds were chosen where the combined error rates were minimised. This is to ensure that sufficient instances are matched whilst matching instances accurately (indicated by low false positive rates). Graph Reasoning is able to match instances based on the use of contextual information which the other methods are unable to exploit. For instance the location information of a given person may not be the same as the provided location information of an equivalent person in a separate social graph. Node Mapping and Node/Edge Overlap fail to match such instances and due to the different information provided. Graph Reasoning assesses existing knowledge bases for a link between the location information and, based on the semantics of the link, confirms a match.

Overall the results indicate that Graph Reasoning provides a high performing technique for interlinking social graphs. The low false positive and negative rates ensure that instances of people in separate social graphs are accurately matched, whilst minimising the number of equivalent instances which are missed. Utilising the semantics of information within separate social graphs allows existing knowledge bases (i.e., DBPedia) to be used in order to discover relations between information and therefore aid the interlinking process. At the time of writing this thesis the use of such resources was beginning to experience widespread coverage, particularly within the Linked Data community.

Table 5.1: Results from interlinking social graphs from Facebook and Twitter

	Node 'Edge Overlap	Node Mapping	Graph Reasoning	GS
True Positives	76	90	211	256
True Negatives	564788	564735	564797	564809
False Positives	21	74	12	0
False Negatives	180	166	12	0
False Pos Rate	3.18×10^{-4}	2.93×10^{-4}	7.96×10^{-5}	
False Neg Rate	0.082	0.289	0.048	

5.8 Summary

This chapter has investigated how digital identity fragments distributed across the Social Web can be reconciled together in order to provide seed data for automated disambiguation techniques. Methodologies have been presented which allow information to be exported from several platforms on the Social Web. Digital identity information found within such platforms is converted to RDF using ontologies to describe the semantics of the exported information: biographical and social network information is defined using concepts from the FOAF ontology and geographical information is conceptualized using the GeoNames ontology, thus fulfilling the following requirement:

- *Provide information in a machine-readable form*

The Facebook FOAF generator (§5.5.2.1) has been widely used by the Semantic Web community for leveraging RDF from an existing information source. At the time of writing this thesis the application had been used 1,375 users with a monthly active user number of 23.

Social graphs exported from Social Web platforms are enriched to reduce resource anonymity and ambiguity. This enrichment is performed by assigning URIs to geographical concepts obtained from querying the GeoNames web service and hash URIs to person instances, thereby addressing the requirement:

- *Alleviate resource ambiguity and anonymity*

In order to leverage seed data from Social Web platforms techniques were required which were able to overcome the limitation of *data portability*. This chapter has presented several methodologies and tools which allow web users to export their digital identity information from Social Web platforms using Semantic Web technologies, thus enabling its reuse in a common format:

- *Export social data contained within data silos*

Disambiguation techniques require seed data which describes the identity of a person to support disambiguation decisions. The presented exportation methodologies function by gathering both biographical and social network information from within disparate Social Web platforms. Each of the methods exports consistent information returned with multiple platforms from querying the hosting service:

- *Capture the maximum amount of information*

In order to interlink distributed social graphs the Graph Reasoning method presented within this chapter detects equivalent person instances in separate social graphs. By using Graph Reasoning the semantics of information within instance descriptions is utilised to infer links between instances of *foaf:Person*. This fulfills the following requirement:

- *Detect instances from separate social graphs referring to the same real world entity*

The effectiveness of Graph Reasoning technique with respect to current state of the art methods demonstrates its applicability to the interlinking of distributed social graphs. Seed data must be accurate and reliable, any errors when gathering the data will have a detrimental effect on disambiguation decisions made by automated techniques. Although the alternative methods presented within this chapter achieve low error rates, Graph Reasoning provides a more accurate technique thereby fulfilling the following requirement:

- *Maximise accuracy when interlinking social graphs*

Seed data is gathered by reconciling a given web user's digital identity fragments through exporting information from several Social Web platforms and linking this identity information together as a complete RDF graph.

- *Generate a reusable aggregation of Social Web data*

The above requirement is addressed by conforming to Linked Data principles: dereferenceable URIs are used to provide links from the aggregated social graph to additional information within the original distributed social graphs. Using a *follow your nose* approach will

take a given software agent/process from the interlinked social graph to specific instances in distributed social graphs containing additional semantic metadata. The data contained within the exported social graphs can then be controlled by a separate access policy. This fits in nicely with recent work to address privacy and trust within the Semantic Web community, where technologies such as FOAF+SSL (to control social graph access) and POWDER²³ (to describe social graph properties) are being adopted.

The later discussed disambiguation techniques use this social graph as seed data by dereferencing the URIs for each person instance in the graph. The descriptions are then collected such that the social graph is populated with information about each instance. Linked Data principles have been followed to facilitate this process as when the URI is *looked up* the response is in a machine-readable form that an automated process can interpret.

²³<http://www.w3.org/TR/2008/WD-powder-dr-20081114/>

Chapter 6

Generating Metadata Models

6.1 Introduction

A web resource encapsulates any accessible content on the World Wide Web defined by a URI. This may include data models, XHTML documents containing embedded metadata and HTML web pages. Data models provide machine processable content such as RDF documents published on the web. Such models are not primarily designed for human consumption and contain structures which are intended for processing by machines and automated techniques. XHTML documents contain a well-formed parseable structure similar to XML often using metadata descriptions such as lowercase semantics (i.e., RDFa (§3.2.3.4) or Microformats (§3.3.3.1)) to provide both human and machine processable content. HTML documents provide information designed for human processing, which lack machine-readable metadata. Such documents are often poorly structured with broken HTML markup (e.g., containing unclosed tags).

For disambiguation techniques to function effectively web resources, which are to analysed and disambiguated, must be provided in a format which is consistent and interpretable, allowing disambiguation techniques to use features from the web resources. This chapter explores possible techniques to generate metadata models describing the underlying knowledge structure of web resources using Semantic Web technologies. In short the following research question is investigated:

4. How can information distributed across the World Wide Web and found in heterogeneous sources be interpreted by automated techniques?

This chapter is structured as follows: section 6.2 discusses automated techniques for generating metadata models from web resources and the associated issues and limitations of such techniques. Section 6.3 presents the requirements which techniques must address in order to overcome the limitations from §6.2 and therefore generate metadata models from web resources. Section 6.4 presents automated techniques to construct metadata models from different types of web resource formats (XHTML pages and HTML pages). Section 6.5 discusses the transformation of metadata models to use consistent semantics. Section 6.6 concludes the chapter.

6.2 Related Work

6.2.1 Handling XHTML Documents

Various methodologies exist for generating metadata models representing the underlying knowledge structure of web resources. For existing metadata models, such as RDF, model generation is not required as information is presented in a machine-readable form, instead the issue of inconsistent semantics arises - where different ontologies are used to describe homogeneous information; this issue is addressed later in this chapter. Web resources which are well-structured XHTML pages often contain embedded lowercase semantics using technologies such as Microformats or RDFa. Existing technologies such as Gleaning Resource Descriptions from Dialects of Language (GRDDL) [W3C07] allow metadata models to be gleaned as RDF from such XHTML documents. GRDDL uses an eXtensible Stylesheet Language Transformation (XSLT) [Kay07] specified within the header of the web page, this *lifts* the XHTML to RDF by providing references to concepts from ontologies specified within the transformation. Given that a web resource encoded using XHTML is well structured, other technologies such as XSPARQL [AKKP08] can also be used to *lift* the data structure within the web resource to RDF through the provision of necessary ontology concepts.

One of the notable limitations of such technologies is their refinement of the generated RDF models to a specific set of ontologies. Given the nature of the approach presented in chapter 4, automated disambiguation techniques rely on comparing known seed data, describing the digital identity of a person, with features present within web resources in order to make a disambiguation decision. Such automated approaches rely on a known consistent vocabulary of semantics to identify features within those resources. However, if the features are described using a set of semantics which are not within the known vocabulary then this process becomes limited. For instance, a gleaned metadata model may contain personal information about a given person described using the VCard ontology [HSW07]. An agent or automated process is unfamiliar with this ontology and therefore cannot interpret the model as containing personal information. Without an understanding of the ontology used it is unable to know that the *vcard:n* property describes a person's name. If the property had been defined using *foaf:name* then the process would be able to interpret that the value defined by the property as a person's name. Therefore gleaned models using concepts from unknown ontologies must be normalised to use a consistent set of known semantics which automated techniques are therefore able to interpret.

6.2.2 Handling HTML Documents

Using automated processes to generate metadata models from web resources containing HTML is limited by the lack of metadata descriptions present within such documents. Instead models must be compiled from scratch by discovering information within a given web resource that may be of interest, such as named entities and the contextual information which is relevant to those entities, and then building a model from this information. HTML provides a difficult format to extract information from, making many existing Natural Language Processing techniques redundant due to the tags and non-formal text used within such pages. Within this thesis the extraction of *person information* - extracting information describing a single person - is required to provide features of web resources which are

consistent with the provided seed data collected from the Social Web. Several efforts within the field of information extraction, attribute extraction and object detection provide efforts to address this problem, these are now discussed.

The extraction of person information from within HTML documents has been addressed in [TCS⁺05] by segmenting HTML documents into components based on the Document Object Model (DOM) of the web pages. Person information is then extracted using induced wrappers from labelled personal pages. Conversely work by [WGLD05] uses manually created name patterns to match person names within a web page and then, using a context window surrounding the match, extract contextually relevant information surrounding the name. The DOM of HTML documents is utilised in work by [ZLY⁺08] to provide clues to regions within the documents from which person information should be extracted. Once regions of extraction have been identified then extraction patterns are used to extract relevant information based on its proximity in the document. An effort to extract personal information (name, email, homepage, telephone number) from within web pages has been presented in [CCDW04] using a system called “*Armadillo*”. A lexicon of seed person names is compiled from several repositories which are then used to guide the information extraction process. Heuristics are used to extract person information surrounding a name which appears within a given web page.

Web pages from the same web site often conform to the same template - particularly if those pages are automatically generated. An approach described in [AGM03] attempts to exploit such consistencies in order to deduce the template used to define the web pages, and therefore extract the values from the pages. Such work is applicable to person information extraction when information is presented using a consistent schema. It also highlights the existence of definitive consistencies in the manner and order of the presentation of person information (i.e., person name followed by the email address of the person) which could be exploited for the extraction of person information.

One possible solution to extracting person information is the use of Hidden Markov Models (HMM) [FM99]. HMMs function as a probabilistic state machine by taking as input a sequence of observations - these are normally tokens or strings - and then outputting the most likely sequence of states, where the states label the observations. Work by [ZLT07] has explored their applicability for extracting medical citations from a citation repository by inputting a sequence of tokens and then outputting the relevant labels for those tokens based on the HMM’s predicted states: Title, Author, Affiliation, Abstract and Reference. Prior to applying the HMMs, windows within HTML documents are derived known as *component zones*, or *context windows*, these zones within the HTML document are considered for analysis in order to extract information from. Similar work by [Het08] has applied HMMs to the task of extracting citation information. The results achieved by [Het08] (99.5% f-measure for extracting a person name) and [ZLT07] (100% f-measure for authors) indicates that HMMs, when used appropriately (i.e., provided with the correct window for extraction) offer an extremely accurate means of extracting information from HTML documents.

Work within the field of attribute extraction has placed emphasis on the need to extract person information from within web pages. For instance [LZL⁺09] uses extraction patterns (i.e., regular expressions) defined for different person attributes to match content within HTML documents. An approach by [WBMM09] to extract person attributes from HTML documents first identifies a list of candidate attributes within a given web page using hand

crafted regular expressions - these are related to different individuals. All HTML markup is then filtered out leaving the textual content of the documents. Attributes which appear closest to a given person name are then assigned to that name. A similar approach to [WBMM09] is described in [dPSF09]: first a list of candidate attributes within a given HTML page are identified using a variety of means; NER for person name and location, and regular expressions for homepages and emails. Context windows are then considered surrounding the candidate attributes, learnt extraction patterns based on the lexical properties of these windows are then applied in order to extract person information by associating the occurrence of a person name within the window with the person's attributes.

6.3 Requirements for Generating Metadata Models

Automated disambiguation techniques must be able to interpret information within heterogeneous web resources. Describing the underlying knowledge structures of such web resources using metadata models provides a solution to this problem by applying ontologies to provide a consistent interpretation of information. However the makeup of the World Wide Web allows web content to be disseminated using a variety of formats - RDF, XHTML, HTML to name but a few. Therefore techniques are required which can cope with this heterogeneity and accurately produce machine-readable information, thereby enabling its analysis by disambiguation techniques. Techniques for generating metadata models must fulfil the following requirements:

- *Capture relevant information from HTML documents*
Information contained within generated metadata models provides the features which disambiguation techniques - supported by seed data - analyse. Information should be extracted from HTML documents where the features of this information are consistent with the provided seed data - i.e., describing people. Therefore techniques are required to identify such person information and enable its extraction.
- *Ensure accuracy when extracting information from HTML documents*
Incorrectly extracting person information from HTML documents may result in the incorrect attributes being assigned to the wrong person - i.e., a person being assigned another person's email address from within the page. Such errors must be avoided as they will affect the performance of disambiguation techniques and result in poor accuracy levels.
- *Build metadata models efficiently*
Where possible existing metadata model generation techniques should be used. For instance, where a web resource is formatted using XHTML and contains lowercase semantics (e.g., RDFa), XSLT transformations should be used to glean a metadata model from the page rather than using information extraction techniques - which are computationally intensive. This also ensures that information is exposed based on the XSLT defined by the web developer who built the web document.
- *Adapt to heterogeneous content structures and layouts*
The formatting and layout of HTML documents in particular varies greatly across the web. Generating metadata models from such web pages requires techniques which are

able to cope with variations in structure and layout and extract person information accordingly.

- *Maintain consistent semantics within generated metadata models*

Automatically generated metadata models may contain concepts from ontologies which are unknown. Chapter 3 presented various in-use digital identity ontologies all of which contain similar conceptual elements within slightly differing domains of discourse. Such variation has lead to different metadata models containing equivalent information yet defined using differing semantics. Therefore metadata models must be transformed to use consistent semantics with the provided seed data in order to facilitate a common interpretation of information.

6.4 Generating Metadata Models from Web Resources

This section presents various techniques by which metadata models can be generated from web resources. As highlighted in the discussion of the state of the art, the format which web resources are provided in varies, therefore requiring different techniques. A technique for generating RDF models from XHTML documents is first presented, before explaining how RDF models can be generated from HTML documents using information extraction techniques.

6.4.1 Generating RDF Models from XHTML Documents

When a given web resource contains metadata defined using lowercase semantics, such as Microformats or RDFa, then the XSL Transformation specified within the header of the XHTML page is used by GRDDL to produce an RDF model. This technique allows the designer of the page to handcraft the transformation such that only the information which they wish to expose or share is placed within the metadata model. It is not essential however that a transformation is defined as several existing XSL Transformations are available to extract information from an XHTML page depending on the lowercase semantics used. For instance Fabian Gandon's RDFa XSLT¹ is widely used to extract the embedded knowledge structure of a page as RDF. Similarly there are XSLTs available for different Microformats: hCard,² hCalendar³ (describes events within a web page) and XFN⁴.

To explain how this technique is adopted within this thesis to generate metadata models from XHTML pages containing lowercase semantics, the Linked Data on the Web (LDoW) workshop web page⁵ is used as an example. In this instance the header of the XHTML page does not contain an XSLT for the document, instead the web page must be assessed to see what XSLT would be suitable based on its intrinsic semantics. The Jena GRDDL reader⁶ is used to take the URL of the workshop web page, and then glean an RDF model from the document. The reader detects the differing formats used within the document and chooses the relevant XSLT transformation to apply. In the case of the LDoW web page the following markup is found within the document:

¹<http://ns.inria.fr/grddl/rdfa/2008/09/03/RDfa2RDFXML.xsl>

²<http://www.w3.org/2006/vcard/hcard2rdf.xsl>

³<http://www.w3.org/2002/12/cal/glean-hcal.xsl>

⁴<http://www.w3.org/TR/grddl-primer/grokXFN.xsl>

⁵<http://events.linkeddata.org/ldow2009/>

⁶<http://jena.sourceforge.net/grddl/>

```

<li about="http://events.linkeddata.org/ldow2009/papers/ldow2009_paper5.pdf"
  typeof="swrc:InProceedings" style="padding-bottom: 0.6em;">
  <a href="http://events.linkeddata.org/ldow2009/papers/ldow2009_paper5.pdf"
    title="Interlinking Distributed Social Graphs"
    property="dc:title"
    style="font-weight: bold;">
    Interlinking Distributed Social Graphs
  </a>
  <span rel="dc:creator">
    <a href="http://data.semanticweb.org/person/matthew-rowe"
      title="Matthew Rowe"
      typeof="foaf:Person" property="foaf:name">
      Matthew Rowe
    </a>
  </span>
  <a href="http://events.linkeddata.org/ldow2009/slides/ldow2009-slides-rowe.pdf"
    title="Slides for Paper: Interlinking Distributed Social Graphs">
    Slides
  </a>
</li>

```

Applying GRDDL to the above XHTML code leverages an instance of *swrc:inProceedings* defined by the URI of the paper. This instance is assigned the title of the paper "*Interlinking Distributed Social Graphs*" using the *dc:title* property. The ** element within the DOM contains the person who authored the paper, this is gleaned as an instance of *foaf:Person* and is assigned the name *Matthew Rowe* and a URI within an existing knowledge base - allowing agents and automated processes reading the page to look up additional information about the resource. The paper is related to the author instance using the *dc:creator* relation. The gleaned model for the above code looks as follows:

```

<swrc:inProceedings
  rdf:about="http://events.linkeddata.org/ldow2009/papers/ldow2009_paper5.pdf">
  <dc:title>Interlinking Distributed Social Graphs</dc:title>
  <dc:creator>
    <foaf:Person rdf:about="http://data.semanticweb.org/person/matthew-rowe">
      <foaf:name>Matthew Rowe</foaf:name>
    </foaf:Person>
  </dc:creator>
</swrc:inProceedings>

```

The above example demonstrates how a metadata model is gleaned from an XHTML document containing RDFa. A similar technique is applied to Microformats by applying an XSLT to extract the relevant information and using metadata descriptions from the correct ontologies (see §3.3.4). The use of lowercase semantics has seen a widespread increase in recent years as web developers begin to understand the benefits of providing machine-readable descriptions of their web content. For instance the business networking site LinkedIn now uses the hCard Microformat within profile pages to allow business contact information to be correctly extracted. The search engine Google now uses a technology called Rich Snippets⁷ which allows web developers to incorporate lightweight semantics in their page markup, which Google's crawler and indexer can interpret and reuse. This information, which was previously hard to extract accurately, is then displayed within search results (i.e., product ratings, service costs).

⁷<http://googlewebmastercentral.blogspot.com/2009/05/introducing-rich-snippets.html>

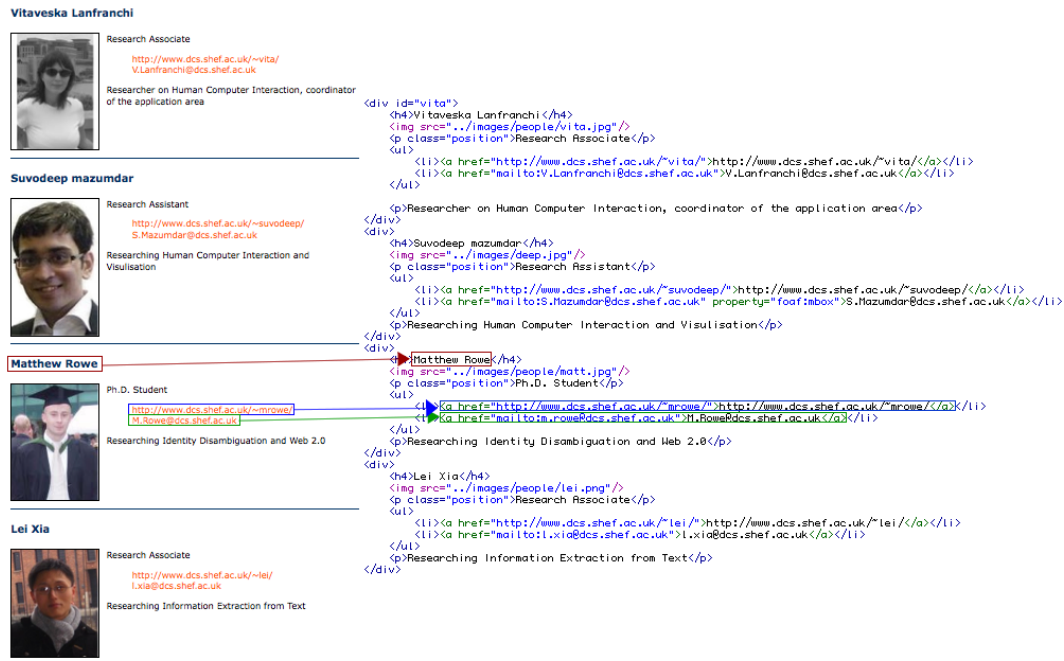


Figure 6.1: Person information residing within a HTML document

6.4.2 Generating RDF Models from HTML Documents

Statistics presented in [MMZ09] from crawling logs present empirical evidence of the increase in usage of lowercase semantics within XHTML documents from 2008 - 2009, such as a 0.4% increase in use of hCard microformat, now at 2% of web documents, and RDFa, now at 0.6%. Despite this rise in usage the majority of web resources are HTML documents. To provide a machine-readable representation of such pages for use by disambiguation techniques, person information must be extracted accurately and enriched with the necessary semantics. The state of the art reviewed at the start of this chapter presented current efforts to facilitate the extraction of person information with varying approaches and degrees of success. The most effective methods, such as [TCS⁺05, ZLT07], involved narrowing down the context of analysis to subsections of the HTML document, known as *context windows*, where such windows contained information describing a distinct person.

Person information, when found on the Web, is often structured so that it is legible and easy to understand for its intended audience. A major factor of this legibility is the presentation and formatting which person information adheres to. Within the context of web pages this formatting is controlled by the HTML markup which is utilised in the page. As such this formatting provides logical distinctions between disjoint person information segments which can in turn be exploited to extract person information from a given web page. For instance, if one considers Figure 6.1 as an example web page containing people listings of the Organisations, Information and Knowledge Research Group.⁸ The developer of the web page has segmented each person within the page into a given section, contained within a `<div>` element. Within each `<div>` element there are several pieces of information related to a single person including their name, email address and web address. Such attributes,

⁸<http://oak.dcs.shef.ac.uk>

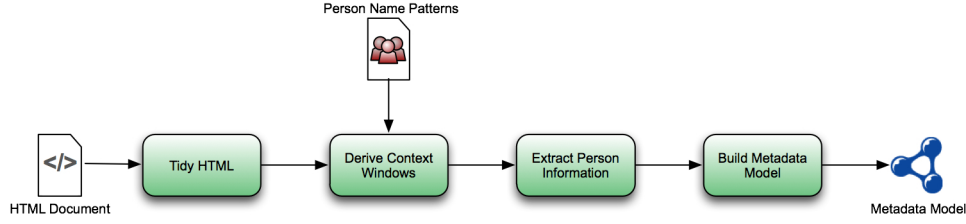


Figure 6.2: Pipeline architecture of the technique for generating a metadata model from a HTML document

once leveraged from the web page and used as contextually relevant information about a given person, provide person features which disambiguation techniques can utilise. In order to identify contextually relevant information, the web page must therefore be segmented into context windows so that information extraction techniques can then be applied to leverage the relevant information and build a metadata model of the web document containing this information.

Figure 6.2 shows the pipeline of the technique to generate a metadata model from a HTML document. First the HTML document is tidied into a well-structured form - this allows the Document Object Model (DOM) to be used by parsing techniques. Second context windows are derived using person name patterns to detect the appearance of person names within a given region or element of the document's DOM. Third an information extraction technique using Hidden Markov Models is applied to each of the context windows to extract information describing a given person. Finally the metadata model is built from the page using the extracted information. Work by [CLH09] assumes that web pages which contain multiple names refers to one person. The technique explained in this section uses a counter-intuition, in that every occurrence of a person name should be considered as a unique person or namesake - i.e., consider the role of person listings pages such as telephone directories or business contact sites.

6.4.2.1 Generating Context Windows

Figure 6.3 shows the DOM structure of the web page from Figure 6.1. In order to derive a set of context windows from a given HTML document, first the HTML document is tidied to provide a parseable form using Apache Maven's HTML Parser.⁹ HTML is often messy and contains poorly structured markup where HTML tags are opened and not closed. This reduces its ability to be parsed where such techniques require a well-formed DOM. Once tidied a person name gazetteer is applied to the HTML document. This gazetteer contains an extensive list of regular expressions designed to detect the appearance of a person name within a given body of text (See Appendix F for the list).

Algorithm 1 describes the process by which context windows are found within a HTML DOM. First the list of name patterns is loaded and applied to the DOM model, for each pattern the list of DOM elements which that pattern matches are collected (line 5). Each of the elements is then verified as having not been processed before (line 6) - this is because different name patterns may match the same person name at the same position in the document. The trigger string is extracted from the element (line 8) noting the person's

⁹<http://htmlparser.sourceforge.net/>

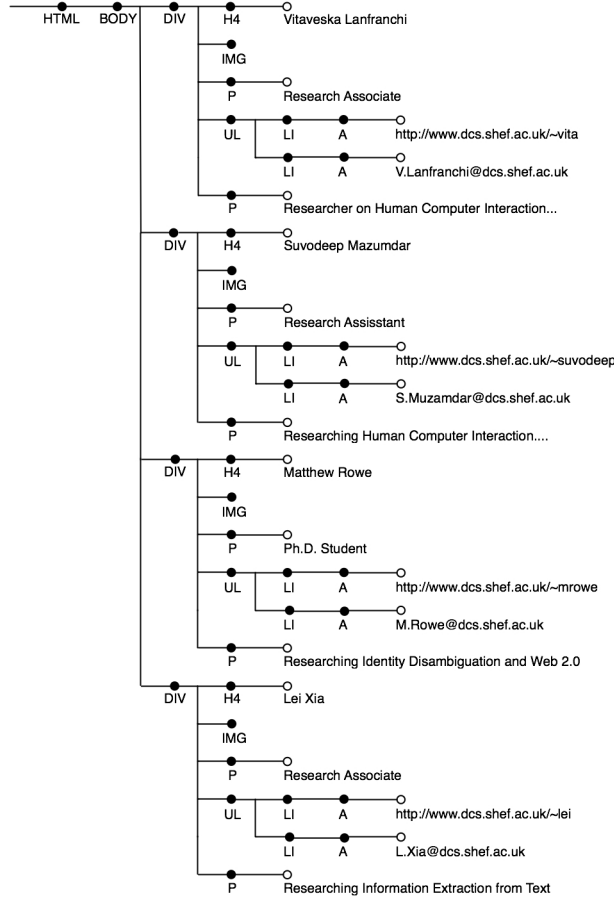


Figure 6.3: DOM Structure of the HTML document from Figure 6.1

name that was matched using the name pattern. The parent node type of the DOM element (e) is then assessed to see if it is a hyperlink (line 9): it is common for a person name to appear within a HTML document as a hyperlinked element. If it is hyperlinked then the grandparent of the element is considered as a possible area from which the context window can be gathered. However should the parent node of the element (e) not be hyperlinked (line 12) then the parent is then passed onto the *domManip* function for assessment together with the trigger string.

Algorithm 2 (*domManip*) takes the trigger node and a node from within the DOM and manipulates the DOM structure to derive a suitable DOM element from which the context window should be derived. First the node type is checked to see if it is a *<td>* element - checking if the trigger appeared within a table in the HTML document (line 1). If this is the case then the trigger string is passed to *extractWin* together within the parent of the *<td>* element: the *<tr>* element which *<td>* is a child of (line 2). If the node is a style element (**, *<h1>*, **, etc) (line 3) then *domManip* is recursively called using the trigger and the parent node. Such elements control the presentation and styling of a HTML document, but do not control or segment the layout like *<div>*, ** or *<td>* elements do. If neither of the above cases are true, and the node is a layout element via the process of elimination (line 5) then the content of the node and the trigger string is passed on to the window extractor. It is worth noting that the content of the nodes which is passed

Algorithm 1 *cwFind(dom)* : Given the DOM of a HTML document, returns a set of context windows

Input: *dom*

Output: Set of context windows *C*

```

1: N = person name patterns
2: C =  $\emptyset$ 
3: visited =  $\emptyset$ 
4: for each n  $\in$  N do
5:   E = getElements(dom, n)
6:   for each e  $\in$  E do
7:     if e.startIndex  $\notin$  visited then
8:       trigger = extract(e, n)
9:       if e.parent.type == < a > then
10:        c = domManip(e.parent.content, e.parent.parent)
11:        C = C  $\cup$  c
12:       else
13:        c = domManip(trigger, e.parent)
14:        C = C  $\cup$  c
15:       end if
16:       visited = visited  $\cup$  e.startIndex
17:     end if
18:   end for
19: end for
20: return C

```

Algorithm 2 *domManip(trigger, node)* : Given a trigger string and the node which contains the trigger, derives the suitable DOM element to extract the window from

Input: *trigger* and *node*

Output: *window*

```

1: if node.type == < td > then
2:   window = extractWin(trigger, node.parent.content.substring(trigger))
3: else if node.type == < style > then
4:   domManip(trigger, node.parent)
5: else
6:   window = extractWin(trigger, node.content.substring(trigger))
7: end if

```

onto *extractWin* contains HTML markup along with textual content. Unlike existing work within the attribute extraction state of the art - described at the start of this chapter - HTML markup is maintained as it provides clues which can aid the process of information extraction (i.e., HTML tags acting as delimiters between person attributes).

Algorithm 3 (*extractWin*) takes the trigger string (i.e., the person name) and HTML content string and derives the context window. First a mapping set is initialised and the list of person name patterns is loaded (lines 1-2). The preamble of the HTML content string contains the trigger string, therefore this is removed to enable the name patterns to match the remaining content. Each pattern is applied to the content string (line 4), if a match occurs (line 5) then the name pattern is added to the mapping along with the index within the content string where the pattern match starts. Once all patterns have been applied to the content string, the mappings, if there are any, are ordered by their start matching points within the content string. The first mapping is then chosen from the ordered set of mappings, given that this provides the nearest point to the start of the content string where a person name appears. The content string is then removed of the content following the earlier match (line 12), the trigger string is appended back on to the start and this is

Algorithm 3 $\text{extractWin}(\text{trigger}, \text{content})$: Given a trigger string and a DOM element's content, extracts the window from the trigger onwards

Input: *trigger* and *content*

Output: *window*

```

1: maps =  $\emptyset$ 
2: N = person name patterns
3: remove(content, trigger)
4: for each n  $\in$  N do
5:   if match(content, n) then
6:     maps = maps  $\cup$   $\langle n.\text{startMatchIndex}, n \rangle$ 
7:   end if
8: end for
9: if  $|\text{maps}| > 0$  then
10:  order(maps)
11:   $\langle i, n \rangle = \text{maps}_1$ 
12:  return trigger + content.substring(0, i)
13: else
14:  return trigger + content
15: end if

```

returned as the context window. Should no mappings be found (line 13) then the content string is returned as the context window.

The derived context window from *extractWin* feeds back to *cwFind* and populates the set of context windows for the given HTML document. These algorithms for context window derivation provide a conservative strategy to discover areas, or zones, of a HTML document from which person information can be extracted. It is conservative in the sense that it does not look above certain DOM element types (i.e., *<div>*) instead it relies on the logical segmentation of the document to provide the necessary features which can be utilised to identify context windows. Referring back to Figure 6.3 and considering the generation of the context window triggered by the person name *Matthew Rowe*, this technique would traverse two nodes up from the element containing the trigger string until a *<div>* element is found. The context window is then returned as the substring of the content within the *<div>* element from the trigger string - the person name *Matthew Rowe* onwards - until the end of the *<div>* element's content, returning the following:

```

Matthew Rowe</h4>

<p class="position">Ph.D. Student</p>
<ul>
  <li>
    <a href="http://www.dcs.shef.ac.uk/ mrowe/">http://www.dcs.shef.ac.uk/ mrowe/</a>
  </li>
  <li>
    <a href="mailto:m.rowe@dc.shef.ac.uk">M.Rowe@dc.shef.ac.uk</a>
  </li>
</ul>
<p>Researching Identity Disambiguation and Web 2.0</p>

```

6.4.2.2 Extracting Person Information using Hidden Markov Models

Given a set of context windows derived from a HTML document person information must then be extracted from these windows. Person information consists of four attributes: *name*,

email, *web page* and *location*, where the appearance and order in which these attributes occur in the context window can vary (e.g., $(name, email, www)$ or $(email, name, location)$). Therefore a technique is required which is able to cope with this variation and extract the correct information associated with the correct attribute.

Hidden Markov Models (HMMs) provide a suitable solution to this problem setting by taking as input a sequence of observations (e.g., tokens within a context window) and outputting the most likely sequence of states where each state corresponds to a piece of information (person attribute) which is to be extracted. HMMs use Markov chains to work out the likelihood of moving from one state to another ($q_i \rightarrow q_j$) and outputting symbol (σ when in state q_j), ignoring previous state-to-state transitions before reaching q_i . A given HMM is described as *hidden* in that it is given known observations while the states in which these observations occurred is hidden, the goal therefore is to identify these states. A HMM (λ) consists of the following parameters:

- **Set of States:** $Q = \{q_1, q_2, \dots, q_T\}$
- **Vocabulary of Symbols:** $\Sigma = \{\sigma_1, \sigma_2, \dots, \sigma_n\}$
- **Transition Probability Matrix:** A where a_{ij} gives $P(q_i \rightarrow q_j)$
- **Emission Probability Matrix:** B where $b_{i\sigma}$ gives $P(q_i \uparrow \sigma)$
- **Start Probability Vector:** π where π_i gives $P(q_{start} \rightarrow q_i)$

The above parameters can either be built, or estimated, from known information, essentially training the HMM in a *supervised* setting from labelled data, or estimated using unlabelled data, for example using the Baum-Welch algorithm [BPSW70], thereby building the parameters using an *unsupervised* approach. Hidden Markov Models are applied within this thesis using the former supervised approach, by providing manually labelled context windows from which the parameters of a HMM are then learnt. Given an input sequence of tokens (X) from a provided context window, the probability of that path being produced by a given HMM (λ) is found using the following function:

$$P(X|\lambda) = \sum_{q_1, q_2, \dots, q_l \in Q^l} \prod_{k=1}^{l+1} P(q_{k-1} \rightarrow q_k) P(q_k \uparrow x_k) \quad (6.1)$$

Equation 6.1 returns the probability of the given token sequence being produced given the hidden Markov model. However, there could be many possible state sequences through the window given the input sequence of tokens, as the same token may be emitted whilst in different states. The most likely state sequence through the window must therefore be found, this can be derived using the Viterbi algorithm [For73] as follows to identify the most likely state that each observation occurred in:

$$V(X|\lambda) = \arg \max_{q_1, q_2, \dots, q_l \in Q^l} \prod_{k=1}^{l+1} P(q_{k-1} \rightarrow q_k) P(q_k \uparrow x_k) \quad (6.2)$$

The parameters of the HMM are estimated using a supervised approach by providing as input a set of labelled context windows. Maximum likelihood estimates are made for the transition probability matrix by counting the number of times a given state s_i transits to state s_j and then normalised this count by the total number of transitions from state s_i :

$$a_{ij} = \frac{c(q_i \rightarrow q_j)}{\sum_{q \in S} c(q_i \rightarrow q)} \quad (6.3)$$

Similar to A , the emission probability matrix B is built from labelled training data. Counts are made of how many times a given symbol is observed in a given state, this count is then normalised by the total number of symbols emitted in that state. B is therefore defined as follows:

$$b_{i\sigma_n} = \frac{c(q_i \uparrow \sigma_n)}{\sum_{\sigma \in \Sigma} c(q_i \uparrow \sigma)} \quad (6.4)$$

The start probability vector is built from the training data by counting how many times a given state is started in. This is then normalised by the total number of start states observed:

$$\pi_i = \frac{c(q_{start} \rightarrow q_i)}{\sum_{q \in S} c(q_{start} \rightarrow q)} \quad (6.5)$$

HMM States

The HMM is provided with a discrete set of states which are divisible into two categories: *major* and *minor* states. The 4 *major* states constitute the 4 pieces of information, or person attributes, which are to be extracted from a given context window: *name*, *email*, *www* and *location*. The 13 *minor* states are defined to provide clues to the HMM and enhance the process of deriving the state sequence. Of those 13 *minor* states there are 2 *pre-major* states (*pre email* and *pre www*), 10 *separator* states (e.g., *between email and name*) and 1 *after* state which contains the symbols emitted at the end of the window. Emissions made within the *minor* states offer clues as to the order of the state sequence and what information is to follow. For instance using the emission of the token *<a* would indicate that a proceeding field might contain an email address or a web address. There is also a single *start* state in which the HMM begins. The start probability vector (π) contains the transition probabilities of moving from this state to another given state.

Smoothing Figure 6.4 shows an example HMM and the transitions which are learnt from provided training data. The sparsely connected nature of this model means that only transitions between certain states are possible, where the separator states act to provide clues as to the order of the attributes within the HMM. However when estimating the parameters of the HMM from labelled training data it is likely that certain state-to-state transitions, or emissions whilst in a given state are not observed. The trained HMM, when applied on test data, may then find previously unknown paths, or symbols emitted in states which have previously not been witnessed. The model must be able to deal with such possibilities by smoothing the transition and emission probabilities to cope with unseen observations and transitions within the training data. Several example smoothing methods are now described.

Naive Smoothing The Naive Smoothing method from [Het08] functions by setting zero probabilities in A and B to a very low constant of 10^{-7} . A and B are estimated using normalised values such that $\sum_{j=1}^{|A|} a_{ij} = 1$ and $\sum_{k=1}^{|B|} b_{i\sigma_k} = 1$, therefore when smoothing the zero probabilities in both A and B the non-zero probabilities must also be adjusted to

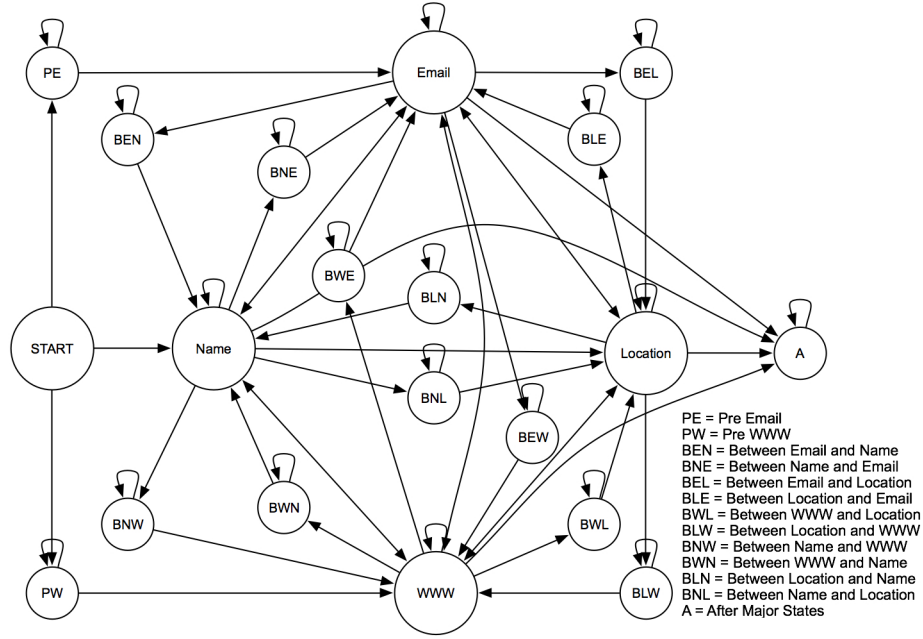


Figure 6.4: Topology of a Hidden Markov Model for person information extraction

ensure that the distributions hold. Therefore all non-zero probabilities have the value of $\frac{n}{m}10^{-7}$ subtracted from the current value where n is number of zero-probability events and m is the number of non-zero probability events.

Additive/Laplace Smoothing Another smoothing method using Additive, or Laplace, Smoothing described in [FM99] increments all zero counts when building A and B by 1. This ensures that the respective transitions and emissions are then assigned a low probability which is non-zero.

Absolute Discounting Smoothing is performed using Absolute Discounting in [FM99] by selecting a fixed discount d such that $0 < d < 1$. The parameters of the HMM are re-estimated for the emission probability matrix B as follows: for zero-probabilities, the number of zero emissions for all symbols whilst in a given state s_i is subtracted from the size of the symbol vocabulary V . This is then multiplied by the discount value d and normalised by the number of non-zero emissions in s_i multiplied by the number of zero emissions for all symbols whilst in the same state. Non-zero emission counts are calculated using the standard, and previously described, parameter estimation technique yet the discount is subtracted from the count for a given state emission. Absolute discounting is defined formally as:

$$P(\sigma_k | s_i) = \begin{cases} \frac{c(s_i \uparrow \sigma_k) - d}{c(s_i \uparrow \sigma)} & c(s_i \uparrow \sigma_k) > 0 \\ \frac{d \cdot (|V| - |Z_{s_i}|)}{c(s_i \uparrow \sigma) \cdot |Z_{s_i}|} & c(s_i \uparrow \sigma_k) = 0 \end{cases} \quad (6.6)$$

It is also worth noting that the use of smoothing is only applicable if the training data does not sufficiently cover possible transitions that are likely to appear and observations which are present in test data.

6.4.2.3 Vocabulary Dimension Control

One of the parameters of a HMM is its vocabulary of symbols - where the term symbol is used to refer to a given observation i.e., word, token, etc. This vocabulary contains all the possible observations or emissions which might be found within an input sequence. In [ZLT07] the vocabulary is compiled from a large corpus of words which make up all the possible symbols that could be observed. This works well where the dictionary is a finite size, dictated by the discrete number of possible emissions that are present within natural language text. For rare tokens that are not observed within the training data, existing work [SMR99] has labelled such tokens with $\langle UNK \rangle$ and estimated the probability density of that symbol for each state. Within the context of applying HMMs to web content, HTML markup is utilised to provide logical distinctions between elements.

Learning a discrete vocabulary of HTML tokens requires a large portion of labelled training, whilst not guaranteeing that the observed tokens within the test data will previously have been seen. This is due to web developers building their own markup code when creating web pages, designing bespoke Cascading Style Sheets using unique class and id labels for HTML elements. Applying existing method of using a separate symbol of all previously unobserved tokens, would place a significant portion of tokens within this class when applying the HMM to test data. Instead the approach is applied to convert tokens into a fixed vocabulary of symbols using a set of transformation functions, thereby controlling the dimensions of the a given input.

Dimension control is performed using mapping functions as follows: a given input - i.e., the context window - is tokenized, each token is then transformed into its respective symbols using mapping functions, this transformed input sequence of symbols is then used to derive the correct state sequence. The vocabulary contains 16 symbols, where each symbol has a mapping function which matches the token to a given symbol - e.g., $f(x) \rightarrow \sigma$. The symbol *First_Name* (FN) is used to identify a person's name. *Capitalised_Word* (CAP), *Lowercase_Word* (LW), *Uppercase_Word* (UW), *Initial* (INI) are used to identify word features of the tokens. *Location_Name* (LOC) represents a given place name that is found within the token sequence. *HTML_Tag_Open* (HTO) represents an opening HTML tag (e.g., $\langle p \rangle$). *HTML_Tag_Close* (HTC) represents a closing HTML tag (e.g., $\langle /p \rangle$). *HTML_Tag_Start* (HTS) represents the start of a HTML tag which still have proceeding content (e.g., $\langle a$). *HTML_Tag_Attribute* (HTA) represents an attribute of a HTML tag (e.g., $class="text1"$). *HTML_Tag_End* (HTE) defines a token which is the end of a HTML tag (e.g., $\rangle$). *URL* (URL) defines a token as being a web address. *Email* (EMA) represents a token which is an email address. *Year* (YR) is a token which contains four integer digits. *Punctuation* (PUN) represents a token which is a punctuation symbol (e.g., $:$). Finally *Alternative_Symbol* (ALT) represents a token which is not identified by the alternative symbols.

The mapping functions work by taking a given sequence of tokens, provided by tokenizing the content window and then transforming each token to its symbol equivalent. Depending on the symbol, different techniques are applied for its mapping function. For instance for the *First_Name* symbol, a person name gazetteer is used to detect whether a token is a person name or not. Similarly a location name gazetteer is used to detect the *Location_Name* symbol. Regular expressions are used to recognise tokens which are HTML tags, and whether

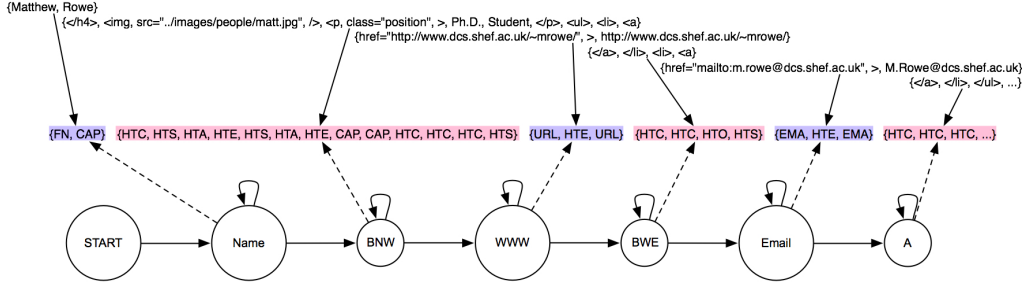


Figure 6.5: Hidden Markov Model providing labels for a given input sequence of tokens which are then transformed to symbols using mapping functions

they are opening tags, closing tags, start tags, etc. Web data is noisy and contains a large amount of variation in content form.

The advantage of this method is such that the heterogeneous nature of HTML markup can be handled, and generic presentation patterns can be observed and applied to extract person information. However, a disadvantage of this approach is that the HMM is provided with very few symbols to learn from, the closed vocabulary of which could lead to incorrect state sequences being produced by the Viterbi algorithm.

6.4.2.4 Deriving Transition Paths

As Equation 6.2 describes, the most likely state sequence through a given input token sequence is found using the Viterbi algorithm. Given a tokenized context window which has been converted into symbols, the Viterbi algorithm [For73] calculates the most probable state path through the window. This path is found using the transition probability matrix (A) and the emission probability matrix (B): given the sequence of observations the path is returned composed of the maximum likelihood estimates of moving from one state to another and then emitting a given symbol. The algorithm is a forward algorithm in that it only considers moving through a sequence of symbols without backtracking. Without smoothing the Viterbi algorithm may fail to return a path as a given symbol may be emitted within a given state which was previously unseen, in this case the probability would be zero, thus returning a path probability of zero and no path.

The Viterbi algorithm uses the learnt HMM, and its estimated parameters, as background knowledge of known transitions and emissions and assesses the input sequence to find clues as to the order of states. This allows consistencies in the layout and presentation of person information to be utilised to extract information for future tasks. For instance, it is common for a person to hyperlink their name with their web address, learning such patterns allows for future similar information extraction tasks to be recognised and the correct information extracted. Figure 6.5 demonstrates how person information is extracted from the earlier derived context window. The HMM identifies the first two tokens of the input sequence as denoting a person name and therefore labels those tokens as belonging in the *major* state *name*. Content is then labelled between the name and the URL of the person as a *minor* or *separator* state. The HMM continues this process by identifying the pattern of symbols within the input sequence and labelling the tokens with the appropriate major and minor state labels.

6.4.3 Evaluation

The construction of metadata models from HTML documents is an important step in the approach presented in chapter 4. The extraction of person information from such documents must be accurate in order to ensure that disambiguation techniques have correct and coherent information to analyse and therefore reach a correct disambiguation decision. Missing features which are present within the seed data may result in an *identity web reference* being missed.

6.4.3.1 Evaluation Measures

To analyse the accuracy of the presented technique to extract person information from HTML documents, the information retrieval metrics precision, recall and f-measure are used. Precision and recall were defined within chapter 2 (see §2.6.2), however the sets are slightly different in this instance: A denotes the set of relevant tokens, and B denotes the set of retrieved tokens such that:

$$precision = \frac{|A \cap B|}{|B|} \quad (6.7)$$

$$recall = \frac{|A \cap B|}{|A|} \quad (6.8)$$

Precision measures the proportion of tokens which were labelled correctly. Recall measures the proportion of correct tokens which were found. These measures therefore measure the accuracy of the labels and the ability of the technique to find person information within the HTML document - as lower levels of recall indicate that person information is missed. Evaluation is therefore performed on a per token basis within a given HTML document by assessing the accuracy of the HMM in labelling tokens with their respective major state labels and the ability of the technique to detect context windows.

In order to combine each of the above measures into a single metric F-measure is used, where the weighting between precision and recall is equal (i.e., using F_1 measure). Extracting the correct pieces of person information is important in order to reduce the impact of noise on the later described disambiguation techniques, given that the generated RDF Models provide the features utilised by such techniques. Additionally it is important to extract the maximum amount of available person information, thereby ensuring that the disambiguation techniques have features to base disambiguation upon. Therefore when combining both precision and recall into a single measure, the harmonic mean of both measures is taken with no bias given to either metric, this is defined as the F_1 measure as follows:

$$F_1 = \frac{2 \times precision \times recall}{precision + recall} \quad (6.9)$$

6.4.3.2 Dataset

The evaluation dataset was compiled by crawling the University of Sheffield's Department of Computer Science web site¹⁰ - in a similar vein to work by [CCDW04]. All internal pages within the web site were collected, totalling 12,000 HTML documents, of this collection

¹⁰<http://www.dcs.shef.ac.uk>

3,500 documents were found to contain person information. Each document had on average 5 context windows containing person information thereby generating 17,500 context windows from which person information was to be extracted. The training data was compiled from 200 randomly selected context windows within the dataset. Each window is already tokenised, however for training each token is labeled with the state in which it appears. The test data was also compiled by randomly selecting 40 URLs from the dataset and their respective context windows, therefore totalling 203 windows. A gold standard was then created for these windows by manually labelling the tokens with their respective states. Each URL was also manually analysed to find context windows which were missed by the technique, these were then added to the gold standard. In doing so this provides the necessary means to assess the coverage of person information extracted by the technique - a previously discussed requirement.

6.4.3.3 Experimental Setup

Various different smoothing techniques have been presented to avoid probabilities of zero in the transition probability matrix (A) and the emission probability matrix (B). Each of these techniques were tested in the evaluation. Absolute Discounting uses a discount d , where $0 < d < 1$, therefore requiring some value d to be set for the evaluation. Different values were tested and a suitable value for comparison following the experiments was found to be 0.1 - see Appendix G for experimental results. The results in the following section show the accuracy of the HMM when smoothing using Absolute Discounting where the discount was set to 0.1.

Table 6.1: Accuracy levels of Hidden Markov Models using different smoothing methods

Attribute	Naive Smoothing			Additive Smoothing			Absolute Discounting		
	P	R	F_1	P	R	F_1	P	R	F_1
Name	0.903	0.875	0.889	0.928	0.703	0.8	0.934	0.696	0.798
Email	1	0.867	0.928	0.578	0.688	0.628	1	0.892	0.943
WWW	0.849	0.833	0.841	0.714	0.714	0.714	0.849	0.833	0.841
Location	0.888	0.444	0.592	0.421	0.211	0.281	0.900	0.450	0.600
Average	0.910	0.754	0.825	0.66	0.579	0.616	0.923	0.717	0.807

6.4.3.4 Results

As the results from Table 6.1 show Naive Smoothing achieves, on average, higher f-measure levels with respect to the two alternative smoothing methods, whilst Absolute Discounting achieves the average optimum performance for precision. Additive Smoothing appears to yield poor scores, particularly for labelling web addresses and locations. Conversely Naive Smoothing and Absolute Discounting perform well for all person attributes, yet each perform poorly in terms of recall when extracting location information. The results indicate that smoothing zero probabilities with lower probability estimates yields more accurate state sequences. This is reflected in the experimental setup when deriving the optimal discount for Absolute Discounting and the low constant which Naive Smoothing utilises. In contrast to this, Additive Smoothing increments zero counts prior to estimating the HMM parameters,

and as a consequence derives probabilities which are much larger than Naive Smoothing and Absolute Discounting. This therefore hinders the Viterbi algorithm when deriving the most likely path, given that unobserved transitions or emissions have probabilities which are similar to observed events, thus yielding poor state selection.

The optimum smoothing method, based on the evaluation results, is Naive Smoothing. Although recall is low for location extraction, this method achieves high levels of precision throughout, ensuring that the extracted information is correct. The described technique of applying HMMs to extract person information provides a form of relation extraction where an entity (a person) is identified and relations are extracted between the person and their attributes. The advantage of applying a HMM in this setting is its ability to cope with variation in the appearance and order of the occurrence of these attributes, and extract the necessary relations when they occur.

In the context of disambiguation of web resources the accuracy of the extracted person information and the subsequent constructed metadata models is essential if automated disambiguation techniques are to discover features which are consistent with the provided seed data. Using the same methods for building seed data modelled using RDF in chapter 5, metadata models are constructed from the extracted person information within a given HTML document - creating an instance of *foaf:Person* and assigning the relevant information to the instance using the necessary relations. Figure 6.6 shows an example RDF model built from information extracted from a HTML document.

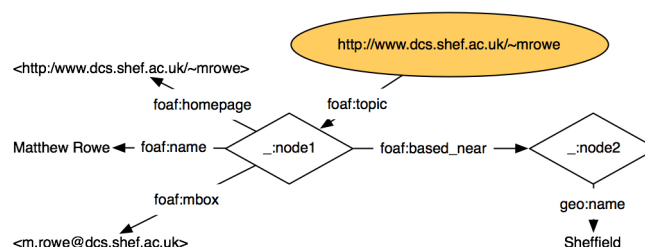


Figure 6.6: Snippet of an RDF Model generated from a HTML document

6.4.3.5 Deployment

The high accuracy levels achieved by the presented technique has facilitated the extraction of person information from HTML documents on a much larger scale than performed for the evaluation. The technique forms an integral part of generating Linked Data describing the University of Sheffield's Department of Computer Science. The dataset used for the evaluation formed the initial collection of data from which Linked Data was compiled. As mentioned the majority of the web site was comprised of HTML documents lacking machine-readable metadata. Therefore the technique described within this chapter was applied to the collected web documents, person information was extracted from the corpus and metadata models were constructed to represent each of the HTML pages - a similar approach is also implemented for extracting publication information. This formed the initial data which was then processed and clustered to produce a Linked Data service¹¹ describing people,

¹¹<http://data.dcs.shef.ac.uk>

research groups and publication information within the Department of Computer Science in an interlinked and machine-readable form.

6.5 Normalising Metadata Models to Consistent Semantics

A major issue when gleaning metadata models using current state of the art techniques (§6.2.1) and searching for web resources on the Semantic Web is the problem of heterogeneous semantics describing homogeneous information. This is particular common when digital identity information is defined within web resources due to the variety of digital identity ontologies (§3.3) available. Automated disambiguation techniques rely on using known seed data, which describes a given person, to analyse a set of web resources. Semantic descriptions of information are used to support this process by providing an interpretation of information within the resources, however if the process is not familiar with the vocabulary of terms used then such techniques are limited. Therefore ontologies and concepts within the RDF models must be consistent and coherent with the seed data. This requires that gleaned models and RDF models returned from querying the Semantic Web are normalised to use consistent semantics - in essence ensuring that only one concept is used to describe a given property, relation or class.

In order to normalise metadata models to consistent semantics, such metadata models must be transformed to use a required set of ontologies. Transformations can only be performed if relations are known between concepts within disparate ontologies in the form of mappings. Mappings specify the semantics of these relations by defining the type of mapping between the associated concepts using the necessary constructs. In this section state of the art mapping and transformation techniques are discussed. A mapping vocabulary based on state of the art approaches is then presented and the process of generating and applying transformation rules from this vocabulary is explained.

6.5.1 State of the Art

6.5.1.1 Ontology Alignment

Ontology alignment remains one of the most challenging and important fields of work within the Semantic Web community, emphasised by the number of entries to the 2008 Ontology Alignment Initiative [CEH⁺08]. As more ontologies are produced each day, discovering relations between concepts within distinct ontologies is essential to enable a common interpretation of information from disparate sources. Euzenet et al [EMS08] define ontology matching as the process of discovering relations between entities within distinct ontologies, where the products of such matches are alignments. Their work also provides a succinct definition of the ontology alignment process:

“...given two ontologies each describing a set of discrete entities (which can be classes, properties, rules, predicates, or even formulas), find the correspondences, e.g., equivalence or subsumption, holding between these entities. This set of correspondences is called an alignment.”

An alignment between two ontologies o and o' contains a set of correspondences, essentially a collection of mappings between the two ontologies, described as quadruples: $\langle e, e', r, n \rangle$ where e and e' are entities in distinct ontologies, r is a relation between the entities and n is the degree of confidence that the relation r is appropriate for the mapping. The scale of the Semantic Web and the creation and dissemination of ontologies requires automated techniques to perform ontology matching. An overview of such techniques is discussed in [EV04] which identifies several distinct varieties of approach: *terminological*: compared concept labels using string similarity measures; *structural*: comparing tree distance, performing graph matching (similar to the techniques discussed within chapter 5); *extensional*: comparison of instance level information; and *semantic comparison*: exploiting formalised knowledge within the semantic definition of ontologies.

Terminological matching is explored in [SSK05] proposes a new string matching metric for comparing entities within ontologies which combines string similarity with commonalities and differences between the compared entities. Similar work by [ES04] describes an approach to ontology matching which utilises several similarity functions to discover a mapping between two entities. The functions are composed of string similarity metrics, previous mapping equalities and a statistical approach known as multidimensional scaling which compares a given entity with all other similar entities to generate a vector representation, vector representations of entities in distinct ontologies are then compared.

This thesis does not attempt to advance the state of the art within the area of automated ontology matching, therefore the discussion of such techniques is not elaborated upon further. Should the reader wish to find out more about such approaches then please refer to [ES07] and for an overview of basic early schema matching work then please refer to [RB01].

6.5.1.2 Semantics of Mappings

The generation of alignments between ontologies requires the semantics of the correspondences, and therefore the mappings, to be expressed in a machine-readable and universally interpretable form. Expressing the relations within such mappings is supported by the Web Ontology Language (OWL) [SWM04] - first referred to within chapter 2 - which allows mappings to be defined between concepts in different ontologies using mapping constructs. Class and property equivalence mapping is defined using *owl:equivalentClass* and *owl:equivalentProperty* respectively, and hierarchical subsumption of a concept is defined by *rdfs:subClassOf*. One of the problems with specifying mappings using OWL based semantics is the strict level of expressivity. There are cases where a mapping may be appropriate to denote similarity, but not equivalence, the strict semantics of OWL do not support this definition. Work by [NGM08] addressed this issue by defining mappings as bridges between ontological concepts and not merely as denoting logical equivalence. Therefore mappings were defined as expressing similarity or associativity along with generalisation, specialisation and equivalence.

Similar work by the UMBEL project¹² has provided a lightweight upper ontology to relate similar ontological information from external ontologies. Concepts to be mapped are defined as instances of either *umbel:SubjectConcept* or *umbel:AbstractConcept*, depending on their level of abstraction. Generalisation between concepts is defined using *umbel:superClassOf*,

¹²<http://www.umbel.org>

and the inverse, specialisation, is defined using *rdfs:subClassOf*. Association, not equivalence, is defined using *umbel:isAligned* expressing some level of relatedness between two concepts.

As mentioned, the strict expressivity of OWL does not allow related/similar relations to be specified between concepts, instead requiring equivalence. When considering the number of ontologies now utilised within the Semantic Web, the duplication of similar conceptual elements is a growing issue. Specifying mappings between semantically related ontological concepts, along with equivalent and subsuming, provides a means to infer additional knowledge according to alternative data sources. The Simple Knowledge Organisation System (SKOS) [IS09] provides a set of mapping constructs in order to define relations between conceptual elements. A mapping vocabulary can be constructed from SKOS by specifying mapped concepts as instances of *skos:Concept*. Instances can then be mapped together using the relation *skos:related*, denoting a fuzzy level of semantic associativity which is less binding than equivalence. SKOS also provides the subsumption definitions of *skos:broader* and *skos:narrower* which, when compared with similar definitions used in OWL, offer flexible definitions of generalisation and specialisation without relying on strict constraints.

6.5.1.3 Transforming RDF Models

Transforming RDF models between ontologies requires inferring new statements about a model based on existing knowledge. Such inferences are possible through the use of rules (§3.2.5) as discussed previously within this thesis. RDF models are transformed using rules in work by [Pee03] by manually creating rules based on mapping semantics. Rules are applied to triples in the model that match the rule syntax, following a match a new triple is produced, which is then added to the model. The matched triple is dropped in the process. In work by [OY08] equivalent concepts in different ontologies are mapped using a *sameAs* relation. This enables model transformation by applying rules to the model thereby altering the definition of the concept within the metadata model. The same work also deals with the need to merge concepts expressed as a *joinTo* mapping.

Handcrafted rules for transforming RDF models between ontologies are presented in [OM05]. Similar to work by [Pee03] and [OY08] these rules allow triples within a given RDF model to be converted from one ontology concept to another. Exact mapping transformations are possible, together with transformations that create a new concept instance in the model by taking the union of two existing concepts using *owl:unionOf*. One of the consistent features of such state of the art approaches is their reliance on a mapping ontology to manage the relations between concepts in separate ontologies. The advantage of using a mapping ontology is that new mappings can be easily assigned to concepts within additional ontologies. This is particularly useful when considering the fast growth of the Semantic Web and its tendency to produce many lightweight ontologies with duplicated concepts.

6.5.2 Social Identity Schema Mapping (SISM)

To provide mappings between digital identity ontologies the Social Identity Schema Mapping (SISM) vocabulary was created for this thesis. The mappings within the vocabulary can be consulted in order to derive a relation between concepts from disparate ontologies, where the relation describes the semantics of the mapping in a machine-readable format. SISM is

available on the web¹³ as an RDF/XML document published using the *Best Practices for Publishing RDF Vocabularies* [BPM⁺08]. The document contains mapping triples where a source concept is the subject and the target concept is the object, the predicate contains the definition of the mapping. The characteristics and design choices of this mapping vocabulary are now discussed.

6.5.2.1 Mapping Identity Ontologies

SISM provides mappings between five ontologies used on the web to define digital identity information: The Friend of a Friend ontology [BM07], the ontology for VCards [HSW07], the XFN ontology [Cyg08], and the Nepomuk ontologies; Personal Information Model Ontology [SvED07], and the Nepomuk Contact Ontology. Each ontology contains both associated concepts that appear in other ontologies. These five ontologies provide the most commonly used metadata descriptions for person information on the Semantic Web, in particular FOAF which is the community standard for defining social network information. The advantage of using a mapping vocabulary such as SISM is its ability to be updated with new and emerging ontologies, for instance once available Facebook's Open Graph ontology¹⁴ will be added to SISM.

Five mapping constructs from the Web Ontology Language (OWL) and the Simple Knowledge Organisation System (SKOS) are used within SISM; *owl:equivalentClass*, *owl:equivalentProperty*, *skos:broader*, *skos:narrower* and *skos:related*. The combination of both OWL and SKOS constructs cover the range of mappings needed to map concepts from the available ontologies. As discussed earlier OWL semantics employ strong bindings of equivalence and subsumption which are suitable for imposing strict conditions on associated concepts. However, there may be cases where two concepts from distinct ontologies are similar or related in some way but are not equivalent, this cannot be expressed using OWL constructs. Therefore SKOS mapping constructs are used to define relatedness and to associate broader and narrower conceptual elements.

6.5.2.2 Collections of Mappings

Mapping concepts within SISM are each defined as an instance of *skos:Concept*. For instance, Figure 6.7 shows mappings between the *foaf:Person* and *vcard:VCard* classes. According to the SKOS primer a resource defined as an instance of *skos:Concept* can encompass any OWL class and property construct (i.e., *owl:Class*, *owl:ObjectProperty* and *owl:DatatypeProperty*). SISM contains a concept scheme (instance of *skos:ConceptScheme*) defined by the hash URI *#sism*. Each mapped concept within SISM is related to this concept scheme through the *skos:inScheme* relation. This would allow the Semantic Web to be queried asking for all the schemes that a given concept belongs to. If one imagines that thousands of mapping vocabularies are published, each containing a SKOS concept scheme, then the results returned by the query would describe the differing domains that a given concept has been applied in.

One of the design decisions when developing SISM was the use of collections to denote distinct sets of mappings. This was based on the intuition that SISM will grow in size as more identity ontologies become available, therefore there must be some way of grouping

¹³<http://purl.oclc.org/NET/sism/0.1>

¹⁴<http://opengraphprotocol.org/schema/>

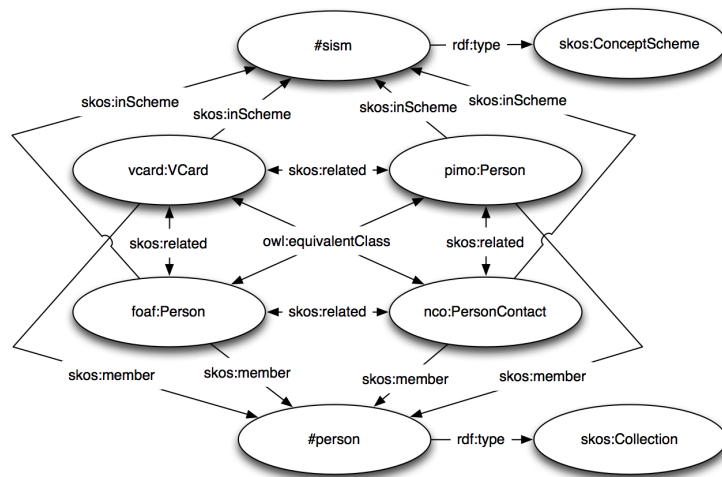


Figure 6.7: Collection of mapped Person concepts

mappings together into a logical set. For example, this would allow SISM to be queried for all the mappings which describe the name or the email address of a person. These collections are expressed as instances of *skos:Collection*. As Figure 6.7 shows, the collection of Person concepts is identified as a resource using the identifier *#person*. Concepts are placed within a collection using the *skos:member* relation to denote membership.

Within SISM five mappings constructs are used, two from OWL and three from SKOS. Figure 6.7 shows mappings between concepts from different identity ontologies where each concept defines a distinct person class. This collection defines the *foaf:Person* and *vcard:VCard* concepts as being related in some way but not equivalent as *vcard:VCard* defines an information resource containing contact information whereas *foaf:Person* defines an instance of a person. Therefore each concept is defined as being related using the *skos:related* relation. The relation between *foaf:Person* and *pimo:Person* is more straightforward as each ontology defines an instance of either concept as describing a person. Therefore the strict expressivity of OWL is used to relate the two concepts as being equivalent using the *owl:equivalentClass* relation.

Hierarchical relations between concepts in SISM are dealt with through the use of the SKOS relationship definitions *skos:broader* and *skos:narrower*. Figure 6.8 presents the collection of mappings between relationship concepts, where the concept *foaf:knows* is related to *xfn:friend* using *skos:narrower*. This defines *foaf:knows* as having a narrower conceptualisation defined by *xfn:friend*. The property *skos:broader* is the inverse of *skos:narrower*, therefore *xfn:friend* is related to *foaf:knows* using *skos:broader*.

SKOS constructs were chosen to express generalisation and specialisation due to their relaxed semantics, OWL constructs also allow the definition of hierarchical relations but impose strict restrictions on subsumption. For instance, the transitive nature of *rdfs:subPropertyOf* and *rdfs:subClassOf* could lead to incorrect conclusions being derived via automated reasoners. Applying SKOS semantics, given that neither *skos:broader* nor *skos:narrower* are transitive properties, ensures that incorrect inferences are avoided. Equivalent property concepts in distinct schemas are defined as equivalent using the *owl:equivalentProperty* relation.

model and the knowledge structure is to be transformed to use the FOAF ontology, SISM is consulted and the following mapping triple is found: $\langle foaf:Person, owl:equivalentClass, pimo:Person \rangle$. Based on the semantics of the mapping ($owl:equivalentClass$) and Definition 1 the following rule can be inferred which is directly applied to the RDF model to transform the class definition and remove the previous triple based on SPARQL UPDATE syntax from §3.2.5.

```
PREFIX foaf:<http://xmlns.com/foaf/0.1/>
PREFIX pimo:<http://www.semanticdesktop.org/ontologies/2007/11/01/pimo#>
INSERT { ?x rdf:type foaf:Person }
WHERE {
    ?x rdf:type pimo:Person
}
DELETE { ?x rdf:type pimo:Person }
WHERE {
    ?x rdf:type pimo:Person
}
```

The semantic relation between two concepts described using *skos:broader* defines a semantic mapping of generalisation. Based on standard hierarchical classifications if a concept is a specialisation of a general type then the concept can be converted into the more general conceptualisation. This is stated explicitly in Definition 1 that a concept with a given meta-data model can be transformed into its general form, should one exist. The XFN ontology, mapped within SISM, provides several concepts defining distinct relationship types. Each of these relationship types are specialisations of the general notion of a relationship expressed using *foaf:knows*. An example this relation is for *xfn:acquaintance* to be related to *foaf:knows* using the *skos:broader* predicate.

Definition 1. *Given the set of all triples T and a mapping vocabulary S . If there exists a mapping triple from S defined as $(\langle a \rangle owl:equivalentClass \langle b \rangle)$ or $(\langle a \rangle owl:equivalentProperty \langle b \rangle)$ or $(\langle a \rangle skos:broader \langle b \rangle)$ where $\langle a \rangle$ defines a conceptual element utilised in T , then $\langle a \rangle$ can be directly transformed (converted) to $\langle b \rangle$.*

Definitions are provided throughout this section to clarify the application of the generated rules. As both SKOS and OWL constructs are used to express relations between concepts from distinct ontologies, rules cannot be applied directly based on the semantics of these mappings in a manner that OWL constructs allow (i.e., applying a reasoning over mappings to infer generalisations). Instead the semantics of a given relation must be consulted and the relevant definition verified to ensure that the correct action is taken. An example of this is where a mapping exists between two concepts using the *skos:related* relation.

6.5.3.2 skos:related

The mapping predicate *skos:related* indicates an association between mapped concepts which is weaker than *owl:equivalentClass* and *owl:equivalentProperty*. A transformation rule can be generated from this mapping, however the model must be checked before it is applied. As the concepts are related and not equivalent applying the rule and therefore transforming the model may result in information loss (Definition 2). To clarify this issue a worked example will now be described. Consider the scenario where an RDF model is gleaned from an XHTML document and includes an instance of *vcard:VCard* describing a person's contact

details using the VCard ontology. This gleaned model must be transformed to use the FOAF and GeoNames ontologies, in the context of the approach from chapter 4, to provide consistent semantics with the collected seed data. To convert the *vcard:VCard* instance, the mapping vocabulary (SISM) is consulted and the concept is found to be mapped to *foaf:Person* using *skos:related*. Therefore transforming the class definition of the instance must not produce information loss within the instance description, this is performed using Algorithm 4. The intuition behind this algorithm is to check the collection of mappings that link concepts within a given metadata model with the target ontologies, this is to ensure that the transformations can be applied to the model without transforming any concepts from a general form to a more specialised form - as this is not possible.

Algorithm 4 `intCheck( $S, T, c_a, c_b$ )` : Check RDF model integrity prior to applying *skos:related* based rule. S is the mapping schema, T is the tripleset of the RDF model, c_a is the concept currently in T and c_b is the concept which c_a is mapped to in S

Input: S, T, c_a and c_b

Output: *true*|*false*

```

1: if (( $c_a$  rdf:type owl:Class) || ( $c_a$  rdf:type rdfs:Class)) then
2:   Get  $?x$  from ( $?x, \text{rdf:type}, c_a$ )  $\in T$ 
3:   Get  $?p$  from ( $?x, ?p, \_$ )  $\in T$ 
4:   count=0
5:   Get the ontology  $O_b$  which  $c_b$  is a concept in
6:   Get  $?m$  from ( $?p, ?m, O_b:\_$ )  $\in S$ 
7:   for each  $?m$  do
8:     if ( $?m == \text{skos:narrower}$ ) then
9:       count++
10:    end if
11:  end for
12:  if count>0 then
13:    return false
14:  else
15:    return true
16:  end if
17: else
18:   return true
19: end if

```

First the mapped concept is checked to see if it is a class definition (line 1), otherwise if it is a property or predicate (line 16) then the property definition is changed directly. If the concept is a class then the resource is retrieved which is an instantiation of the class within the model (line 2) and the predicates and relations attributed to the resource are gathered (line 3), this forms the properties of the instance within the model. A count is then established (line 4) and the ontology which the target concept belongs to is retrieved (line 5). All the mapping predicates are gathered (i.e., *skos:related*, *owl:equivalentClass*) from the mapping triples between properties in the instance description and concepts within the target ontology (line 6). Each mapping predicate is then verified that it is of type *skos:narrower* (line 8), if so the count is incremented. A concept definition within the model cannot be transformed from a generalisation to a specialisation (e.g., *foaf:knows* $\rightarrow$ *xfn:sibling* from Figure 6.8), therefore information within the instance description will be lost. The count then verifies if the rule should be applied (line 14) or not (line 12) based on the count.

Using the example in Figure 6.9 all predicates attributed to the resource *#me* from the VCard ontology are mapped to predicates within the FOAF ontology using *owl:equivalentProperty*

apart from the *vcard:adr* relation. In this instance the concept is mapped to *foaf:based_near* using the *skos:related* predicate. As the concepts are relation definitions a transformation rule can be applied directly. The instantiation of *vcard:Address* represents location information about the person within the web resource, it is mapped to *geo:Point* within the GeoNames ontology using *skos:related*. Therefore properties within the instance description are checked and as Figure 6.9 shows the predicates of the instance are mapped to concepts within the GeoNames ontology using *owl:equivalentProperty*, therefore a transformation rule can be applied to alter the class definition. An example rule generated from a *skos:related* mapping between *foaf:based_near* and *vcard:geo* is as follows:

```
PREFIX foaf:<http://xmlns.com/foaf/0.1/>
PREFIX vcard:<http://www.w3.org/2006/vcard/ns#>
INSERT { ?x vcard:geo ?geo }
WHERE {
    ?x foaf:based_near ?geo
}
DELETE { ?x foaf:based_near ?geo }
WHERE {
    ?x foaf:based_near ?geo
}
```

Definition 2. *Given the set of all triples T and a mapping vocabulary S . If there exists a mapping triple from S defined as $(\langle a \rangle \text{ skos:related } \langle b \rangle)$ where $\langle a \rangle$ defines a conceptual element utilised in T , then the integrity of the data model must be verified before $\langle a \rangle$ can be transformed to $\langle b \rangle$.*

6.5.3.3 skos:narrower

The mapping predicate *skos:narrower* defines a given concept (e.g., *foaf:knows*) as having a more specialised conceptual form (e.g., *xfn:friend*). Due to the semantics of this mapping, and as expressed within Definition 3, *foaf:knows* cannot be exchanged for *xfn:friend* within a given metadata model as the latter is a specialised form of the former concept. For instance, defining the relation between two people using *foaf:knows* expresses that the two people know one another, it does not make assumptions of the type of relationship between the two individuals. Changing this relationship definition for *xfn:friend* changes the semantics of the relationship, expressing the two people as being friends, something which could be incorrect (e.g., they could be work colleagues, not like one another, etc).

Definition 3. *Given the set of all triples T and a mapping schema S . If there exists a mapping triple from S defined as $(\langle b \rangle \text{ skos:narrower } \langle a \rangle)$ where $\langle b \rangle$ defines a conceptual element utilised in T , then $\langle b \rangle$ cannot be transformed to $\langle a \rangle$.*

6.5.4 Applying Rules: Model Transformation

Once rules have been generated from a mapping vocabulary they can then be applied to a given RDF model. The **WHERE** clause of the rule matches the triple in the model and the **INSERT** clause infers a new triple which is added to the model, dropping the triple within the **WHERE** once the triple is added. Based on the semantics of the RDF model unique identifiers are maintained within the model, the concept definitions are merely altered. Figure 6.9 presents an RDF model describing a person's information using the VCard ontology.

The model is to be converted to use the FOAF and GeoNames ontologies, thus providing consistent semantics with the seed data generated from the approach within chapter 5.

By consulting SISM rules are generated that map the concepts within the metadata model to concepts within the required ontologies. As Figure 6.9 demonstrates several mappings were defined using *skos:related*, therefore the integrity of the model was verified to make sure that no information was lost (as described in §6.5.3.2). Before applying any transformation rules, each of the rules generated from *skos:related* mappings are checked using Algorithm 4, if the algorithm states that the model's integrity will be sustained then the set of rules are applied. Figure 6.9 demonstrates the resulting transformation. It is worth noting that URIs are maintained for resources within the models. Given the domain of application for this technique - modelling identity information found within a web resource - instance descriptions which are altered are limited to person concepts or locations as these are consistent concepts with the provided seed data.

An important property of this method for normalising metadata models to use consistent semantics with the seed data is the provision of information, where transformations cannot be applied. As stated, if the model's transformation will result in information loss then the transformation is not applied. This is to preserve information for the later disambiguation techniques as certain techniques are able to cope with different vocabularies - this will become apparent in later chapters when the dynamics of the techniques are discussed.

6.6 Summary

This chapter has investigated techniques to construct metadata models describing the underlying knowledge structure of web resources. Creating metadata models provides features which the disambiguation techniques can then assess using a decision process supported by the provided seed data. For HTML pages the features are restricted to instances of people and their locations - where those instances contain features themselves (i.e., the email address of a person, a location name). For XHTML pages the features are provided by the model gleaned from the document - this is in turn governed by the transformation provided in the header of the page - which could contain an instance of an event, or a calendar. By using Semantic Web technologies such as RDF and ontologies, this chapter has demonstrated how machine-readable metadata models can be generated automatically from web resources provided as different formats.

The majority of web resources deployed on the WWW are represented using HTML. Therefore this chapter has presented an approach based on Hidden Markov Models and context window derivation via HTML DOM manipulation, to extract person information from HTML documents. The results of an evaluation of this approach demonstrate the high precision levels it achieves, particularly with respect to person attributes which can be used to uniquely identify a person - i.e., email address and web address. The high accuracy levels therefore fulfill the following requirement:

- *Ensure accuracy when extracting information from HTML documents*

The evaluation of the technique for extracting person information from HTML documents yields high levels of precision for all variations of smoothing methods whilst achieving slightly lower levels of recall. Such results ensure that the extracted information is accurate - in that

it minimises incorrect information describing a person. Despite precision being relatively high, a large amount of person information is still retrieved from within HTML documents - reflected by the recall levels - ensuring that features are collected for web resources which disambiguation techniques are able to analyse:

- *Capture relevant information from HTML documents*

Current trends of web design and web document publication indicates a large rise in the use of lowercase semantics within XHTML documents [MMZ09]. As discussed previously, this rise allows both human and machine-readable content to be disseminated across the Web. To ensure that information can be extracted efficiently and effectively from XHTML documents existing techniques were discussed and utilised such as GRDDL. The use of GRDDL allows any information which is semantically marked up within a web document to be extracted automatically and converted into an RDF model for reuse:

- *Build metadata models efficiently*

This chapter has explored how person information can be extracted from a variety of web resources covering different formats, structures and layouts. Existing techniques are deployed to automatically glean RDF models from XHTML documents, and trained Hidden Markov Models are then used to extract person information from context windows within HTML documents. The derivation of context windows uses a technique consisting of several algorithms capable of handling different content structures and exploiting the layout of the HTML markup to detect suitable windows for person information extraction. Such methods address the following requirement:

- *Adapt to heterogeneous content structures and layouts*

One of the main contributions within this chapter is an approach to normalise metadata models to use consistent semantics. As discussed previously, metadata models returned from knowledge bases and metadata models gleaned from XHTML pages containing lowercase semantics may contain relevant information described using distinct ontologies. Therefore the presented approach is able to convert the metadata descriptions within such models to use the required ontologies - this ensure that metadata models use the same ontologies as the provided seed data from the approach within chapter 5:

- *Maintain consistent semantics within generated metadata models*

It may appear naive at this point not to detect information within a collection of web resources which is consistent with the seed data. For instance by taking a given person's social graph and simply searching for people from the graph within the web resources. Although this may find many web references, it may be the case that the seed data does not contain a complete description of a given person whose *identity web references* are to be found. Instead additional features must be learnt so that they can be used to detect future web references. As such, the techniques presented in this chapter function in a modular fashion, they act as a data preparation step by taking a given web resource and converting information found within the web resource into a format which can then be used by the disambiguation techniques. The following three chapters of this thesis present three such techniques, each of which utilise the provided seed data and set of web resources - represented as metadata models - to disambiguate *identity web references* in a unique way.

Chapter 7

Rule-based Disambiguation

7.1 Introduction

Rules have been extensively used within expert systems and many Semantic Web based applications which use ontologies and their axioms to perform reasoning. As discussed within chapter 3, rules allow conclusions to be logically deduced from existing information. This notion of logical deduction is well suited to the disambiguation of *identity web references* for a given individual. For instance, rules constructed from digital identity information describing the person - i.e., his/her email address, web address, etc - can then be applied to web resources to detect the presence of his/her personal information, and therefore infer that the web resource cites the individual. This logical process is similar to the cognitive process by which web users often perform disambiguation of web resources which cite a given person who has an ambiguous name - e.g., does the web page contain the person's name? Does the web page contain the email address of the person?

Defining such logical structures from which inferences can be made requires the construction of rules to match information within a web resource. Given the structured data provided from converting web resources into metadata models (using the methodologies laid out in chapter 6), rules are ideally suited for application to such knowledge structures. One of the advantages of rules is their strict nature. The syntactic constraints placed within the antecedent of a rule (the conditions within the *if* part) require that the antecedent's premises are completely satisfied for the consequent (the outcome within the *then* part) to be true. One of the major problems with learning rules and their creation, if handcrafted, is its expensive and time consuming nature. In certain cases domain experts are required to compile a rule base in order to ensure accurate matches, and correct derivation of consequents. This chapter presents a disambiguation technique which is able to construct rules from the seed data according to the structural constraints of this information. A rule construction technique is presented which constructs a rule base from a limited example set - provided by the seed data - thereby employing a *supervised* technique.

This chapter is structured as follows: section 7.2 presents related work, detailing various strategies for inducing rules and presents several examples of rule-based classification systems. Section 7.3 outlines requirements, gleaned from the literature and related work, which a rule-based technique must fulfil. Section 7.4 presents the proposed disambiguation technique using rules, describing in detail the rule building process, presenting several example

rules constructed from seed data and detailing how rules are applied. Section 7.5 completes the chapter by summarising the presented technique.

7.2 Related Work

Rule-based classification systems are comprised of a two-stage procedure; first rules are built from known information, supplied as labelled examples, and then the rules are applied to data in order to classify it. Techniques for building rules often use machine learning to induce rules from known and labelled examples - induction is the process of learning from known facts which in this case builds a rule to cover them. Two main rule induction strategies exist: general-to-specific induction and specific-to-general induction. As these terms suggest, the former strategy creates general rules and then specialises those rules, while the latter creates specific rules and then gradually generalises the rules.

7.2.1 Rule Construction

General-to-specific rule induction techniques use a sequential covering strategy by gradually specialising a general skeleton rule resulting in coverage of the example set for a given class label. For example, the CN2 algorithm [CN89] is provided with a set of class labelled examples, from the examples CN2 constructs a general rule for that class (i.e., denoted by the empty set within the antecedent of the rule). This rule is then gradually specialised by adding attribute-value pairs to the antecedent (e.g., $\langle name, Matthew\ Rowe \rangle$). The FOIL algorithm [QCj93] functions in a similar manner by specialising a general skeleton first-order rule using literals (e.g., a predicate *parent* to be associated with variables (e.g., $parent(?x, ?y)$) - which are most *gainful* to the antecedent of the rule. Both CN2 and FOIL use a greedy approach by selecting the premise to add to the antecedent of the rule which maximises gain. As a result highly precise rules are created from the labelled examples which may be ineffective over those examples yet to be classified. General-to-specific induction strategies begin with general rules that underfit the example set. Therefore such strategies attempt to fit the rules to the data by increasing the coverage of the rule sequentially.

Specific-to-general rule induction algorithms employ a simultaneous covering strategy by constructing many rules simultaneously from the examples and then generalising those rules. For instance the FIND-S algorithm [Mit97] begins by constructing the most specific rule for a given class label, this rule is then generalised to cover more examples. The RISE algorithm [Dom95] generates specific rules for each class labelled example, for each rule the nearest examples not covered by the rule are then gathered using a selection heuristic, and the rule is generalised (by removing literals) to cover those examples. The C4.5 algorithm [Qui83] functions by creating a decision tree from the labelled examples, where each decision node represents an attribute decision and the paths from those nodes depicting the value of the attribute. C4.5 converts paths from the root node to the end nodes into rules where the antecedent is comprised of the attributes (nodes) and values (edges) along the paths. Specific-to-general induction strategies begin with specific rules that overfit the example set. The premises within the antecedent of these rules are removed until the rules fit the example set for that class label.

7.2.2 Rule-based Classification Systems

Various classification approaches rely on dynamically inducing rules for information classification. An approach presented in work by [HF04] induces rules and then utilises those rules to classify web pages by partitioning them into multiple classes. Rules are induced for classifying web pages in [MLK06] using the C4.5 algorithm by constructing rules from web pages represented as graphs. Web page classification is also performed in an approach by [SM00] which induces rules using FOIL where rules detect the presence of hyperlinks between pages as discriminative features. A general-to-specific rule induction strategy is employed in [SHH02] to induce classification rules for labelling web pages with categorical labels, thereby assigning metadata descriptions to web pages.

7.3 Requirements for Rule-Based Disambiguation Techniques

The discussion of current rule induction strategies and their use within Web based classification scenarios explains the limitations and requirements of such approaches. Utilising such techniques within the domain of disambiguating *identity web references* imposes constraints on the need for seed data which is sufficient in size such that rules may be built. Rules must also consider features which are beneficial to the disambiguation process and therefore yield a trade-off between specificity and generalisation. In short a rule-based disambiguation technique must address the following requirements:

- *Rules must not overfit or underfit examples*

Inducing rules from seed data must achieve a level of specialisation which ensures the correct identification of an *identity web reference*, whilst being sufficiently general that web resources which are web references are missed. Therefore this provides the need for rules which are both specific - to match known information - and general - to match new examples.

- *Build rules from limited examples*

Current state of the art approaches to rule induction from limited examples rely on users to provide a feedback mechanism in order to guide the induction process, thus providing an active learning strategy. Within the context of *identity web reference* disambiguation such feedback mechanisms are impractical - given that disambiguation techniques must be automated. Therefore techniques are required which are able to induce rules from example sets which are limited in size, whilst rich in features - this is apparent if one considers the makeup of the provided seed data (a single Linked Data social graph containing many features describing the digital identity of an individual).

- *Ensure suitable features are used to construct rules*

State of the art rule induction algorithms and rule-based classification systems induce rules according to feature patterns detected from a set of examples. Therefore given the seed data provided from the reconciliation of digital identity fragments, rules must be built according to features which are present within this data.

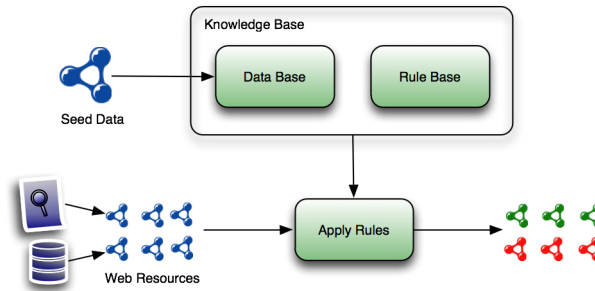


Figure 7.1: A proposed technique to build and apply inference rules to disambiguate *identity web references*

7.4 Inference Rules: Rule-based Disambiguation

The technique for rule-based disambiguation presented within this section uses the background knowledge - describing the digital identity of a single person - to construct Inferences Rules. Should information appear in a web resource which is related to the given individual then the rules match that information and infer that the web resource cites the person, and is therefore an *identity web reference*. The rule-based technique which is presented hereafter consists of several key elements, an overview of which is shown in Figure 7.1. The first component is the data base which is initially populated from the seed data (supplied from techniques in chapter 5) modelled as RDF, containing all the background knowledge about the individual whose *identity web references* are to be disambiguated. The second component is the rule base which is populated by constructing rules from the seed data. Together these components form the knowledge base containing information about the person together with mechanisms for inferring web citations. Disambiguation functions by matching information related to the individual in a web resource model. Web resources which are found to not contain any such information are therefore deemed to not refer to the individual.

At a low-level the rules contain triples within the antecedent which when matched against the triples within the metadata model of the web resource detect information which is consistent with information known about the individual whose *identity web references* are to be found. The following section discusses the construction of the rule base, the selection of features from the available example set and demonstrates how rules are applied.

7.4.1 Constructing the Rule Base

Rules must be constructed from the supplied background knowledge that are accurate, whilst also offering sufficient data coverage. For instance, imagine a rule is compiled that infers an *identity web reference* if a web resource contains a given person's name and their email address and their homepage and information about where that person lives. If such a rule were to match information within a web resource then one could confidently deduce that the web resource refers to that person. However the inclusion of that amount of information reduces the possibility of a match occurring, as the rule is too specific. Conversely if the rule is too general, then large scale data coverage is achieved yet the rule achieves low levels of precision (e.g., *If a webpage contains a person's name then it mentions that person*).

Therefore when constructing rules from known background knowledge it is essential that the rules are not too general to preserve precision whilst not being too specific to preserve data coverage (identified by recall levels).

Existing rule induction strategies, as described in the state of the art for this chapter, rely on a set of examples from which consistent feature patterns can be recognised. As stipulated previously, one of the requirements of a rule-based disambiguation technique is that it must be able to cope with a limited number of examples (given the limited size of the seed data). In such a scenario existing rule induction algorithms such as FOIL [QCj93] and C4.5 [Qui83] are not directly applicable to building a rule base. For instance, FOIL employs an accuracy measure to evaluate a given rule thus ensuring that it reaches a limit of specialisation. This measure can only function when several examples are available. Similarly C4.5, or rather specific-to-general strategies, also rely on entropy measures over a set of examples.

7.4.1.1 Instance Extraction

Background knowledge is provided in the form of a social graph defined using RDF, which describes both the biographical and social network information of a given person. This single RDF model provides the solitary example from which rules can be constructed - given that this is the only information which is known to describe the person whose *identity web references* are to be disambiguated - thereby limiting the ability of state of the art rule induction techniques such as FOIL and C4.5 to function effectively. Rather than relying on a large example set, rules are instead constructed from RDF instances within the social graph.

An RDF instance represents a resource within a given RDF model which can either be an anonymous node or identified by a URI. An instance is a unique object, which in the case of the background knowledge derived from the Social Web, can be a social network member - identified as an instance of *foaf:Person* - or a location related to a given person - identified as an instance of *geo:Feature*. Such instances form a useful basis for inferring a web resource as referring to an individual, given that if information describing the instance is also found within a web resource then the web resource can be inferred as citing the person. For example if a person appears in a web page with another person from their social network then one can deduce that the web page cites that person. To ensure that such information is detected within web resources, RDF instance descriptions must be gathered from within the social graph. An RDF instance description forms contextually relevant information about a given resource which surrounds the instance within an RDF model. Gathering such descriptions requires extracting a triples set from within the RDF model, however decisions must be made when collecting this set about the depth, or distance, which should be considered when gathering the triples.

Work by [GEP08] has investigated this issue by proposing several instance extraction techniques, each varying in their strategy for extracting information attributed to a given resource. One such technique, known as *Naive Instance Extraction*, is employed to extract the instance description belonging to a given resource r . This technique functions by selecting the immediate properties of the resource, thereby producing a set of triples T_r . However this strategy includes triples which link two instances together (e.g., $\langle ?p \text{ foaf:knows } ?q \rangle$ which relates two instances of *foaf:Person*). Instead such triples must be ignored such that the

leveraged triplesets contain only leaves (literals and resources which are not RDF instances) of the resource within the RDF model. This ensures that RDF instances are considered independently and are not reliant on one another; for instance a rule may be created which requires that a social network member is present within a web resource together with the location information of that person. Such restrictions, although they ensure precise matches, reduce the generalisation of the rule and its applicability. Therefore the Resource Leaves are extracted surrounding the resource (r) within a given RDF graph (G) using the following formal construct:

$$RLS_G(r) = \{ \langle r, p, o \rangle \mid \langle r, p, o \rangle \in G \wedge \nexists p', o' \langle o, p', o' \rangle \in G \} \quad (7.1)$$

The above construct functions by selecting all the triples ($\langle r, p, o \rangle$) attributed to a given resource (r) from the RDF model/graph (G) where the object (o) of each triple does not act as the subject of other triples ($\langle o, p', o' \rangle$). This returns a list of resources and literals which form leaves surrounding the resource (r) such that no paths exist beyond those leaves. To clarify the application of the above construct, consider the example shown in Figure 7.2 where on the left is a social graph with an instance of a person - denoted by the hash URI `#me` - who is associated with a person they know - expressed by the hash URI `#samchapman`. Applying the Resource Leaves construct to the graph identifies resources within the graph (`#me` and `#samchapman`) and then extracts triples associated with those resources which describe attributes (i.e., triples where the object is not the subject of another triple). On the right of Figure 7.2 is the extracted resource descriptions using Equation 7.1. In comparison, should the *Naive Instance Extraction* method have been employed to extract instance descriptions then the instance description of `#me` would have contained the triple $\langle \#me, \text{foaf:knows}, \#samchapman \rangle$. As will be discussed shortly, the inclusion of such relations in the instance description is not required for constructing rules.

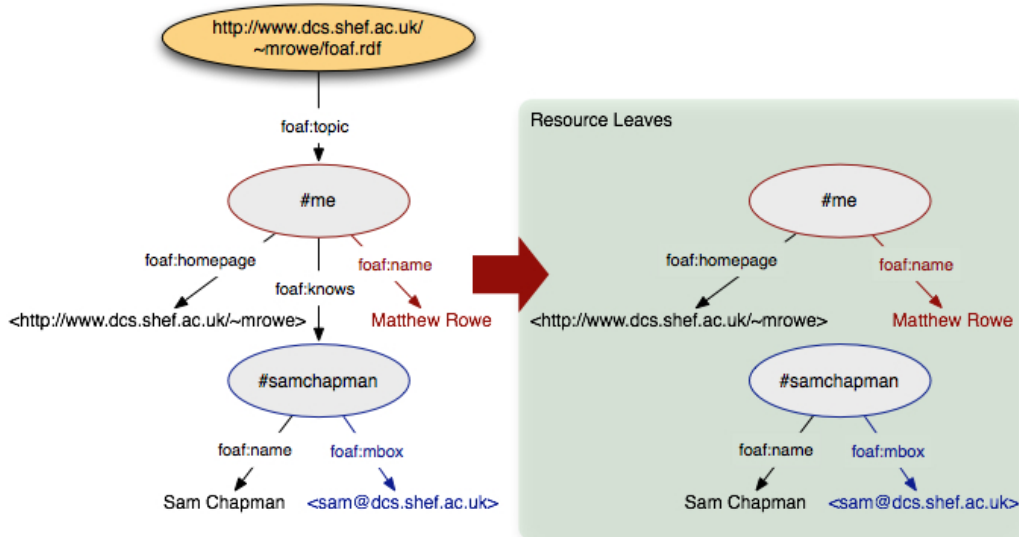


Figure 7.2: Applying the Resource Leaves construct to extract RDF instance descriptions from a given metadata model.

Algorithm 5 $\text{buildRules}(r_p, G)$: Induces rules from RDF instances. Input to this algorithm is the social graph (G) and the person whose web resources are to be disambiguated (r_p). Rules are induced and added to the rule base (RB).

Input: G, r_p

Output: RB

```

1:  $RB = \emptyset$ 
2: for each resource  $r \in G$  do
3:    $R = \text{CONSTRUCT } \{ \langle r_q \text{ foaf:page } ?url \rangle \text{ WHERE } \{ \langle r_p \text{ foaf:name } ?n \rangle, \langle ?url \text{ foaf:topic } ?p \rangle, \langle ?p \text{ foaf:name } ?n \rangle \}$ 
4:    $T_r = RLS_G(r)$ 
5:   for each  $\langle s, p, o \rangle \in T_r$  do
6:     if ( $\langle r \text{ rdf:type foaf:Person} \rangle \ \&\& \ (r \neq r_p)$ ) then
7:       Add  $\langle r_p \text{ foaf:knows } ?q \rangle, \langle ?q \text{ p } ?o \rangle, \langle ?url \text{ foaf:topic } ?x \rangle, \langle ?x \text{ p } ?o \rangle$  to WHERE clause of  $R$ 
8:       if ( $\langle p \text{ rdf:type owl:inverseFunctionalProperty} \rangle$ ) then
9:          $Q = \text{CONSTRUCT } \{ \langle r_q \text{ foaf:page } ?url \rangle \text{ WHERE } \{ \langle r_p \text{ foaf:name } ?n \rangle, \langle ?url \text{ foaf:topic } ?p \rangle, \langle ?p \text{ foaf:name } ?n \rangle \}$ 
10:        Add  $\langle r_p \text{ foaf:knows } ?q \rangle, \langle ?q \text{ p } ?o \rangle, \langle ?url \text{ foaf:topic } ?x \rangle, \langle ?x \text{ p } ?o \rangle$  to WHERE clause of  $Q$ 
11:         $RB = RB \cup Q$ 
12:      end if
13:    else if ( $\langle r \text{ rdf:type foaf:Person} \rangle \ \&\& \ (r = r_p)$ ) then
14:      Add  $\langle r_p \text{ p } ?o \rangle, \langle ?p \text{ p } ?o \rangle$  to WHERE clause of  $R$ 
15:      if ( $\langle p \text{ rdf:type owl:inverseFunctionalProperty} \rangle$ ) then
16:         $Q = \text{CONSTRUCT } \{ \langle r_q \text{ foaf:page } ?url \rangle \text{ WHERE } \{ \langle r_p \text{ foaf:name } ?n \rangle, \langle ?url \text{ foaf:topic } ?p \rangle, \langle ?p \text{ foaf:name } ?n \rangle \}$ 
17:        Add  $\langle r_p \text{ p } ?o \rangle, \langle ?p \text{ p } ?o \rangle$  to WHERE clause of  $Q$ 
18:         $RB = RB \cup Q$ 
19:      end if
20:    else if ( $\langle r \text{ rdf:type geo:Feature} \rangle \ \&\& \ (\langle r_p \text{ foaf:based_near } r \rangle)$ ) then
21:      Add  $\langle r_p \text{ foaf:based_near } ?g \rangle, \langle ?g \text{ p } ?l \rangle, \langle ?p \text{ foaf:based_near } ?h \rangle, \langle ?h \text{ p } ?l \rangle$  to WHERE clause of  $R$ 
22:    end if
23:  end for
24:   $RB = RB \cup R$ 
25: end for
26: return  $RB$ 

```

7.4.1.2 Building Rules

The compiled rule base contains SPARQL rules which are formed using a general-to-specific strategy inspired by FOIL. Only a single positive example is used - the social graph - for building the rule base. The strategy for building rules works according to Algorithm 5, which will now be explained. Each resource is extracted from the supplied social graph (seed data) (line 2). A skeleton rule is then created for each resource (line 3) comprised of the name of the social graph owner - the person whose *identity web references* are to be found - together with triples identifying a person within a web resource with the same name. The Resource Leaves construct ($RLS_G(r)$) is then used to extract a triples set for the resource (r) from the social graph (line 4) - this forms the information which is used to build the rules. The algorithm then goes through the collection of triples and builds the rules as follows: if the resource (r) is an instance of *foaf:Person* and is not the social graph owner - and therefore a member of his/her social network - then the triple ($\langle s, p, o \rangle$) is added to the antecedent of the skeleton rule R (line 7). Additional triples are added to relate the social graph owner with the person and relate a person within a web resource with the object of the triple. At this stage, a rule would be created which matches a member of the social graph owner's social network by finding a person within a web resource with the same name (if in this instance $\langle s, p, o \rangle$ defines a person's name i.e. $\langle ?p, \text{foaf:name}, ?n \rangle$). The predicate (p) of the triple ($\langle s, p, o \rangle$) is also checked. If it is inverse functional (line 8) then another rule (Q) is created using the same skeleton construct from earlier and the addition of triples to the antecedent of Q . This is then added to the rule base RB . Q resembles a rule which is specific to the resource (r) such that the discovery of the value (o) of the property (p) uniquely identifies the resource in a web resource - e.g., a rule would match the homepage

of the social graph owner's friend.

If the resource which is being analysed is the social graph owner (line 13) then the triple ($\langle s, p, o \rangle$) from the tripleset of r is added to the antecedent of R (line 14) by directly relating the social graph owner with the triple - rather than using *foaf:knows* as done previously. At this stage the rule would match information describing the social graph owner within a given web resource. Similar to above, if the predicate of the triple defines an inverse functional property then a separate rule Q is created using the skeleton rule from before and $\langle s, p, o \rangle$. This rule (Q) is then added to the rule base.

If the resource which is being analysed denotes a location (line 20) and the location is the same as the social graph owner's location then the details of the location are added to the rule. At this stage an example rule would match the place name within a given web resource which is the same as the social graph owner's location together with the name of the social graph owner.

The cycle of populating the rule R is repeated until all the triples of the resource r have been analysed. Essentially the strategy functions by populating the rule with all the triples such that the structure of the social graph can be utilised to match information within web resources. The rule (R) once complete is added to the rule base (line 24). Once all resources within the seed data have been analysed and rules compiled then the rule base is returned. The following section presents example rules which are created from this technique.

7.4.2 Example Rules

The construction of rules from a given social graph uses a general-to-specific process and provides rules which are able to detect information within web resources which is consistent with digital identity of a given person. Variables are used within the rules to detect the presence of information which is associated with the social graph owner. Several example rules are now presented, each of which use an example social graph (Appendix B).¹

7.4.2.1 Inverse Functional Property Rules

Separate rules are induced from predicates within instance descriptions which uniquely define a given resource. For example, utilising this information in such a way can produce a rule which matches the presence of the social graph owner's name together with his/her homepage within a web resource. The rule functions by matching triples in the antecedent of the rule with triples within the RDF model of the web resource to confirm that the web resource cites the social graph owner². Using SPARQL rules allows graph patterns to be included in the **WHERE** clause of the rule such that the pattern creates a link between the social graph owner and the information:

¹Note that the URIs within the Social Graph are dereferenced and the relevant information populated within each instance description.

²The URI ($\langle \text{http://www.dcs.shef.ac.uk/~mrowe/foaf.rdf\#me} \rangle$) of the social graph owner is used hereafter within the examples to illustrate the association of the rule's triples to the individual.

```

PREFIX foaf:<http://xmlns.com/foaf/0.1/>
CONSTRUCT { <http://www.dcs.shef.ac.uk/~mrowe/foaf.rdf#me> foaf:page ?url }
WHERE {
    <http://www.dcs.shef.ac.uk/~mrowe/foaf.rdf#me> foaf:name ?n .
    <http://www.dcs.shef.ac.uk/~mrowe/foaf.rdf#me> foaf:homepage ?h .
    ?url foaf:topic ?p .
    ?p foaf:name ?n .
    ?p foaf:homepage ?h
}

```

One of the intuitions behind the use of seed data obtained from the Social Web, is that a given person - whose identity web references are to be found - will appear within a web resource with other people who are members of his/her social network. Therefore rules are required which detect the appearance of a person together with members of his/her social network. The previously described strategy for inducing rules allows the creation of such mechanisms by adding triples from the resource descriptions of social network members into a rule. This would therefore create a rule which attempts to find a person from the social graph owner's social network within a web resource by matching the value of the literal denoting the person's name and the resource defined by their email address within a web resource's metadata model as follows:

```

PREFIX foaf:<http://xmlns.com/foaf/0.1/>
CONSTRUCT { <http://www.dcs.shef.ac.uk/~mrowe/foaf.rdf#me> foaf:page ?url }
WHERE {
    <http://www.dcs.shef.ac.uk/~mrowe/foaf.rdf#me> foaf:name ?n .
    <http://www.dcs.shef.ac.uk/~mrowe/foaf.rdf#me> foaf:knows ?p .
    ?p foaf:homepage ?h .
    ?url foaf:topic ?q .
    ?url foaf:topic ?r .
    ?q foaf:name ?n .
    ?r foaf:homepage ?h
}

```

7.4.2.2 Person Rules

A continuation of the reasoning behind using seed data from the Social Web demonstrates that another rule induced from the example social graph matches the presence of a person's name, together with the presence of the name of a social network member. This simply detects the co-occurrence of the name of the person whose *identity web references* are to be found and the name of an individual which that person knows within the knowledge structure of a given web resource.

```

PREFIX foaf:<http://xmlns.com/foaf/0.1/>
CONSTRUCT { <http://www.dcs.shef.ac.uk/~mrowe/foaf.rdf#me> foaf:page ?url }
WHERE {
    <http://www.dcs.shef.ac.uk/~mrowe/foaf.rdf#me> foaf:name ?n .
    <http://www.dcs.shef.ac.uk/~mrowe/foaf.rdf#me> foaf:knows ?p .
    ?p foaf:name ?o .
    ?url foaf:topic ?q .
    ?url foaf:topic ?r .
    ?q foaf:name ?n .
    ?r foaf:name ?o
}

```

Rules are also induced from RDF instance descriptions which contain the complete triple-set within the antecedent of the rule. This forms a maximally specific rule for a given resource by detecting the complete instance description:

```
PREFIX foaf:<http://xmlns.com/foaf/0.1/>
CONSTRUCT { <http://www.dcs.shef.ac.uk/~mrowe/foaf.rdf#me> foaf:page ?url }
WHERE {
    <http://www.dcs.shef.ac.uk/~mrowe/foaf.rdf#me> foaf:name ?n .
    <http://www.dcs.shef.ac.uk/~mrowe/foaf.rdf#me> foaf:homepage ?h .
    <http://www.dcs.shef.ac.uk/~mrowe/foaf.rdf#me> foaf:mbox ?e .
    ?url foaf:topic ?p .
    ?p foaf:name ?n .
    ?p foaf:homepage ?h .
    ?p foaf:mbox ?e
}
```

7.4.2.3 Location Rules

So far only instances of *foaf:Person* within the social graph have been discussed. Another example of an RDF instance present within the model is an instance of *geo:Feature* describing the location of the social graph owner. As described in §7.4.1 a triple-set is compiled from this instance description and a rule is induced from this set which matches the presence of the person's name and location within the metadata model of a given web resource:

```
PREFIX foaf:<http://xmlns.com/foaf/0.1/>
PREFIX geo:<http://www.geonames.org/ontology/>
CONSTRUCT { <http://www.dcs.shef.ac.uk/~mrowe/foaf.rdf#me> foaf:page ?url }
WHERE {
    <http://www.dcs.shef.ac.uk/~mrowe/foaf.rdf#me> foaf:name ?n .
    <http://www.dcs.shef.ac.uk/~mrowe/foaf.rdf#me> foaf:based_near ?g .
    ?g geo:name ?l .
    ?url foaf:topic ?p .
    ?p foaf:name ?n .
    ?p foaf:based_near ?h .
    ?h geo:name ?l
}
```

The above rules are only basic examples of what is built from a single social graph. The induction process entirely depends on what RDF instances are present within the seed data and what triples are used within the instance descriptions. It may be case that several social network members have information defining their name and email, whereas others may only have their name. In both cases the rules are induced to include variables (e.g., *?n*) thus providing a structure within the antecedent which must be matched. This strategy for building rules from RDF instances is employed under the intuition that instances resemble unique objects, or resources, which are related to a given individual. Leveraging descriptions of these instances and creating rules allows features associated with the social graph owner to be utilised to identify *identity web references* by matching features present within a given web resource.

7.4.3 Applying Rules

To contextualise how rules can be applied to RDF models of web resources, this section considers the scenario depicted in Figure 7.3. Rules are built from RDF instances within

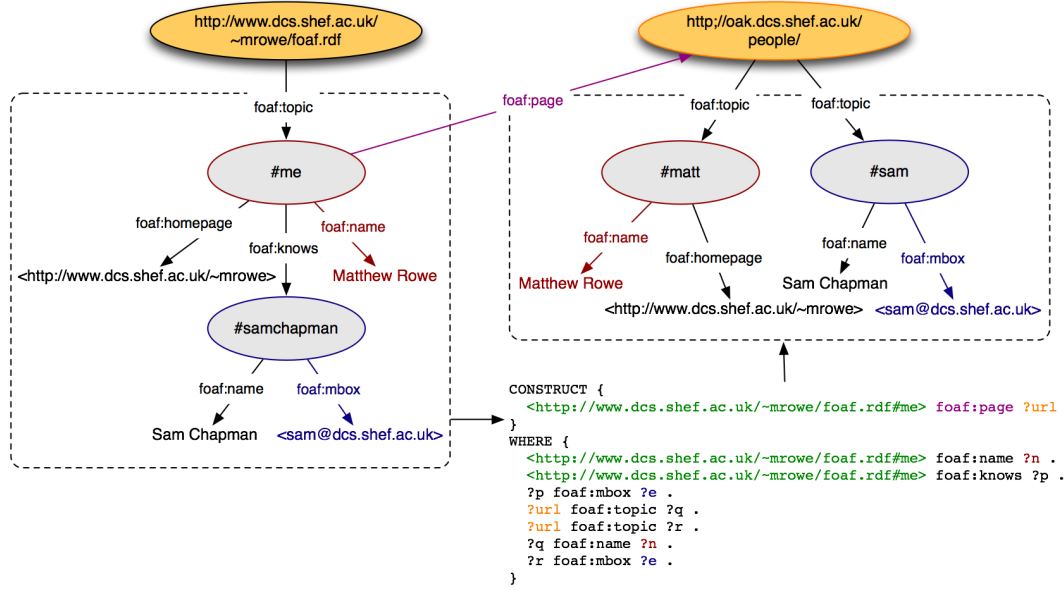


Figure 7.3: Applying a rule to the RDF model of <http://oak.dcs.shef.ac.uk/people> matching a social network member's email address in the web resource

the seed data, which are then applied to metadata models of web resources in order to match information, and infer an *identity web reference*. For instance, a rule built from the seed data first matches the social graph owner's name within a given web resource, and then detects the email address of a member of the social graph owner's social network. The consequent of the rule then infers a relation between the URL of the web resource and social graph owner - therefore defining the web resource as citing the person. Figure 7.3 shows how the metadata model of the web resource provides a common interpretation of information within both the web resource and the seed data, and as such allows the rule to match information found within the social graph.

Another example application of the constructed rules is shown in Figure 7.4 by applying a different rule to the same web resource as used in Figure 7.3. In this instance however another the rule infers an *identity web reference* by finding the social graph owner's name and homepage within the metadata model of the web resource. The consequent is the same as before, and infers a relation between the social graph owner and the web resource, and therefore the presence of a web reference for the individual.

7.5 Summary

This chapter has presented a rule-based disambiguation technique which builds a set of rules from a limited amount of seed data using a general-to-specific strategy similar to algorithms such as FOIL and CPAR. The algorithm for constructing the rule base produces rules which match specific information to RDF instances - and therefore resources - from within the supplied seed data modelled as a social graph. By exploiting the semantics of the information within the RDF instance descriptions, specific rules are built to match unique properties (i.e., detecting the email address of a social network member within a web resource) of those instances. Conversely the available information within the social graph

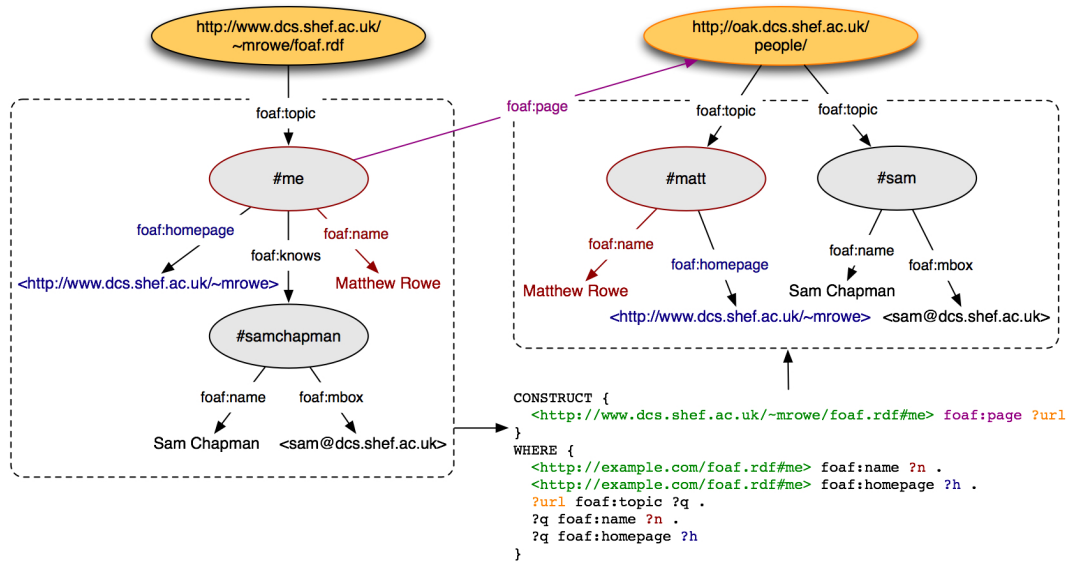


Figure 7.4: Applying a rule to the RDF model of <http://oak.dcs.shef.ac.uk/people/> matching the social graph owner’s homepage and name

dictates the structure of the rules and their applicability. RDF instances which only contain the name of a social network member allow rules to be induced which search for the presence of a person’s name within a web resource.

Building rules based on first-order predicate logic allows the inclusion of variables and the application of rules which attempt to discover relations between an individual and information within a given resource. Disambiguation is therefore performed by taking the set of rules and applying them to the metadata models of the web resources. Representing web resources in a common format with the social graph facilitates the common interpretation of information in either source over which rules can be applied. This application allows automated inferencing of the existence of an *identity web reference* by matching information which is consistent with an individual’s seed data - representing his/her digital identity - with information within a given web resource.

Chapter 10 presents a detailed evaluation of this technique with respect to several baseline measures and two alternative disambiguation techniques. Based on these results the requirements listed for this technique are discussed in the concluding discussion of this thesis in chapter 11. The following chapter presents one of these alternative techniques which employs a graph-based technique to disambiguate *identity web references*.

Chapter 8

Graph-based Disambiguation

8.1 Introduction

Graph-based disambiguation techniques capitalise on data structures to construct a graph-space and then analyse the characteristics of the graph in order to partition nodes. In recent years - as will be demonstrated within the following section discussing the current state of the art - graph-based disambiguation techniques have been extensively used to disambiguate content found on the World Wide Web, largely due to the intrinsic graph-like structure of the Web - i.e., hyperlinks connecting web pages. Despite this rise in uptake and application no work has explored the effects of using graph-based disambiguation techniques over a graph-space formed from knowledge structures such as RDF. Within the context of this thesis both seed data and the set of web resources, which are to be analysed by disambiguation techniques, are provided in a common graph-based format. The provision of information in such a form provides a clear motivation for the investigation of how graph-based disambiguation techniques can be applied to such knowledge structures. This chapter discusses current state of the art techniques by explaining their methods for graph-space compilation, together with the methodology by which the graph-space is harnessed for disambiguation. A graph-based disambiguation technique is presented which uses a Markov Random Walks model to traverse a graph and measure distances between nodes from which clustering is then applied. This method provides a *semi-supervised* disambiguation technique by using some prior knowledge - provided by the seed data - together with a large portion of unlabeled data - provided by the set of web resources to be analysed - to perform disambiguation.

This chapter is structured as follows: section 8.2 presents current graph-based disambiguation techniques, explaining how such techniques construct a graph-space and exploit it to facilitate the disambiguation process. Section 8.3 details the requirements gleaned from analysing the literature, explaining how the limitations present in current graph-based disambiguation techniques must be examined and overcome. Section 8.4 presents the investigated disambiguation technique utilising a first-order Markov chain model known as Random Walks to derive transition probabilities between nodes within the graph-space. These transition probabilities are then employed to calculate distance measures between nodes and cluster proximal nodes. This chapter concludes with section 8.5 which presents a summary of the techniques discussed within this chapter.

8.2 Related Work

A graph is defined formally as containing a set of nodes, or vertexes, V and a set of edges E which connect those nodes (e.g., $e_{ij} \in E$): $G = \langle V, E \rangle$. Graph-based disambiguation techniques function by constructing a graph-space and then using the properties of the graph-space to perform disambiguation. For instance the technique described in [CMHJ⁺09] investigates the problem of entity disambiguation within the web of Linked Data to address the problem of distinct entities (resources) being given the same URIs. A graph-space is constructed from the Linked Data Web, where entities (resources) represent nodes and edges represent the relations between entities expressed as equivalence (e.g., *owl:sameAs*) or non-equivalence links. The technique infers relationships between entities using a probabilistic model constructed from the graph-space, which supports the inference of relations between entities overriding the currently prescribed edge labels, given that these are assumed to be approximated (i.e., *owl:sameAs* links maybe inaccurate).

Another graph-based technique described by [IXZ07] performs identity disambiguation from a set a web pages - segmenting the collection of pages into groups where each group cite a given person. Various features are extracted from the set of web pages such as named entities, hyperlinks, and lexical tokens. Each feature label is modelled as an edge in the graph-space, thereby connecting the web page to the value of the feature; for example, an edge may be labelled *mentions_person_name* and the connected node denotes the name of that entity. This allows common features within the graph-space to interlink web pages with one another. Traversals are then made through the graph-space using the Random Walks model to record the cost of moving through the graph-space and visiting certain nodes on the way. Similar web pages are then clustered together based on the traversal costs. The use of Random Walks to traverse and record paths through a graph-space is also investigated in [Mal05] using a graph-space constructed from web pages linked together by common person names. The intuition being that similar web pages, which all refer to the same person, can be clustered together based on the similarity of social network information.

Graph-based disambiguation is performed in [BM05]; given a set of web resources to be disambiguated, an interlinked graph-space is compiled from hyperlinks between web resources, where the hyperlinks "*share something in common*". Clustering is then used to group similar web resources based on their link structures, by comparing linked components within the graph-space for similarity. An approach presented in [AW06] uses a partially labelled graph to bootstrap the classification of unlabelled nodes within the graph-space. A technique known as *relaxation labelling* is then used to correct labelled nodes within the graph-space based on the labels of their neighbouring nodes. The intuition being that the linked graph-space represents commonalities between neighbouring nodes and therefore the labelling should reflect this.

Graph-based classification of web pages is explored in [SLBK03] by using graph formalisations to capture the information structure of web pages. Web pages are then partitioned into clusters using the k -Nearest-Neighbours clustering algorithm. Clustering of web pages using their underlying graph structure is also addressed in [Ai05] by forming a graph-space from web pages connected by hyperlinks and then partitioning web pages using their semantic similarity.

Identity disambiguation is performed in work by [JWA⁺09] to partition web pages into

clusters - where each cluster contains web pages referring to a given person. Features are extracted from web pages according to several person attributes (referred to as tags): *person name*, *organisation*, *location*, *email address*, *phone number* and *date*. A graph-space is built where nodes represent unique features (e.g., a person name) and edges represent the appearance of a feature within the web page (e.g., *web page*, *contains*, *person name*). Clustering is then used to partition the graph-space into maximally connected cliques, the tag with the highest cohesion within that clique is then used as the identifier of the cluster (i.e., the most frequently appearing attribute in the cluster).

Another graph-based disambiguation approach is described in [KCMNT08]. A graph-space is constructed using named entities as nodes found within a web resource as features and edges representing the relation between the resource and the feature - in the same manner as work by [JWA⁺09] and [IXZ07]. Agglomerative clustering is then used to segment the graph-space into partitions where each partition refers to a given person. The clustering algorithm uses a measure of connection strength between web resources which is derived by measuring the attraction between two nodes within the graph-space - based on the features they share (the greater the number of features the larger the attraction). A threshold is then tuned for clustering such that nodes with similarities over the threshold are kept together.

Identity disambiguation is addressed within the context of citation listings - given that publication authors often share the same names and abbreviations - in work by [HZG05]. The graph-space is constructed such that each node is a citation. A feature vector comprised from the vector space model of citation attributes (*author names*, *paper title* and *publication venue*) is used to represent each node. Edges are formed and weighted between nodes using TFIDF similarities of each node's vector. The graph-space is then partitioned using spectral clustering by using the spectrum of the graph and its Eigen decomposition. As a graph is in effect a matrix, the eigenvectors are employed to partition elements in order to produce clusters where the elements within each cluster have a high level of similarity [BJ05] and elements within separate clusters have high levels of dissimilarity.

Similar work to [HZG05] by [OL07] has also applied spectral clustering to perform identity disambiguation. In this instance the graph space is formed from a collection of web documents, such that an individual node represents a single web document. Like [HZG05] each node is represented as a feature vector which in this instance is built using the TFIDF measure of each token within the document with respect to the vector space model of all tokens within the set of documents. Edges between nodes (and therefore documents) are weighted according to a TFIDF similarity function. Spectral clustering is then used to segment the graph space into clusters of highly similar nodes within each respective cluster.

The techniques discussed thus far use a consistent two-stage approach to disambiguation: first a graph-space is constructed from known features and second the graph-space is partitioned somehow. Compilation of the graph-space requires consideration over the level of granularity at which nodes are represented: either one node represents one web resource [HZG05, OL07] or a node represents a feature of the resource [JWA⁺09, KCMNT08, IXZ07]. Current techniques either group similar web resources by performing clustering of nodes within the graph-space, therefore requiring some comparison metric or measure of the nodes, or use neighbouring nodes to perform labelling [AW06]. One of the notable characteristics of the current state of the art is the lack of techniques which utilise already existing graph-structures such as knowledge representation formats. The rate of uptake and adoption of

technologies, such as RDFa and Microformats, requires that graph-based techniques are explored to exploit the growing inclusion of knowledge structures within web documents.

8.3 Requirements for Graph-based Disambiguation Techniques

The application of graph-based techniques discussed thus far suggests that such techniques are well suited to the problem setting investigated within this thesis - given the provision of seed data derived from reconciling digital identity fragments and the representation of web resources which are to be disambiguated as RDF. In order to apply a graph-based disambiguation technique over such knowledge structures, the following requirements must be addressed:

- *Provide an interlinked graph-space*

Disambiguation techniques can only function effectively if a graph-space is provided which connects web resources which are to be disambiguated by their common features. Therefore a method must be used to link resources which exploits correlations between web resources and creates an interlinked graph-space.

- *Ensure suitable features are used to build a graph-space*

Features play the role of interlinking nodes within the graph-space, thereby forming paths. Within the state of the art, the choice of node representation varies between approaches. A suitable strategy must therefore be used such that disambiguation can be performed by comparing nodes to derive suitable edges - i.e., by weighting edges appropriately. Given the provision of both the seed data and the set of web resources as RDF, decisions must be made as to what features of such metadata models are used.

- *Accurately partition the graph-space*

Forming a graph-space provides a model of correlated information from disparate web resources, denoted by the links within the space. The properties of the graph-space must be utilised in some manner to enable disambiguation to be performed. For instance, if clustering methods are to be used then the properties of the graph-space must be harnessed in some manner, whereas if a labelling technique is to be used then the properties of a portion of the graph must be considered.

8.4 Random Walks: Graph-based Disambiguation

By representing knowledge structures using Semantic Web technologies such as RDF, knowledge is formalised using a flexible graph structure. The subject-predicate-object format of triples corresponds to a node-edge-node graph triplet where the object of one triple is the subject of another triple. This provides a suitable setting from which an interlinked graph structure can be compiled and graph-based disambiguation techniques applied. Several state of the art graph-based disambiguation techniques use feature sets which are limited and do not fully capture the intrinsic properties of a web resource (e.g., only hyperlinks). Knowledge

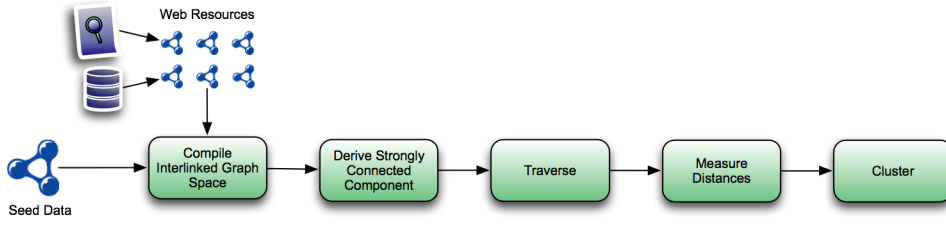


Figure 8.1: The pipeline architecture of the proposed graph-based disambiguation technique.

from within web resources, when represented as RDF, provides a rich feature set allowing resources to be linked together via many different common features, not merely hyperlinks.

Following the compilation of the graph-space, the graph can be partitioned using various similarity or dissimilarity measures. Examples of the latter are most common and are often reduced to a distance metric, as described in [WF94] as measuring the distance from vertex u to another vertex v as $d(u, v)$. The most basic measurement function is the shortest path between u and v , defined as the *geodesic*, where the shortest path length is composed from the number of traversed edges on the path. Distance (and thus dissimilarity) can be measured using the euclidean distance between nodes in a graph-space: following the compilation of a local similarity (adjacency) matrix the euclidean distance can be calculated as the direct distance from node u to another node v . Local similarity measures are utilised in work by [KCMNT08] which uses a notion of *connection strength* between two nodes via a summation of edges weights on the shortest path between two nodes. Similar work by [NL10] explores different node similarity measures in the context of word sense disambiguation, defining both local and global measures for analysis, where the former includes measures such as pagerank, centrality and betweenness and the latter includes compactness, entropy and edge density.

Simple graph-based distance metrics consider the shortest path between two nodes, however such metrics are not applicable to graph spaces which have a short diameter, or where object nodes are connected through a short path. Instead the number of possible paths must be considered in order to gauge dissimilarity. One solution is to apply a Random Walks model to measure distances between nodes by considering the weighting of edges together with the number of paths possible between nodes. This section investigates the use of such a model to measure distances between nodes within the graph-space and then apply clustering to group nodes based on proximity measures. The technique is comprised of five stages (see Figure 8.1): the first stage compiles an interlinked graph-space from the RDF models of the social graph and the set of web resources to be disambiguated. The second stage derives strongly connected components within the graph-space. The third stage assigns weights to edges and uses a Random Walks model to traverse the graph-space. The fourth stage measures distances within the graph-space, and the fifth stage clusters the web resources according to those distances. Clustering aims to partition the graph-space into two clusters: *positive* (web resources which cite a given person) and *negative* (web resources which do not cite a given person). The former cluster must therefore contain the web resources which are most similar to the social graph - represented by the provided seed data. Each of the above stages are now discussed in greater detail.

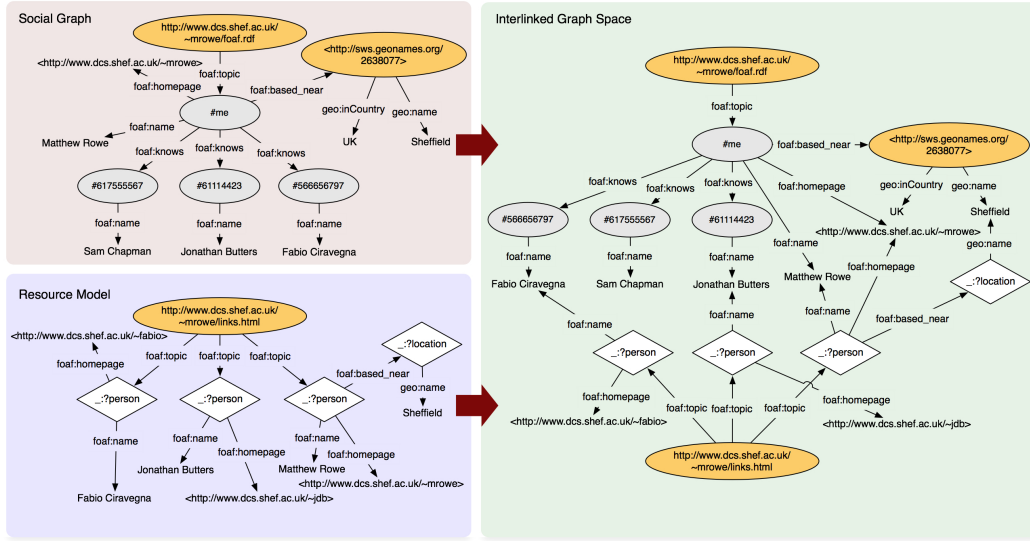


Figure 8.2: The graph-space compilation process

8.4.1 Building an Interlinked Graph Space

The graph-space is initially populated with the social graph of the person - whose references on the Web are to be found - and iteratively with the RDF model of each web resource to be disambiguated using an indexing paradigm. Nodes are represented in the graph-space by resources and literals from the RDF models and edges are represented in the graph by properties and relations. In effect, the feature set used to interlink web resources is comprised of literals and resources from RDF models, rather than RDF instances. This format of the compiled graph-space for the use of Random Walks as a traversal mechanism is similar to previous applications and their data structures - there are similarities between the fine grained feature set of individual web resources in [IXZ07] and the literals and resources present in RDF graphs. The use of a traversal mechanism, such as Random Walks, requires a multitude of paths of movement through the graph-space. Utilising literals and resources presents the necessary feature set to provide such paths.

The indexing paradigm functions by comparing the resources and literals from an RDF model, which is yet to be integrated into the graph-space, with all the resources and literals currently in the graph-space. If a match occurs, for example if an RDF model contains a person's name which already resides in the graph-space, then the target node of the edge in the RDF model is replaced with the matched node in the graph-space, thereby providing a link. It is common for literals to contain misspellings therefore the Levenshtein string similarity metric is used (using tuning in Appendix C). Moreover Levenshtein distance ignores the order of the characters within the string making it suitable for person names where the surname proceeds the first name (i.e., on publication listings). When comparing literals, if the derived measure is above an assigned threshold then a match is confirmed and the RDF model is integrated into the graph-space.

Resources within the RDF models, denoted by URIs, are compared for strict equivalence (e.g., geographical locations with a URI). There are many cases where anonymous nodes appear in RDF models (e.g., instances of *foaf:Person* with no URI), such nodes are ignored

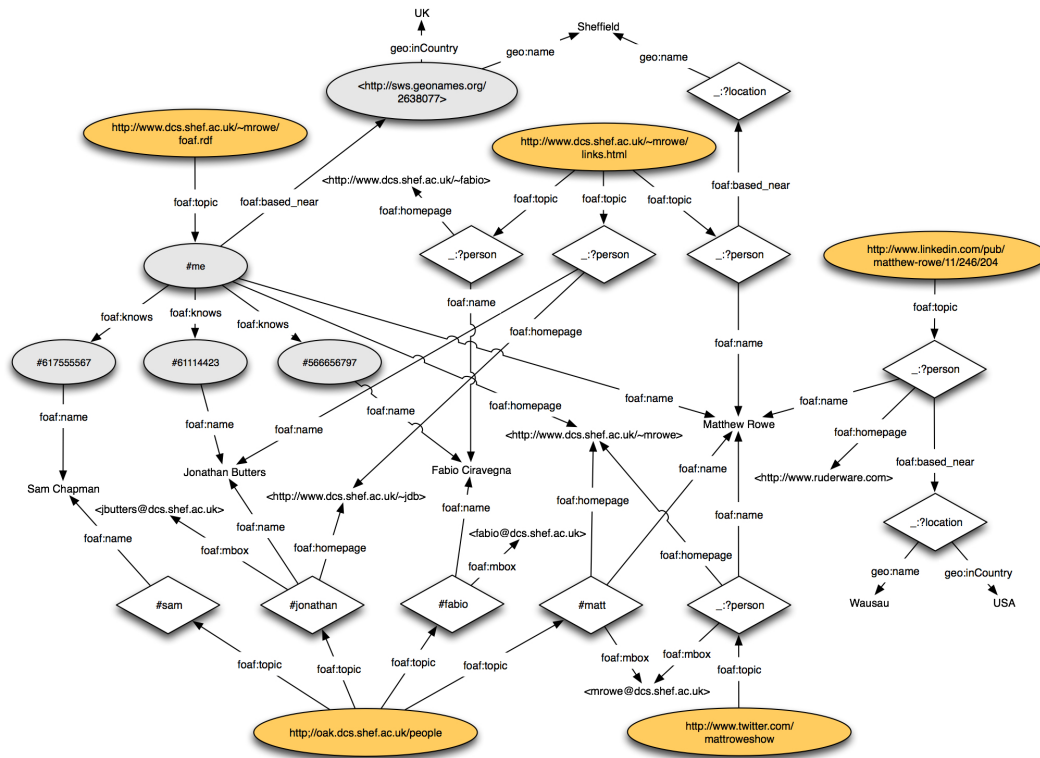


Figure 8.3: Snippet of a compiled graph-space. A complete graph-space would be too large to display graphically in a legible manner.

and are not compared with one another, as this could cause many edges to link to a single anonymous node. The process of populating the graph-space is performed until all the RDF models of the web resources have been integrated into the graph-space as shown in Figure 8.2. Each of the web resource models and the social graph have a document URI which is essentially the root of their respective subgraphs as shown in Figure 8.3. The nodes which contain these URIs will hereafter be referred to as the *web resource root node* and the *social graph root node* respectively.

8.4.2 Identifying Strongly Connected Components

Once the graph-space has been compiled, it may contain islands of connected nodes called strongly connected components (as shown in Figure 8.4). These islands are created due to the RDF models integrated into the graph-space only having common literals and resources with the metadata models of certain other web resources. Therefore the graph-space, following compilation, is not fully connected: this inhibits the ability of a walker from traversing across the full diameter of the graph-space. In order to perform traversals through the graph-space, the islands in the graph-space, which themselves are strongly connected components, must be identified.

These strongly connected components are derived using Tarjan's algorithm [Tar72]. This algorithm functions by starting at a given node within the graph-space and performing a depth first search, returning all connected nodes and edges from the starting node and therefore the strongly connected component which that node resides in. This process can

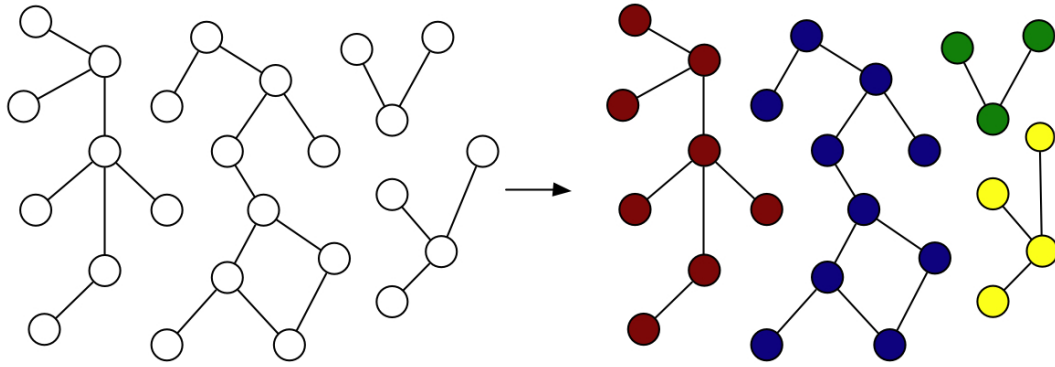


Figure 8.4: Identifying Strongly Connected Components

be costly however, if all the strongly connected components within a graph-space must be found, as this requires all remaining nodes within the graph-space to be used as the root for the algorithm.

Given that the disambiguation technique adopted in this chapter is able to use seed data, which is modelled as a social graph, the root node is specified as the *social graph root node*. Web resources which have common literals and resources with the social graph form the basis for the strongly connected component within the graph-space that is to be considered for traversal. Moreover, by simply removing the other strongly connected components, information is removed that is irrelevant to the person (due to the lack of transitive links to the social graph). For instance, the social graph contains seed data describing literals and resources which web resources should also contain if they are *identity web references*. Even if certain web resources do not exhibit any of those features, they may share features with other web resources which in turn share features with the social graph (thereby providing transitive linking). Therefore, it is very likely that resources, which form a separate island from the island containing the social graph, refer to another person.

8.4.3 Traversing the Graph Space

Once the strongly connected component has been found containing the *social graph root node* the graph-space can be traversed using Random Walks [Lov93]. The walker moves from a given node to a random adjacent node, and then randomly to any adjacent node of that node, thus forming a stochastic transition process. Edges must first be weighted within the graph-space to effect the transition probabilities of the random walker as follows:

8.4.3.1 Weighting Edges

Weighting edges within a graph-space is discussed in work by [IXZ07] stipulating that edge weights should form a probability distribution, however no emphasis is placed on the importance of certain edges over others. For instance, the goal of traversing a graph-space and measuring distances is to group similar web resources together. Therefore edges which denote unique links and relations between resources should carry more weight. This is reflected in the weighting function used by [Mal05] which places emphasis on edges between entities which are rare and are therefore more likely to denote uniqueness and also work by

[SFYD04] which states that the more important the relation between two nodes then the greater the weight should be.

A similar strategy is implemented in this technique by weighting edges according to their labels. This is where the semantics of information plays an important role, as the edges within the graph-space represent concepts from web accessible ontologies. These ontologies, when constructed using OWL (§3.2.4.2), allow characteristics to be specified for those concepts. For instance, the RDF models representing the social graph and each of the web resources to be disambiguated use concepts from the FOAF ontology, and given that FOAF is constructed from OWL, several properties contain characteristics which denote uniqueness. Therefore edges that are typed as *owl:inverseFunctionalProperty* or *owl:functionalProperty* - such as edges which refer to the *foaf:mailbox*, *foaf:homepage*, and *foaf:weblog* properties - are weighted *more* than other edges. It is worth noting however that gleaned models from web pages may contain ontological concepts from ontologies which are at present unknown. Chapter 6 attempts to control this by converting the concepts to the known vocabulary, enabling consistent semantics with the supplied seed data. However the conversion may not be possible if the ontology is not mapped. The metadata models are still included and the edges are weighted accordingly by looking up the defined concepts within the respective ontologies - given that this can be automated as the concept definition is machine-processable.

Referring back to the definition of the graph earlier within the chapter: the set of edges within the graph is denoted by E and the set of vertexes/nodes is denoted by V . The set of edge labels is denoted by $L = \{l_1, l_2, \dots, l_k\}$ where l_i denotes a label corresponding to a distinct concept from an ontology (e.g., *foaf:name*). Therefore a weighting function is defined that returns the weight associated with a given edge (and therefore a distinct ontology concept) $w(l_k)$. The weights are distributed using the same strategy from [IXZ07] as follows:

$$\sum_{l_i \in L} w(l_i) = 1 \quad (8.1)$$

The set of edges labels L can be divided into two subsets, edges which are unique L^U and edges which are not L^N such that $\{L^U \cup L^N\} = L$. The set of unique edges is composed of ontology concepts that are typed as *owl:inverseFunctionalProperty* such as associating a person with an email address, or the relation that defines a product as having a given serial number. By dividing the edges up into these subsets of L , the distribution of edge weights to be written as:

$$\sum_{l_j \in L^U} w(l_j) + \sum_{l_k \in L^N} w(l_k) = 1 \quad (8.2)$$

To date no work within the state of the art has explored heuristics or methods for setting the edge weights within a given graph-space and exploring their effects on the Random Walks process. As mentioned previously [SFYD04] states that edges should be weighted more which carry a unique property to encourage traversals. Each of the unique edges carry the same weight, regardless of their label, the same is also true for the normal edges. Let λ^U denote the weight for a unique edge and λ^N denote the weight for a non-unique edge, therefore Equation (8.2) can be rewritten as follows:

$$\lambda^U \cdot |L^U| + \lambda^N \cdot |L^N| = 1 \quad (8.3)$$

The difference between the weights for unique edges (λ^U) and non-unique edges (λ^N) can be derived as a proportion such that the varying of this proportion will effect the random walks over a given graph:

$$\eta = \frac{\lambda^U}{\lambda^N} \quad (8.4)$$

Using basic algebra λ^U can be derived from (8.4) as $\lambda^U = \eta \cdot \lambda^N$ and substituted into (8.3) to produce:

$$\lambda^N(\eta \cdot |L^U| + |L^N|) = 1 \quad (8.5)$$

By interlinking a graph-space from metadata models it is possible to refer to the machine-readable descriptions of the concepts from their respective ontologies. The characteristic of each concept denotes its membership in either of the two subsets of L : as being either unique (L^U) or non-unique (L^N). The variance of the value of η alters the proportional difference of weights between the labels of the two respective sets. This therefore permits the exploration of different weighting heuristics and their effects on the Random Walks process, the results of which are shown within chapter 10 when evaluating this technique.

8.4.3.2 Random Walks

The graph-space must be traversed in order to discover available paths from which distances between nodes can be measured. Random walks are performed through the graph-space by starting at a given node and then moving to any adjacent node, repeating this process along all possible transition paths. In order to perform this process several parameters must be set for the graph-space. First an adjacency matrix A is derived to compute the local similarity of adjacent nodes in the graph-space as described in [SFYD04]. The adjacency matrix stores the local similarity of node i to each adjacent node j (A_{ij}), thus forming a local similarity matrix which is populated as follows:

$$A_{ij} = \begin{cases} \frac{w(l(e_{ij}))}{|(e_{i.}) \in E : l(e_{i.}) = l(e_{ij})|}, & (i, j) \in E \\ 0, & otherwise \end{cases} \quad (8.6)$$

The above equation functions by distributing edge weights uniformly to edges leaving a given node (i) which are of the same type obtained by returning the label (l) for a given edge $e_{ij} \in E$ should one existing between nodes i and j . The intuition is that certain nodes may act as transitional hubs given that many nodes of the same type are leaving them, random walks should therefore be influenced by traversals along paths through which nodes containing few edges are available, this will in turn enable clustering to identify the discriminatory features within the graph-space which are unique to a given cluster. The diagonal degree matrix is computed using:

$$D_{ii} = \sum_{k=1}^{|A|} a_{ik} \quad (8.7)$$

A random walk through the graph is a stochastic process and is defined according to a first-order Markov chain [MT93]: future states within a model do not depend on past states

- this is the same assumption which is used within chapter 6 when applying Hidden Markov Models to derive state sequences from an input sequence of HTML tokens. Therefore the probability of moving to node j given that the random walker is currently in node i and t steps have been traversed is completely independent of past traversals that the walker has made through the graph-space, prior to reaching i . Using the Markov chain, the transition matrix is defined using $P = D^{-1}A$ for one step, and $P^{(t)} = (D^{-1}A)^t$ for t steps. Consulting a transition matrix, for example for 1 step (P), returns the probability of *teleporting* from one node to another in the graph, and for 3 steps (P^3) moving from one node to another after 3 steps.

8.4.4 Measuring distances within the Graph Space

8.4.4.1 Commute Time

Random walks produces a transition probability matrix P which when consulted returns the probability p_{ij}^t of moving from node i to node j given that t steps have been traversed. These transition probabilities can be used to calculate the distances between nodes within the graph-space, from which similar nodes may be clustered together. One example distance measure is Average First-Passage Time [YVW⁺05], which calculates the number of steps taken when performing a random walk through the graph-space from one node to another, in essence this returns the *hitting time* for the traversal:

$$m(k|i) = \begin{cases} 1 + \sum_{\substack{j=1 \\ j \neq k}}^{|V|} p_{ij} m(k|j), & i \neq k \\ 0, & otherwise \end{cases} \quad (8.8)$$

This formula recurses through the graph-space using the transition probabilities populated within P to cumulatively derive the time taken until the expected node k is reached for the first time. However, this only returns the Average First-Passage Time taken from leaving a given node and reaching another. Therefore by calculating the Commute Time it is possible to derive the round trip cost of leaving a given node, reaching another node and then returning again to the previous node:

$$n(k|i) = m(i|k) + m(k|i) \quad (8.9)$$

Both $m(i|k)$ and $m(k|i)$ will return different values (given that P is not symmetric in the context of this work), therefore by summing their traversal costs a symmetric matrix is produced for the Commute Time N where $n_{ij} = n_{ji}$. The number of steps t which are considered when measuring Commute Time distances effects the resolution of the walk [CS07]. Smaller values of t form clusters within the matrix given that probabilities are well defined for the walker to make transitions to nodes nearby, but are set to 0 for nodes which are far away, as the walker is unable to reach them. Conversely as $t \rightarrow \infty$ the transition probabilities moves to a stationary distribution (π) thereby making distinctions between nodes hard - this is a result of local information being lost.

Work by [QH07, YFD⁺09] has examined the behaviour of Commute Time and found that as the number of paths between nodes increases, where these paths are short in length, then the commute time decreases. Furthermore extensive work by [SJ02] has explored the

effects of t on partially labelled graphs and clustering using random walks. Such work found that smaller values of t are better suited to graphs where class membership alters over short distances. This is related to the setting of the compiled graph-space within this technique. For instance it is common for a web resource which is not an *identity web reference* to be connected to the social graph root node by the name of the individual whose *identity web references* are to be found.

The makeup of the compiled graph-space is such that the shortest number of steps which can be made from one *web resource root node* to another *web resource root node* is 4: e.g., *web_resource* $\rightarrow$ *person_node* $\rightarrow$ *name* $\rightarrow$ *person_node* $\rightarrow$ *web_resource* (refer back to Figure 8.4 for an example). As such the literals and resources which form the central node in this transition path act as hubs within the graph-space and take 2 steps to reach - this is a similar assumption to that used by [HZS06]. Based on this setting the number of steps taken by the random walker should be set to consider traversals between nodes which are *web resource root nodes* and literals and resources from triples. Therefore in order to apply commute time effectively as a distance measure and focus on short paths within the graph-space, and maintain local information, t is set to 2.

Using the Commute Time within the graph-space allows a clustering technique to then be applied in order to group the web resources which are most similar to the social graph, and therefore take the least commute time to reach. In essence this step can be regarded as a *Commute Time Kernel* [YFD⁺07] by reducing the density of the graph-space to only the *resource root nodes* and the *social graph root node*. Figure 8.5 shows the graph space reduced from Figure 8.3.

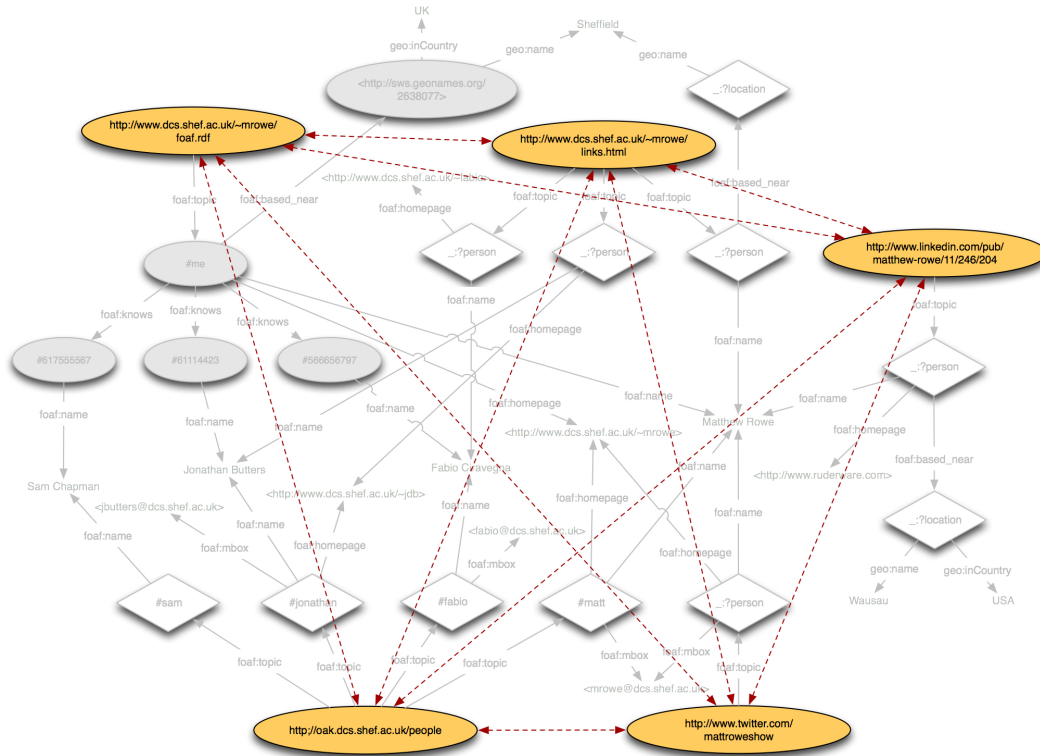


Figure 8.5: A transformed graph-space connecting nodes by Commute Time Measures

8.4.4.2 Optimum Transitions

The variation in the number of steps that the random walker takes (denoted by t) can be exploited to measure the transition probabilities. In [Mal05] walks are made through the graph-space until a desired node is reached or when t exceeds 50. In [CS07] the size of t is iteratively increased in order to investigate the effects on an image retrieval model. As mentioned previously the t parameter has interesting properties when its size is altered. It is possible to measure distances between nodes within the graph-space by deriving the optimum number of steps needed to traverse from a given node to another node. The optimum number of steps between two nodes is found by incrementing t until the transition probability has peaked for the first time. This involves iteratively increasing the matrix power of the transition probability matrix P . When traversing from node i to node j , the optimum number of transitions (and therefore the value of t) is calculated using Algorithm 6.

Algorithm 6 optTrans(P) : Derives the Optimum Transitions matrix O from incrementing the matrix power of P

Input: P

Output: O

```

1:  $O_{|P|} = 0$  {Initialise the Optimum Transitions Matrix as the zero matrix}
2: for  $i = 0$  to  $|P|$  do
3:   for  $j = 0$  to  $|P|$  do
4:     for  $t = 1$  to  $\infty$  do
5:       if  $(p_{ij}^t \geq p_{ij}^{t-1})$  and  $(i \neq j)$  then
6:          $o_{ij} = t$ 
7:       else
8:         break
9:       end if
10:    end for
11:  end for
12: end for
13: return  $O$ 

```

8.4.5 Clustering Web Resources

Using the above distance measures web resources must be clustered into two distinct groups: *positive* and *negative*. Agglomerative clustering is used [WGLD05] to produce the two clusters. This clustering technique functions using a *bottom-up* strategy by placing each element to be clustered in an individual cluster - therefore if there are n elements there will be n singleton clusters. Clusters are then compared with one another and if their similarity - computed by comparing and summing the individual similarities of each respective element within the clusters and normalising this value by the number of elements - is above a given threshold then the clusters are merged. The process is repeated until no more clusters can be merged. Agglomerative clustering is defined formally as:

$$average_dist(c_1, c_2) = \frac{\sum_{r_i \in c_1} \sum_{r_j \in c_2} dist(r_i, r_j)}{|c_1||c_2|} \quad (8.10)$$

Where $dist(r_i, r_j)$ is defined for Commute Time Distance (using the Commute Time

distance matrix N where N_{ij} is the Commute Time distance between node i and node j) as:

$$dist(r_i, r_j) = \frac{n_{r_i r_j}}{max(N)} \quad (8.11)$$

And for Optimum Transitions (using the Optimum Transitions matrix O where O_{ij} is the optimum number of transitions between i and j) as:

$$dist(r_i, r_j) = \frac{o_{r_i r_j}}{max(O)} \quad (8.12)$$

For two clusters to be merged together $average_dist(c_1, c_2)$ must be less than a given threshold σ where $0 < \sigma < 1$. The value of the threshold is proportional to the maximum Commute Time from N , when clustering according to Commute Time distances, or proportional to the maximum number of transitions from O , when clustering using Optimum Transitions. This clustering strategy assumes that the strongly connected component will contain incorrect references, which only share a low number of features in common with the social graph. Therefore clustering based on the proportional distance will identify the maximum path range incorporating incorrect references (this is essentially the maximal diameter of the graph-space). However, this approach is limited if the web resources within the strongly connected component are all in fact positive references. This clustering strategy therefore dictates that a value for σ must be derived by tuning on known data (this is discussed in the experimental setup of the evaluation in §10.4.2).

Using agglomerative clustering via distance measures within the graph-space omits web resources from the cluster containing the social graph which take a long time to reach. When using the Random Walks model, the walker becomes enclosed in a given clique within the graph-space. As such, web resources which belong to that clique are less distance from the social graph, whereas web resources which are outside of this clique take longer to reach - given that they shared a limited number of features with the seed data. The underlying theory behind clustering based on such distances is that the resultant *positive* cluster will contain elements which have a high similarity with one another and a low similarity with elements in the *negative* cluster.

8.5 Summary

This chapter has discussed the application of a graph-based disambiguation technique to the disambiguation task addressed within this thesis. The Random Walks technique is presented as a suitable method based on the correlation between current state of the art applications and knowledge models using Semantic Web technologies. The Random Walks model forms a stochastic transition process allowing two distance measures: Commute Time and Optimum Transitions to be used to measure node proximities. Agglomerative clustering is then used to group web resources within the graph-space with the social graph based on those proximity measures. The compilation of a graph-space is well suited to the representation of the seed data and web resources as RDF models. Furthermore one of the notable contributions of this technique is an edge weighting function based on the division of edge labels into denoting either unique or non-unique edges. This allows unique concepts from the metadata models

of the web resources to be weighted based on their ontological properties.

Chapter 10 compares the use of Random Walks against the disambiguation techniques discussed in the previous chapter and the following chapter through a detailed evaluation using real world data. The requirements listed in this chapter that a graph-based disambiguation technique must fulfill are discussed in chapter 11 with respect to the results of the evaluation in chapter 10. The following chapter explores the use of semi-supervised machine learning as an approach to disambiguation.

Chapter 9

Semi-supervised Machine Learning for Disambiguation

9.1 Introduction

In real world disambiguation and information classification tasks there is a common scenario where only a limited amount of labelled data is provided - given that labelled data is expensive to produce - whereas there is a large quantity of unlabelled data available. *Supervised* methods - such as the rule-based disambiguation technique presented in chapter 7 - require labelled data to support the decision process. Should the amount of labelled data be limited then the performance of the technique can be restricted, given the lack of coverage and ability to adapt to unlabelled data. Therefore there is a clear motivation to somehow incorporate unlabelled data into the decision process to improve the accuracy and coverage over the unlabelled data. In the context of disambiguating *identity web references* this allows features which are not present within the seed data to be learnt in order to detect *identity web references* which were previously not found, thereby enhancing accuracy. The use of Random Walks in chapter 8 uses the provided seed data to label nodes within a compiled graph-space, this prior information is then used to group similar web resources by their common features - given the inductive process by which Random Walks functions.

Machine learning techniques are ideally suited to this problem setting: *unsupervised* techniques learn from only unlabelled data by constructing a model which best fits the features of the data. *Supervised* techniques use labelled data to construct a classification model which is then applied to unlabelled data. *Semi-supervised learning* is the middle ground between these approaches by initially using labelled data to construct a classifier, and then using unlabelled data to improve or enhance the classifier. This form of machine learning has various different learning strategies, each differing in their method of incorporating unlabelled data into the learning process. This chapter discusses example state of the art semi-supervised machine learning strategies and classification models which can be trained for disambiguation. A technique is presented within this chapter which employs a semi-supervised learning strategy known as *Self-training* as a wrapper algorithm to train different machine learning classifiers. An explanation is presented of how machine learning instances can be constructed from RDF models and how features can be extracted from

such models. A novel variation of feature similarity measures when constructing machine learning instances is explained, and the effects of this permutation are discussed during the evaluation of the technique within chapter 10.

This chapter is structured as follows: section 9.2 reviews machine learning techniques, explaining various learning strategies and classifiers. Section 9.3 details the requirements which a semi-supervised machine learning technique must fulfill. Section 9.4 presents the explored Self-training strategy to be used for disambiguating *identity web references*; detailing the generation of seed data (to accompany the social graph from reconciling digital identity fragments), the extraction of features from web resources and the variation of feature similarity measure and the algorithm used by the learning strategy. Section 9.5 concludes the chapter with a summary of the discussed work.

9.2 Related Work

Semi-supervised machine learning offers a means through which an initial classification model can be learnt using labelled data, and enhanced through the use of unlabelled data. Different learning strategies exist within this form of machine learning, three examples of which will now be discussed. The term classifier will be used to refer to a classification decision model in this first subsection, before moving on to give concrete examples and explanations of three widely used classifiers.

9.2.1 Semi-Supervised Learning Strategies

9.2.1.1 Self-training

The Self-training learning strategy [CSZ06] uses labelled data to construct an initial hypothesis. This hypothesis is then applied to unlabelled data to derive class labels. Instances which exhibit the strongest classification confidences are then added to the seed data and the classifier is retrained and then reapplied to the unlabelled data. This process repeats until a stopping condition is reached. The approach presented in [YHcC04] employs a Self-training strategy for learning a web page classifier. The classifier is then used to label web pages as being either *positive* or *negative*. The approach is divided into two distinct stages: *mapping* and *convergence*. The mapping stage generates negative training data to be used alongside the positive data in order to learn the classifier: negative examples are generated by extracting examples from the unlabelled set that do not contain any positive features (given that the positive set is already known). The convergence stage then uses both the negative and positive training sets to iteratively learn a classifier, whereby after each iteration the negative and positive training sets are enlarged with the strongest labelled instances. The classifier is then retrained on the training sets and the process is repeated until all unlabelled instances have been classified: this acts as a *stopping condition* for the learning process.

The Self-training approach described in [WYZ07] learns to extract relations from Wikipedia text using only initial positives examples as input. In order to discriminate between relations that should be extracted and relations that should not be extracted negative training examples are generated - thereby providing a binary classification problem. Negative examples are generated by building a Rocchio classifier [BSA94] using the positive examples and applying this to the complete set of unlabelled relations, the negative examples are chosen

based on their similarity with the vectors built by the classification model. Self-training is then used to iteratively learn and improve a classifier using classified instances with the strongest classification confidences.

A similar Self-training strategy to [WYZ07] is described in [FL06] in order to classify texts using only positive seed data initially. In [FL06] a binary classification problem is investigated thereby requiring the generation of negative seed data to complement the positive seed data. Positive seed data is assumed to be sufficiently large and therefore a statistical distribution of the features within the data is compiled. Negative instances are extracted from the unlabelled instance set which fall below the distribution mean. The intuition behind this strategy is that strong negative instances will be generated as a result. Using a similar learning strategy to [YHcC04] and [WYZ07] the positive and negative training sets are used to learn a classifier which is then iteratively improved through retraining by enlarging the training sets.

Self-training has been used within the domain of object detection within images in work by [RHS05]. A classifier is trained on a set of labelled images (either containing an object or cluster), and then applied to a set of unlabelled images, iteratively enlarging the respective training sets with the images exhibiting the strongest classification scores. The same strategy has been used by [Mih04] in order to perform word sense disambiguation by constructing an initial classifier using only a limited set of known word senses and gradually retraining the classifier.

The Yarowsky Algorithm One of the most commonly used algorithms within the domain of word sense disambiguation is the Yarowsky algorithm [Yar95] - first mentioned within this thesis in chapter 4. The algorithm functions in a similar manner to the standard Self-training approaches discussed thus far. An analysis of this algorithm by [Abn04] explains how the algorithm is composed of two loops: an *inner loop*, which is a supervised learning algorithm such as a classification model which takes as input some labelled and unlabelled data and predicts class labels; and an *outer loop* wrapping the classifier by rebuilding the labelled training data with suitable examples.

Algorithm 7 presents the Yarowsky algorithm in more detail. The input is the set of examples and some initial labelling, designated by X and Y^0 respectively. For a set number of iterations (the *outer loop*) a classifier is trained using the labelled examples (line 2), the classifier is then applied to the example set and attempts to assign labels to each example (the *inner loop*): choosing the strongest label based on the confidence of the classifier. Line 5 shows how new class labels are assigned to each example (thus rebuilding the training set): if the example was supplied to the algorithm as a labelled instance ($x \in \Lambda$) then the label remains the same, alternatively if the example was not supplied as labelled example and the classification confidence of the classifier is above a given threshold (ζ) then the example is given the classifiers decided label ($\hat{x}$), otherwise the example is not labelled. This process repeats until the parameters of (Ψ) converge, and the labels from the previous iteration are the same as the current iteration.

A key difference between the Yarowsky algorithm and the Self-training approaches discussed previously is in the maintenance of the labelled data sets following each iteration. The Yarowsky algorithm allows the labelled datasets to be altered following each iteration of the *outer loop* thereby altering previously made disambiguation decisions. Conversely

Algorithm 7 yarowsky(X, Y^0) : Takes as input a set of examples (X) and an initial labelling (Y^0). A classifier is trained and applied to the examples repeatedly until its parameters converge. The algorithm outputs a final labelling (Y).

Input: X and Y^0

Output: Y

```

1: for  $t \in \{0, 1, \dots\}$  do
2:   Train  $\Psi$  using labelled examples  $(\Lambda^t, Y^t)$ , where  $\Lambda^t = \{x \in X | Y^t \neq \perp\}$ 
3:   for each  $x \in X$  do
4:     Set  $\hat{y} = \arg \max_j \pi_x^{t+1}(j)$ 
5:     Set  $Y_x^{(t+1)} = \begin{cases} Y_x^0 & \text{if } x \in \Lambda^0 \\ \hat{y} & \text{if } \pi_x^{(t+1)}(\hat{y}) > \zeta \\ \perp & \text{otherwise} \end{cases}$ 
6:   end for
7:   if  $Y^{(t+1)} = Y^t$  then
8:     return  $Y^t$ 
9:   end if
10: end for

```

most Self-training approaches maintain a fixed collection of labelled data, where past disambiguation decisions are not corrected - thus ensuring that the labelled data is not altered, simply added to.

9.2.1.2 Expectation-Maximisation

The Expectation-Maximization (EM) algorithm [DLR77] is an unsupervised algorithm which attempts to build a maximum likelihood model over some data distribution, it can however be adapted as a semi-supervised learning strategy. A basic definition of EM in a semi-supervised setting is as follows: beginning with labelled data a probabilistic classification model is created using this data and the internal parameters of the model are calculated. The model is then applied to unlabelled data in order to predict the expected labels of the data (Expectation step). The predicted labels are assumed to be correct and the parameters of the model are retuned accordingly (Maximisation step). The expectation and maximization steps are repeated until the parameters of the model converge [Mit97] on a stable state.

EM is often applied to train generative classification models which predict the probability of a given instance belonging to a given class - where such models attempt to build an underlying probability distribution of the data. The Expectation step predicts class labels and the Maximization step computes the maximum likelihood estimates of class predictions using only a partially observed portion of the entire dataset (the labelled subset). An example application of EM appears in text classification [TT03] where EM is used to train a Naive Bayes classifier. One of the limitations of EM is that it is not suited for semi-supervised learning of discriminative classification models - as it relies on tuning the likelihood estimates for class membership. Another limitation of EM is its behaviour in that it is a "*hill-climbing*" algorithm that tends to converge on a local maxima which may not be the optimal hypothesis. That said, several pieces of work have provided solutions to overcoming the local maxima problem such as running the algorithm many times and choosing the best convergent as the final result [SB08], or running the algorithm for a set number of steps and choosing the best thread which is then continued until convergence.

9.2.1.3 Co-training

Co-training is a learning strategy which begins by training two classifiers simultaneously on labelled data. Each classifier has a distinct view, or perspective, of the data [BM98] i.e., considers different features, or uses different classification models. The classifiers are then applied to unlabelled data, the strongest predicted classifications derived by one classifier are then added to the training data of the other classifier and vice versa. The resultant strategy therefore functions by using the classifiers to train one another. Co-training can be regarded as a form of Self-training given that the strongest classifications are added to the classifiers, and the labelled data sets are not altered, merely added to. One of the intrinsic requirements is the variation in the perspectives used by each classifier. This is the use of different feature sets when observing data, for instance when observing a web page one classifier may consider named entities to be features, while the other may consider only hyperlinks. Co-training has been used in a number of domains including email classification [KM01], reference reconciliation [MRS02] and text classification [DLGT03].

The approaches discussed thus far have described how a classifier can be learnt from seed data and subsequently improved through gradual training using previous classification decisions. Several limitations of such approaches are evident. If only positive seed data is available, and the classification task is addressed as a binary one, then negative seed data must be approximated somehow in order to weight the bias of the classification model.

The problem settings of the discussed approaches are similar to the problem setting faced when disambiguating *identity web references*. For instance, the labelled data supplied following the reconciliation of digital identity fragments is the only known positive training data - given that it defines the person whose information is to be found. If a binary classification task is to be considered as a mode of disambiguating *identity web references* then it is essential that negative seed data is also compiled. Without this seed data, the learnt classification hypothesis would be biased towards web resources which cite a person. Instead this hypothesis would have to be balanced such that the model learns to discriminate a web resource which cites a person from one that does not, this is an essential property of the disambiguation process.

9.2.2 Classifiers

The presented learning strategies function by training a given machine learning classifier (Ψ) using both labelled and unlabelled data. In essence such strategies act as wrappers for classifiers by not altering the internal functionality of the algorithms used to build a classification model, but instead provide different inputs to learn from and using the outputs from the classifiers to rebuild the training sets. This section describes three classifiers from the literature and alternative models which utilise proximity based methods for classification.

9.2.2.1 Perceptron

The Perceptron classifier is a basic form of an Artificial Neural Network [Bis95]. The classifier takes as input a given instance x represented as a binary feature vector - such that element x_i corresponds to the i th element in the vector of x - and applies a weight vector w and a bias b to derive a given real numbered outcome. The elements within the weight vector

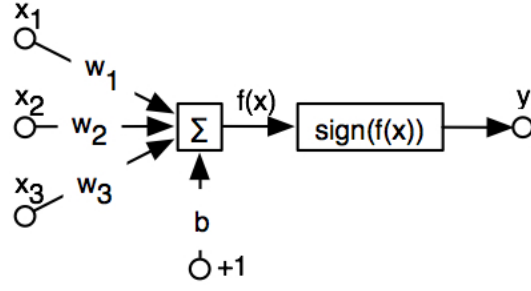


Figure 9.1: Perceptron Architecture for an input vector x with 3 features

w contains weights for the corresponding elements within the input vector x . This can be written formally as follows:

$$f(x) = w \cdot x + b = 0 \quad (9.1)$$

The value of $f(x)$ returns a real number ($\mathbb{R}$) which must be assessed for class membership, given that a single Perceptron forms a binary classifier. Therefore a hardlimiter or sign/step function is defined. This sets a given threshold which when exceeded sets the output to a binary outcome $y \in \{0, 1\}$. For instance in terms of the above function this could be defined as follows:

$$y = \text{sign}(f(x)) = \begin{cases} 1 & f(x) > 0 \\ 0 & \text{otherwise} \end{cases} \quad (9.2)$$

This is where the similarity between the Perceptron and a neuron within the brain becomes evident. If the derived outcome $f(x)$ is above a desired threshold then the neuron fires, otherwise the neuron does not. In terms of information classification Perceptrons function by being trained to weight features of the input vector, apply a bias and then determine the outcome. In this sense a Perceptron is often applied in a binary classification setting, however multiple Perceptrons can be used together for multi-class classification problems [Bis95].

To function effectively the Perceptron must be trained for a given class decision. For this the Perceptron Learning Rule is used [Gal90]. The Perceptron is assumed to be blank and therefore have its parameters set to zero: weights and bias. The classifier is given a set of inputs and their target outputs (where the outputs belong to $y \in \{0, 1\}$). The learning process adjusts the weights and bias such that the decision boundary classifies the inputs according to their targets. Weights and the bias are adjusted using the following equations:

$$w' = w + (y - \hat{y}) \cdot x \quad (9.3)$$

$$b' = b + (y - \hat{y}) \quad (9.4)$$

The learning process is repeated by passing each labelled input vector into the Perceptron and checking the outcome against the desired outcome. The weight vector and the bias are repeatedly adjusted by testing one input at a time therefore using an *on-line* learning

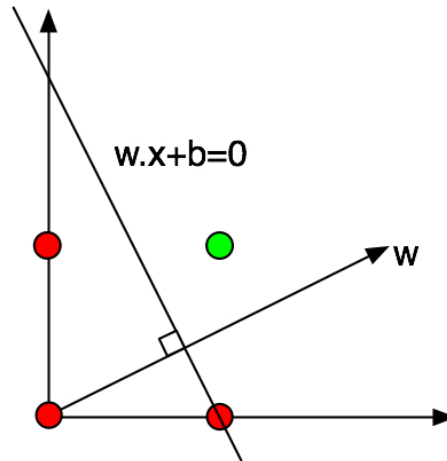


Figure 9.2: Learnt Perceptron with a separating hyperplane which is orthogonal to the weights vector

strategy. The weight vector converges when all outcomes from each of the inputs matches their target outcome, at this stage learning is stopped and the weight vector and bias are used to calculate the decision boundary within the feature space. There are multiple solutions to learning the weight vector and bias, given that a single hyperplane is learnt within the feature space rather than the optimal one.

The weight vector runs orthogonal to the decision boundary within a vector space and the bias shifts the decision boundary away from the origin within the space allowing its interception points of the axis to be calculated. If the inputs only contain two features then the decision boundary can be represented graphically as shown in Figure 9.2 where a Perceptron is learnt for an *AND* logic gate using the step function from Equation (9.2).

Several modifications of this classifiers exist, most notably Average Perceptron, which constructs several decision boundaries within the feature space and computes the average hypothesis from those, and Voted Perceptron, which constructs several decision boundaries and chooses the most agreed on hypothesis. The use of Perceptrons forms the basics for classification of linearly separable data, however this is where their limitation lies. If points within the feature space are not linearly separable then a distinction cannot be made for class membership and the Perceptron will misclassify instances. The learning of weights within a Perceptron functions by learning patterns for classification through observations of the common features within the input vectors. This form of classification forms the basis for discriminative classifiers which segment the feature space and therefore partition data points. Another note about the description used within this section is that the example Perceptron has only considered features within two dimensions - to aid with visualising the technique - however additional dimensions can easily be added as the model allows for this. When extra dimensions are included the decision boundary alters from a hyperplane into a discriminative surface which segments the data points.

9.2.2.2 Support Vector Machines

Advancing on the principle of linear classification through a discriminative hyperplane is the use of Support Vector Machines (SVMs). The same intuition is used as with classification

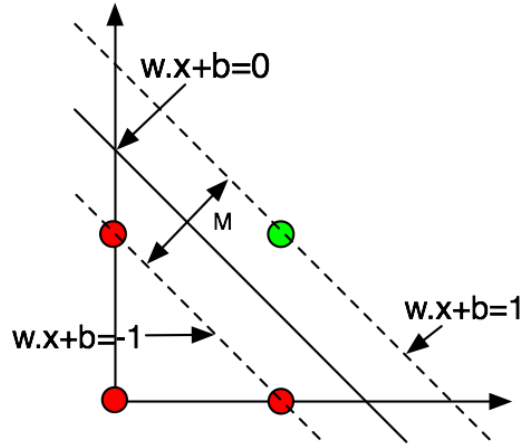


Figure 9.3: Learnt separator hyperplane with two supporting hyperplanes where the margin is maximised

using Perceptrons such that a decision boundary is learnt which segments the vector space and its data points. A Support Vector Machine differs however by learning supporting hyperplanes which run parallel to the separator hyperplane (decision boundary), such that the margins between the separator plane and the supporting hyperplanes are maximised [DPHS98]. This principle is best illustrated through a visual example. Consider Figure 9.2, the hyperplane which is learnt by the Perceptron, although sufficient to classify the data points is not the best hypothesis. Instead, the best hypothesis is the one where the distance between the closest vectors from distinct classes is maximised.

Figure 9.3 shows a more appropriate decision boundary which would classify the inputs. Recall the definition of the discriminative hyperplane from the Perceptron classifier in (9.1). In Support Vector Machines the margin between the data points is maximised and thus divided into two class outcomes:

$$w.x + b \geq 1 \quad \text{for } y = 1 \quad (9.5)$$

$$w.x + b \leq -1 \quad \text{for } y = 0 \quad (9.6)$$

This allows the supporting hyperplanes as shown within Figure 9.3 to be derived as planes which run in parallel to the separator hyperplane. The points which are closest to the hyperplane from each representative class $y \in \{0, 1\}$ are called the *support vectors*. Support Vector Machines are referred to as such because they use these vectors to learn an optimum hyperplane which separates the inputs within the vector space.:

$$w.x + b = 0 \quad (9.7)$$

In order to learn an optimum hyperplane the margin, represented as M within Figure 9.3, must be maximised. The margin can be represented using geometry as $M = \frac{2}{\|w\|}$ where its maximization is a case of minimizing $\|w\|$. This can be rewritten as follows:

$$\Phi(w) = \frac{1}{2}(w.w) \quad \text{where } y_i[(w.x_i + b \geq 1] \quad \forall i \quad (9.8)$$

The problem is now reduced to a constraint optimization problem, such that the value of $\|w\|$ must be minimized whilst satisfying the above constraints. The constraint ensures that $w \cdot x_i + b = 1$ if the input vector x_i is a support vector and > 1 if it is not a support vector. Note how the outcome of y_i alters the polarity of the value such that it is always a positive real number ($+\mathbb{R}$). Support Vector Machines solve the constraint optimization problem by using Lagrange Multipliers which allows quadratic programming to be used. For a full discussion and explanation of this principle and proof of the theorem, the reader should refer to [Bur98]. Lagrange multiples are used ($\alpha_i, i = 1, 2, \dots, n$) to correspond to each of the inequality constraints from Equation (9.8). Derivation of the multipliers finds that for a given input x_i , the multiplier is greater than 0 if the input is a support vector, and equal to 0 otherwise. The role of support vectors is such that all other vectors within the input space can be ignored, the optimum hypothesis only depends on the support vectors and will therefore not alter if vectors are removed which are not support vectors. Going back to the previous function $f(x)$, this can now be rewritten as follows (where l denotes the set of all labelled vectors):

$$f(x) = \sum_{i=1}^l y_i \alpha_i (x \cdot x_i) + b \quad (9.9)$$

Support Vector Machines, as discussed so far, work well for data which is linearly separable as in the case of the example in Figure 9.3. However there are cases where the data is nonlinearly separable. In such cases a *kernel trick* is used to alter the dynamics of the feature space so that a linear classifier can be applied [Sch98]. This transformation is one of the powerful aspects of SVMs and allows features within inputs to be considered as of greater importance than others thereby providing prioritisation with regards to specific conditions or aspects.

One of the advantages of SVMs is their ability to generalise to new inputs. By choosing the optimal hyperplane the decision boundary is ensured to divide the feature space at the optimal part. This provides one of the main advantages of SVMs over Perceptrons. In the case of Perceptrons the learnt hyperplane is not necessarily optimized to be the best fit for separating the provided input data, given that this is the first hypothesis that is converged on following training. As Figures 9.2 and 9.3 have shown a more intuitive hyperplane can be learnt through maximisation of the margins with respect to support vectors. SVMs learn a decision function which does not overfit the training data, thereby providing a *robust* learning process which in turn classifies new instances with high accuracy. [HGZ⁺04] has applied SVMs in the context of author disambiguation by training an individual classifier for each author and then testing multiple models on a global citation set. Citations were then attributed to authors based on the classifier of the author which maximised the classification confidence (i.e., achieved the highest value for $f(x)$).

Support Vector Machines have been adapted for single-class classification scenarios in [SPST⁺99], and applied within the domain of document classification in [MY02]. The approach described in [SPST⁺99] treats the collection of single-class labelled instances (X) as the known subset in a given feature space (S), the origin is then treated as the counter-class ($\hat{S}$). The known dataset is then separated from the origin, thus inducing a classifier which maximises distances between the two sets. Single-class SVMs were tested in [MY02] which found them to be sensitive to the choice of kernel and feature weighting approach, rendering

the generalisation of parameters to be limited and requiring domain-specific tuning.

9.2.2.3 Naive Bayes

A Naive Bayes Classifier constructs a generative probabilistic model for classification. Generative models differ from discriminative models by building a density model over the labelled data, in terms of Naive Bayes this model acts a probabilistic model over the labelled data. An instance to be classified, such as x , can be expressed as a set of features $x_i \in x$ present in the instance. Classification attempts to assign the most probably class to the instance - i.e., $v \in \{0, 1, \dots, n\}$ - based on the presence of features within the instance. Using conditional probability therefore this label assignment can be written as:

$$V_{NB} = \arg \max_{v_j \in V} P(v_j|x) \quad (9.10)$$

$$V_{NB} = \arg \max_{v_j \in V} P(v_j|x_1, x_2, \dots, x_n) \quad (9.11)$$

Therefore a Naive Bayes classifier attempts to choose the most likely class label given the features present within an instance. Using Bayes theorem this can be rewritten as follows:

$$V_{NB} = \arg \max_{v_j \in V} \frac{P(x_1, x_2, \dots, x_n|v_j)P(v_j)}{P(x_1, x_2, \dots, x_n)} \quad (9.12)$$

$$V_{NB} = \arg \max_{v_j \in V} P(x_1, x_2, \dots, x_n|v_j)P(v_j) \quad (9.13)$$

The denominator in Equation (9.12) is dropped as it acts as a constant which is independent of the probably of the class label [Mit97]. At this stage if one is to classify a new instance x then the probabilities must be estimated for $P(x_1, x_2, \dots, x_n|v_j)$ and $P(v_j)$. Estimating the former is difficult if the training data is insufficient and does not cover the presence of the features given the class. Instead Naive Bayes uses the assumption of variable independence, thus allowing the probability of $P(x_1, x_2, \dots, x_n|v_j)$ to be derived from the product of the probability of each feature in X given the class v_j as follows:

$$V_{NB} = \arg \max_{v_j \in V} P(v_j) \prod_i P(x_i|v_j) \quad (9.14)$$

The probabilities are built from training data by using frequency distributions over the observations - in the same manner as the transition and emission probability matrices are built for Hidden Markov Models within chapter 6 of this thesis using maximum likelihood estimates. Assuming that there is a binary classification problem where the probability of a the example x belonging to a given class can be written as:

$$P(v|x) = P(v) \prod_i P(x_i|v) \quad (9.15)$$

By taking log probabilities Equation (9.15) can be rewritten as follows (where $\bar{v}$ is the compliment of v):

$$\log \frac{P(v|x)}{P(\bar{v}|x)} = \log \frac{P(v)}{P(\bar{v})} + \sum_i \log \frac{P(x_i|v)}{P(x_i|\bar{v})} \quad (9.16)$$

This presents the classifier as learning a linear classification model which separates two class decisions ($y = \{v, \bar{v}\}$), where if $f(x) > 0$ then $y = v$ otherwise $y = \bar{v}$. Several bodies of work have studied the properties and effects of Naive Bayes and have contrasted their performance against discriminative classification models, such as Perceptron and Support Vector Machines, and have shown their performance to be comparable. One of the interesting qualities of the Naive Bayes classifier is its use of the assumption of independence between variables, as at first glance this appears to be counter-intuitive i.e., given a text classification task, assuming words within a document are independent of one another contravenes natural language structure.

One of the advantages of Naive Bayes over discriminative classifiers is the generation of a hypothesis via a generative model, therefore not requiring the computation of multiple hypotheses. Due to the construction of a generative model, Naive Bayes classifier is often used in conjunction with Expectation Maximization as described in work by [NMTM00] to compliment its maximum likelihood nature when selecting a class label. The properties of Naive Bayes are explored extensively in [Ris01] which found Naive Bayes to be effective in situations when probability estimates were inaccurate yet the classifications were correct.

9.2.3 Proximity Measures

Classification is not limited to a binary assumption of class labels (e.g., positive vs negative) as discussed above. Although single-class SVMs go some way to avoiding the task of negative training data generation, there are other approaches based on the analysis of class distribution within a given vector space. At the heart of such methods is the assumption that the data is distributed using a statistical distribution, therefore any instances which do not follow the same distribution as known data are regarded as not belonging to the same class. At a low-level such analyses involve the conversion of labelled data for a single class into feature vector representations and plotted within a vector space model (similar to the previous Figures). Graphically this is akin to observing points which lie outside of the known distribution within the feature space, where if data points fall within the known labelled distribution they are labelled as being set members, if they fall outside of the distribution they are termed *outliers*.

This process of *outlier detection* [CBK09] has been applied in many domains where training data to counter a desired class outcome is hard to come by such as fraud detection, author verification, anomaly detection and network performance. Three example proximity measures are now described, each of which measure the distance from a given instance x to the mean of the sample distribution for a given class μ :

$$\mu = \frac{1}{|X_p|} \sum_{x_i \in X_p} x_i \quad (9.17)$$

9.2.3.1 Cosine-Angle Distance

One of the most basic and widely used proximity measures is the Cosine Angle distance which derives the similarity between two vectors by calculating the cosine of the angle between them. Using the centroid vector of a given class distribution from Equation (9.17) the maximal radius of the known class distribution can be calculated. Each unlabelled

example can then be plotted in the same feature space to derive the position with respect to the known distribution: either *inside* or *outside* of the class boundary. A spherical shape is therefore drawn surrounding the prototype vector for the class, thereby ignoring the shape of the distribution. The Cosine Angle distance is calculated using Equation (9.18), the output from the measure ranges between -1 and 1 , where the former indicates complete dissimilarity and the latter indicates complete similarity.

$$\text{cosine}(x, \mu) = \frac{x \cdot \mu}{|x| \cdot |\mu|} \quad (9.18)$$

9.2.3.2 Euclidean Distance

The second proximity measure that can be applied to detect statistical outliers is the Euclidean distance. Similar to Cosine-Angle distance, the Euclidean distance is calculated from the centroid of the class distribution to the further point in that class. Unlabelled data is then plotted in the same feature space and its position analysed with respect to the decision boundary. The Euclidean distance [QSGP04] between two points is the straight line between them (i.e., as the crow flies), it is calculated by finding the root of square of the differences between the features of the centroid vector and the example vector. It is similar to Cosine-Angle distance in the sense that it measures a spherical radius surrounding a given class centroid in order to plot the class decision boundary.

$$\text{euclidean}(\mu, x) = \sqrt{\sum_{x_i \in x} (x_i - \mu_i)^2} \quad (9.19)$$

9.2.3.3 Mahalanobis Distance

Unlike the Cosine-Angle distance and the Euclidean distance, the Mahalanobis distance [HA04] takes into account the shape of a given class distribution together with the location of the centroid vector. By using the covariance matrix (S), the Mahalanobis distance models the variance of features within the class distribution. It functions in the same manner as the above two measures by using the centroid vector (μ) to identify the location of the centre of the class distribution, however by utilising the covariance matrix of the distribution the scatter and shape of the labelled data points are incorporated into the measure. One of the limitations of the Mahalanobis distance however, is the requirement for a significant labelled sample set in order to calculate the covariance matrix: thereby capturing the shape of the distribution. It is defined formally as follows:

$$\text{mahalanobis}(\mu, x) = \sqrt{(x - \mu)^T S^{-1} (x - \mu)} \quad (9.20)$$

The above measures are often applied within clustering problems - an example of this is described in [QSGP04] which explored the relation between Euclidean Distance and Cosine Angle Distance when applied as distance measures for nearest neighbour queries.

9.3 Requirements for Semi-supervised Machine Learning Techniques

Based on the state of art with regards to semi-supervised machine learning techniques several limitations have been observed when using such a learning strategy. Therefore in order to train a classifier using a semi-supervised machine learning strategy the following requirements must be fulfilled:

- *Maximise the accuracy of the seed data*

It is essential to use an accurate strategy for generating the initial training data. Semi-supervised learning for binary classification requires a careful selection of the seed data: for instance when using Self-training, if false negatives are accidentally created and used to start the learning process, then such mistakes can have a negative “*snowball*” effect by reinforcing themselves. When applying Expectation-Maximization, labelled data should provide suitable coverage to ensure that a global maxima can be achieved.

- *Ensure accurate feature matching*

Machine learning classifiers will be supplied with instances for learning which contain a set of features extracted from an RDF model. The RDF model must therefore be transformed into the set of features. A careful choice of the right features reduces the number of variables involved in the learning problem, leading to classifiers that do not overfit the training sample and thus generalise better to new samples. Moreover, by modelling features in such a way different feature similarity measures can be investigated to work alongside the classifiers.

- *Accuracy must improve as the classifier is trained*

One of the hallmarks of semi-supervised learning is the strategy’s ability to iteratively improve the performance of a classifier. Unlabelled instances, once classified, may contain features which were previously unknown. By enriching the known feature set instances can then be classified correctly, which were previously labelled otherwise. The technique must be able to demonstrate this empirically by altering the performance of the hypothesis through learning.

- *Enlarge training data with suitable instances*

Semi-supervised learning strategies, such as self-training, improve the performance of a learned classifier by enlarging the training sets with selected examples. Careful selection of these instances is required, particularly given the possibility of mistakes to reinforce themselves.

9.4 Self-training: Semi-supervised Machine Learning

This section presents the application of Self-training to learn and improve a machine learning classifier for disambiguating *identity web references*. The strategy was chosen due to the limited amount of initial seed data; EM works well on datasets containing hidden variables but with sufficient labelled data to perform many restarts and choose the best convergent. Furthermore work by [NG00] explained the resilience of Self-training to convergence on a local maxima as a new labelled instance is added to the training data at every training

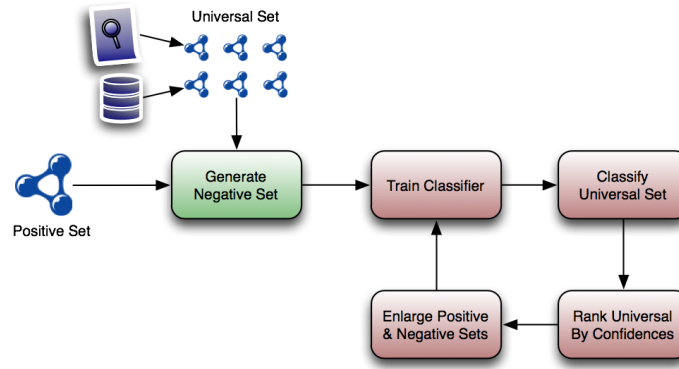


Figure 9.4: Applying Self-training to disambiguate *identity web references*

step. Co-training is a natural progression from the application of Self-training, therefore its application within the setting of this thesis is planned for future work (discussed later in chapter 12).

Through the application of Self-training as a wrapper algorithm two variations are permitted: a) different machine learning classifiers (both discriminative and generative), and b) different feature similarity measures. Therefore the investigation of different permutations of classifier and feature similarity measure provides a novel contribution to the current state of the art, particularly when investigating how the variation of the feature similarity measure will affect the performance of a self-trained classifier. Figure 9.4 presents an overview of the Self-training technique, which is capable of utilising an initial small set of positive labelled data to bootstrap the learning process.

The technique is divided into two distinct stages: the first stage gathers seed data to be used for learning a classifier. The positive set is created from the social graph produced from the reconciliation of digital identity fragments (chapter 5). The universal set of unlabelled data is comprised of the set of web resources to be disambiguated, where each resource is represented as a metadata model (using techniques from chapter 6). The negative set is then generated from the instances within the universal set using a strategy explained in the following subsection (§9.4.1).

The second stage of the approach uses the positive and negative sets as training data. Given that the disambiguation task is a binary classification problem (i.e., Does this web page mention person X or not?) labelling is restricted to two class labels; *positive* (citing a given person) and *negative* (not citing a given person). Once the classifier has been trained, instances from the universal set are labelled using either of the two labels coupled with a confidence score. The training data is then enlarged (positive and negative sets) by the classified resources exhibiting the strongest confidence scores. This process is repeated until a stopping condition is reached. It is worth noting however that the literature disagrees on what constitutes a suitable stopping condition. For instance, work by [RHS05, NG00] states that training should be performed until all unlabelled instances have been labelled. Work by [Yar95] believes that training should be performed until the classifier’s parameters converge on a stable state (similar to the stopping condition used by EM). And finally work by [Mih04, Koz06] perform training for a set number of iterations, observe the effects post classification and choose the best performing iteration at which future training is stopped.

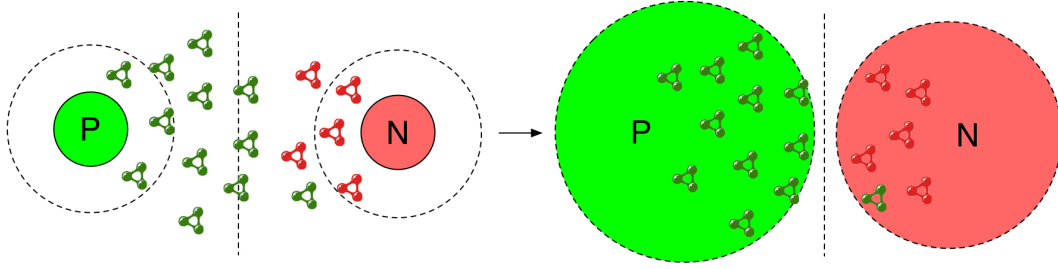


Figure 9.5: Initial training sets iteratively enlarged using resources with the strongest classification confidences

In the context of disambiguating *identity web references* it is important to return as many *identity web references* as possible whilst maximising the accuracy of the classifications. The ideal stopping condition would be to end training once the accuracy of the classifier’s hypothesis reaches a global maxima, however without a validation set which the classifier can be tested against after each training iteration this is not practical given that trained classifiers may behave differently when disambiguating web references for different people. Moreover as this technique investigates the effects on classifiers combined with different feature similarity measures, the stopping point will be different for distinct permutations of classifier and similarity measure. Therefore Self-training is performed until all unlabelled instances within the universal set have been classified, thereby employing the same stopping condition as [RHS05, NG00]. As Figure 9.5 shows, the general principle is that the positive and negative training sets are enlarged with all the instances from the universal set.

9.4.1 Generating Binary Seed Data

The social graph, represented as an RDF model and constructed from seed data collected from the Social Web, provides the initial machine learning instance to be used as positive training data. It is common for web users to provide links to additional profiles or web pages containing their work or information on their profile page. Based on the semantics of the exported social graph from such a profile, any resources described using *foaf:homepage* are assumed as being an *identity web reference*. Therefore an RDF model is generated from the linked web resource and is added to the positive set - using the techniques outlined in chapter 6. The resultant positive set therefore contains RDF models describing web resources which are *identity web references* for a given person.

In order for a binary classifier to be trained using Self-training both positive and negative seed data is required. Selecting negative examples from the universal set is a vital procedure which if performed incorrectly can result in a “*snowballing*” effect where false negatives reinforce themselves through training. Several strategies have been explained in the literature as solutions for this problem. For instance, work by [YHcC04] generates negative instances by compiling a list of positive features from the positive set. All instances from the universal set are then filtered out that contain any strong positive features from the list. Experiments using this method found that too many false negatives were generated (see Appendix M for experimental results using this method) resulting in poor levels of recall. Another strategy described in [FL06] builds a statistical distribution of features within the entire set and selects negative instances based on their features with regards to

this distribution. This strategy requires a sufficient amount of initial labelled data to enable the creation of the distribution model, however within the context of the described problem setting this strategy is not feasible due to the limited number of initial positive instances.

Instead a Rocchio classification model [BSA94] is used to generate *reliable* negatives. Rocchio classification is a method widely used within the information retrieval community as a form of relevance feedback. The general principle is that an optimum query (q) must be found that maximizes similarity with relevant documents whilst minimising similarity with non-relevant documents. By using a vector space model to represent the centroid vector of each document set the original query is modified such that it is either closer or further away from the set of relevant documents and the set of non-relevant documents [BSA94]. In terms of Rocchio classification, the original query vector is dropped and instead a prototype vector is constructed using the centroids of each of the document sets.

Algorithm 8 $\text{rochio}(P, U)$: Takes as input the positive set (P) and the universal set (U) and compiles positive and negative prototype vectors which instances are then compared against to derive negative set membership. The new universal set (U) and the negative set (N) are returned.

Input: P and U

Output: N and U

```

1:  $\vec{c}^p = \alpha \frac{1}{|P|} \sum_{x \in P} \frac{\vec{x}}{\|\vec{x}\|} - \beta \frac{1}{|U|} \sum_{x \in U} \frac{\vec{x}}{\|\vec{x}\|}$ 
2:  $\vec{c}^n = \alpha \frac{1}{|U|} \sum_{x \in U} \frac{\vec{x}}{\|\vec{x}\|} - \beta \frac{1}{|P|} \sum_{x \in P} \frac{\vec{x}}{\|\vec{x}\|}$ 
3:  $N' = \emptyset$ 
4: for each  $x \in U$  do
5:   if  $\text{cosine}(\vec{c}^n, \vec{x}) > \text{cosine}(\vec{c}^p, \vec{x})$  then
6:      $N' = N' \cup x$ 
7:   end if
8: end for
9: Rank  $N'$  by  $\text{cosine}(\vec{c}^n, \vec{x})$  where  $x \in N'$ 
10:  $N = \text{top } |P| \text{ from } N'$ 
11:  $U = U - N$ 
12: return  $N$  and  $U$ 
```

Rocchio classification functions by building prototype vectors for each class and then attributing instances to the vectors with which they have a higher similarity. The prototype vectors are constructed for the positive set and negative set as shown in Algorithm 8. For each prototype vector, the vector average of all relevant instances (positive in the case of the positive prototype vector) is derived, and from this the vector average of the set of non-relevant instances is subtracted (negative in the case of the positive prototype vector), thus providing *relevance feedback*. In the case where no negative examples are known, the universal set is treated as the negative set and the prototype vectors are constructed accordingly. The values of α and β affect the position of the derived vector within the feature space. For instance when building $\vec{c}^p$ if α is greater than β then the vector will be closer to the centroid of the positive set, conversely if β is greater then the vector will be closer to the negative set.

Once a prototype vector is constructed for the positive set and the negative set, then each instance in U is compared to the prototype vectors using cosine similarity (line 5). Feature vectors are compared at a statistical level by comparing the weights assigned to each feature

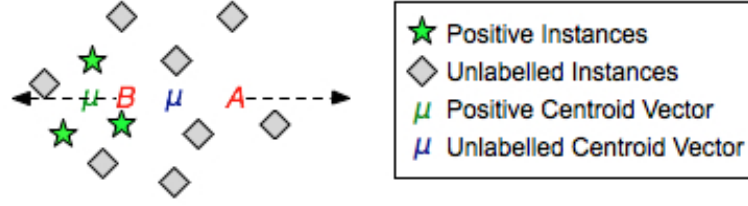


Figure 9.6: Rocchio classification in a vector space used to generate negatives

instance, it is worth noting that such comparisons are influenced by the feature similarity measures used during the generation of the feature vectors - the different measures and the process of building the feature vectors is explained in the following section. If the similarity is greater with the negative vector than the positive vector, then the instance is added to the negative set. This step identifies the set membership of the instance. The negative set is then ranked based on the cosine similarity scores and the top k instances are chosen from the set where k is the same size as the positive set (line 10).

The weights assigned to each centroid vector, defined by α and β for the set of relevant and non-relevant documents respectively, control the feedback provided to the given prototype vector. Work by [Mos03] defines the relation between setting suitable values for α and β by expressing a ratio of $\rho = \frac{\beta}{\alpha}$. Figure 9.6 present an example of how the proportion between the weights, defined by ρ , shifts the prototype vector of the negative set. As this prototype is built from the unlabelled data, at lower values of ρ - where $\alpha > \beta$ - the prototype vector (denoted by B in Figure 9.6) is moved towards the positive set - given that the feedback provided by the positive instances is minimised. As $\rho \rightarrow 0$ the likelihood of false negatives being produced increases. Conversely, at larger values of ρ - where $\alpha < \beta$ - the prototype vector (denoted by A in Figure 9.6) moves away from the positive set - given that the feedback provided by the known positive instances is increased. As $\rho \rightarrow \infty$ the likelihood of producing true negatives decreases - given that unlabelled instances will be identified as positives.

Within the literature the most widely used values for α and β are 16 and 4 respectively [BSA94]. The setting of these two values has been extensively studied in work by [Mos03], the results from the study suggest that $\alpha = 16$ and $\beta = 4$ (therefore $\rho = \frac{1}{4}$) performs well on input vectors of low dimensionality, whereas higher dimensions are better suited for $\rho = 1$. The most widely used application of Rocchio classification is for text classification, in this thesis negative set generation is performed using Rocchio classification for RDF models, thus presenting a unique problem setting. In short, a suitable value of ρ must be used to generate reliable negatives, the evaluation of this technique in chapter 10 presents experiments to derive such parameters and explains the limitations in basing the use of Rocchio classification on default, or community established, values.

The advantage of Rocchio classification is its ability to build a vector space model and then compute a decision boundary between classes using a linear classification model - similar to Perceptron and SVM. In the absence of pure negative examples the intuition therefore would be to choose the instances in the vector space which have a greater similarity with the negative prototype vector rather than the positive vector. The cosine similarity between a given instance and prototype vector measures the strength of membership to a given set.

The strongest negative instances will have a large similarity with the negative prototype vector, thereby indicating a strong bias towards that class membership, likewise the positive instances will have a large similarity with the positive vector. Instances which lie between these two planes are likely to contain features which are yet to be observed as discriminating a positive or negative classification. Rocchio classification has been shown to achieve high levels of recall [LL03] when approximating set membership over unlabelled data.

Related work within the literature such as [LL03] has investigated the use of Naive Bayes and SVM as classification models for generating the negative instances but found such models to work poorly without pure negative training data. The evaluation of Self-training in chapter 10 explores different values for ρ and their effects on the resultant classifications.

9.4.2 Learning to Classify Identity Web References

Labelled data is provided in two sets; *positive* and *negative*. The positive set contains the social graph and any resource associated with the social graph owner. The negative set contains web resources which do not cite the person whose *identity web references* are to be disambiguated. Unlabelled data is contained within the universal set. In all three of these sets, containing both labelled and unlabelled data, RDF models are supplied. Machine learning classifiers require feature vectors as an input dataset, therefore the provided RDF models must be converted into a suitable representation by extracting features from the RDF models. For simplicity hereafter the positive set will now be referred to as P , the negative set will be referred to as N and the universal set will be referred to as U .

9.4.2.1 Feature Extraction

RDF instances found within RDF models depict resources of information covering a variety of domains of discourse (i.e., people, publications, locations, events). Within chapter 4 of this thesis the use of social network information is discussed as a useful information source for aiding disambiguation, finding people from a person's social network will indicate that a given web resource cites the person. However, the use of merely instances of *foaf:Person* limits the scope of the features, given that the presence of a location or event which is associated with a person may act as definite indicator of their presence in a web resource. Instead RDF instances are used as features within machine learning instances. In order to extract these RDF instances the Resource Leaves construct (Equation (7.1)) is used from §7.4.1 which leverages the instance description as a triple set from the RDF model (such as the model from Figure 9.7).

By using this formal construct RDF instances are identified as shown in Figure 9.8 which are then used as features within the machine learning instance representation for that web resource. For example the web resource from Figure 9.7 would consist of 5 features each identified as a triple set, one example feature would be the RDF instance denoted by the node URI *#matt* and the relevant instance description. The decision to use such RDF instances therefore allows RDF instances within unlabelled web resource models to be incorporated into the learning process. For instance, if a web resource contains an RDFa description of an event which occurred recently, the leveraged model would contain an RDF instance describing the event together with those who attended it. As the classifier learns that this web resource is a positive instance - by matching the people appearing within the web

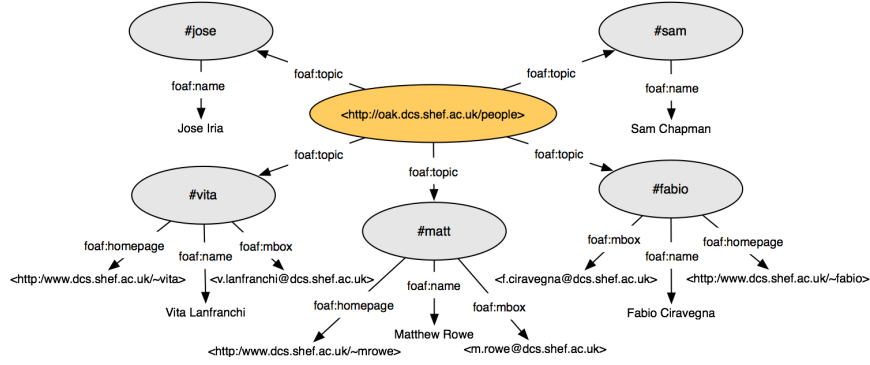


Figure 9.7: Section of an example RDF model representing information found within a given web resource

resource with people from the social network information in the seed data - then the event is weighted appropriately. Additional web resources which contain this event can then also be labelled as *positive*. This is one of the key advantages of using Semantic Web technologies combined with machine learning strategies, by formalising the knowledge structure, unique resources can then be represented, which in turn aid the classification of web resources with respect to a given person. In terms of machine learning theory, this stage in the approach performs dimensionality reduction given that the feature space could be large - i.e., by using resources and literals as features - but is instead reduced to use RDF instances.

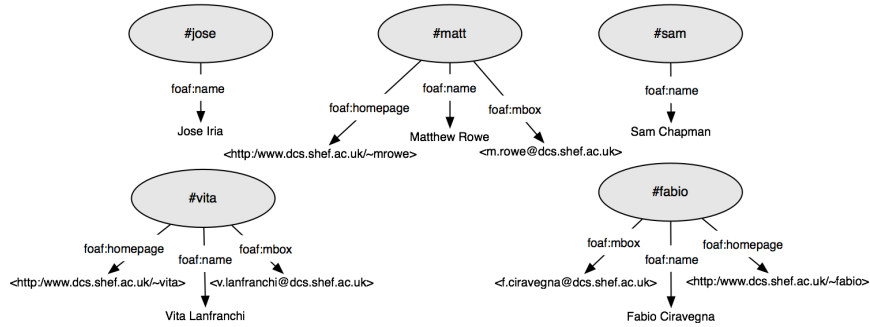


Figure 9.8: RDF instances to be used as features

9.4.2.2 Feature Similarity Measures

Machine learning instances are compared for learning backed by RDF graphs. Given that features of these learning instances are extracted as RDF instances it is possible to vary the feature similarity measure used. This variation explores the use of different RDF graph comparison techniques from the literature. Three such techniques are now presented, which in terms of the strategy, are designed to detect equivalence between RDF instances.

Jaccard Similarity: RDF Graph Equivalence

The Jaccard similarity of graphs as described in [BDKW06] (and introduced initially in §5.2.2.1) looks at how similar two given graphs are based on the number of edits required to transform one graph into the other. Jaccard similarity is a basic measure of graph

similarity and requires the learning of a given threshold (σ), which when exceeded classes two graphs as equivalent, therefore declaring two RDF instances as representing the same entity, or 'thing'. For a formal definition of Jaccard similarity please refer to §5.2.2.1. Within chapter 5 this same similarity measure is used for graph matching when integrating social graphs from disparate Social Web platforms where the threshold, which when exceeded classes two graphs as being matched, is set based on tuning. Within the context of this technique however heterogeneous RDF instances are often compared, thereby restricting the applicability of a tuned threshold.

Instance descriptions vary largely in their size therefore making tuning not feasible: an instance of a publication defined as an instantiation of the concept *bibtex:Entry* could have many properties attributed to the resource (title of the paper, author name, conference, publication journal, etc), whereas an instance of *foaf:Person* may only include the name and the email address. This variability in the amount of information attributed to the instance description can yield unreliable results, and therefore miss matches which should have been previously matched yet simply vary in the amount of information. Therefore rather than attempting to tune the threshold which the Jaccard similarity must exceed it is instead set to 1, thereby requiring strict equivalence between instances. The intuition behind setting the threshold at such a high value is to guarantee precise results over recall levels. Moreover, at this stage in the approach it is impossible to know for sure what RDF instances may appear within gleaned models from the collected web resources. To this end setting the threshold to denote strict equivalence will avert any misclassifications of instances when comparing features.

Inverse Functional Property Matching

One of the key advantages of modelling information semantically is the expressivity of certain properties and relations. One important example of expressivity is the use of inverse functional properties. According to the FOAF ontology, properties such as *foaf:mbor* and *foaf:homepage* are typed as *owl:InverseFunctionalProperty*, therefore depicting a one-to-one relation between an instance of a class and the attributed resource. For example, consider that a person has an email address, no other person can have that same email address, therefore it can be said that the email address is a unique identifier of that person. Therefore RDF instances are matched, and defined as being equivalent, when inverse functional properties match from both instances (using the same "graph smushing" principle as described in §5.2.2.3).

RDF Entailment

It is common for equivalent RDF instances to differ in size depending on the amount of information within the instance descriptions. Consider two RDF instances g and h describing people: g contains the person's name, email address and web site, whereas h contains the person's name and email address. One can infer that both g and h refer to the same person, and therefore the tripliset of h is a subset of the tripliset of g . According to Pat Hayes' Interpolation Lemma¹ and work in [LZ06], g and h are matched when h entails g ($h \models g$) (every model of h is also a model of g). The intuition behind this approach is the variability in the amount of information available from certain web resources. When relying on the

¹<http://www.w3.org/TR/2004/REC-rdf-mt-20040210/#entail>

comparison of model structures (i.e., verifying that structural properties of the models are similar) equivalent instances can be missed.

9.4.2.3 Mapping RDF models into a Machine Learning Dataset

In order to use RDF models as instances from which classifiers can learn, these models must be represented as binary feature vectors where each feature represents a distinct RDF instance from the feature extraction step. Therefore a given RDF model is mapped into a machine learning dataset using the following three steps:

1. One RDF subgraph per instance is extracted from the RDF model of the web resource using the Resource Leaves construct from chapter 7.
2. Each extracted RDF subgraph is converted into a triplesets from the instance description to enable comparison using the feature similarity measures. The metrics work at the statement level by comparing the triplesets of each RDF instance and deriving a match.
3. Machine learning instances are constructed as binary features vectors using a feature indexer (i.e., x_i where the index i denotes a given feature). To compile a new feature vector representation of a machine learning instance, indexes are derived for the features of the given web resource by comparing those features with existing features within the indexer using the similarity measures from the previous section. If a feature is matched then the relevant index is returned, otherwise a new index is created. Given that the features are represented as RDF subgraphs, the feature matching technique can be varied thereby effecting the indexes within the feature vectors and the resultant dataset.

The above steps produce a binary feature vector representation for a given web resource where each binary feature represents the presence of a given RDF subgraph. The feature similarity measures therefore vary the indexing structure and results in unique feature vectors being produced for each distinct similarity measure. This is achieved through the modelling of web resources as RDF models, and instance features as RDF instance descriptions in the RDF model. More generalised features could also be included within this representation such as the presence of relations between instances within a given model (e.g., the presence of a *foaf:knows* relation between two people) or a count of unique person co-occurrence within a given page.

Within this disambiguation technique three machine learning classifiers are compared: Perceptron [Bis95], Support Vector Machines [DPHS98] and Naive Bayes [Mit97]. Naive Bayes derives the conditional probabilities from labelled instances and their features to build the initial generative classification model. SVM and Perceptron construct their discriminative hyperplanes within the feature space based on the labelled instances within the dataset. By mapping the RDF models of each web resource into such a dataset, the internal functionality of the machine learning classifiers is not altered. Instead the classifiers are used merely as a “*blackbox*” by inputting the labelled and unlabelled data for training and classification and returning the classification labels for instances within U along with the classification confidences of those labels.

9.4.2.4 Self-training

The Self-training strategy (as Algorithm 9 explains) begins by first using the seed data - positive training data is given by the social graph and negative training data is given by instances returned using Rocchio classification - to train a classifier. At this stage the size of the training sets are relatively small. Instances from U are classified using the initially trained classifier and the instances are ranked based on their classification confidences. The top k positive instances and the top k negative instances from U are chosen according to rankings, and P and N are enlarged with those instances respectively. The term *classification confidence* is used as a general term to encapsulate the different outcomes from the classifiers. For Perceptron and SVM this is the value returned by the function $f(x)$ with respect to the inputs and for Naive Bayes this value corresponds to the likelihood estimates of the class membership over the learnt probability distribution.

The strongest k positives and strongest k negatives are also removed from U , thereby decreasing the size of the universal set. This process is repeated by retraining the classifier and reapplying it to the universal set until the universal set becomes empty. The intuition behind this strategy, and as has already been mentioned, is such that the trained classifier will improve over time. The hypothesis will also gradually move towards a stable state and the classification confidence of instances will begin to converge.

Algorithm 9 $\text{itEnl}(P, N, U)$: Iterative self-training based on enlargement of training data according to classification rankings. P is the positive set, N is the negative set and U is the universal set

Input: P , N and U

Output: P and N

```

1: while  $U \neq \emptyset$  do
2:   Train  $\Psi$  using  $P$  and  $N$ 
3:   Classify  $U$  using  $\Psi$ 
4:   Rank  $U$  based on classification confidences
5:   Enlarge training sets by strongest  $k$  positive ( $U_k^P$ ) and negative ( $U_k^N$ ) instances
6:      $P = P \cup U_k^P$ 
7:      $U = U - U_k^P$ 
8:      $N = N \cup U_k^N$ 
9:      $U = U - U_k^N$ 
10: end while
11: return  $P$  and  $N$ 

```

The ranking strategy was implemented based on work by [FL06]. Instances are ranked based on their classification confidences and then the relevant training sets are enlarged with the top k instances. In doing so only the optimum classifications are made, in order to avoid any mis-classifications or reduce the likelihood of that occurring. For instance, adding all negative labelled examples into the training set could lead to false negatives which cannot be later re-classified as positive (given that Self-training maintains a fixed set of labelled data unlike the Yarowsky algorithm). The Rocchio classification approach used to generate negative examples may not yield the best predictions, in this instance the enlargement of the training sets with all negatives following each training iteration could result in missed web references. Work by [Mih04] tested different *growth sizes* when performing Self-training for text classification and found $k = 1$ to yield the optimum setting, similarly [NG00] chose the single strongest instance to enlarge the training data with when bootstrapping a classifier.

The evaluation within chapter 10 assesses the difference in performance when enlarging the negative set with the strongest instance against enlargement using all negative labelled instances.

9.5 Summary

In this chapter a semi-supervised machine learning strategy called Self-training has been discussed. Using this technique machine learning classifiers are iteratively trained, and used to disambiguate *identity web references* for a given person. Such processes require both positive and negative seed data to train the initial classifier, therefore a technique has been explained within this chapter for approximating negative seed data to accompany the already supplied positive data - which is provided as a social graph from seed data collected from the Social Web (chapter 5). Self-training acts as a wrapper algorithm for training the machine learning classifiers Perceptron, SVM and Naive Bayes, thereby providing a framework to investigate the variation in learning both discriminative and generative models for classification. However one of the limitations of the Self-training strategy is its sensitivity to mistakes within the seed data which subsequently reinforce themselves and cause a “*snowball*” effect on later classifications, therefore an evaluation is conducted in chapter 10 to observe the effects of training by comparing the resultant trained hypothesis against the initial learnt hypothesis.

By mapping RDF models into a suitable machine learning dataset, machine learning classifiers can be used as “*blackboxes*” by inputting the dataset and returning labelled instances together with their classification confidences. This therefore fits neatly into the Self-training strategy, where the labelled instances are added to the training sets after each training iteration. Learning is bootstrapped with the intention of minimising the need for supervision or intervention. Using Self-training allows for different permutations of machine learning classifier and feature similarity measure to be analysed. Chapter 10 presents a detailed evaluation of the differing performance of distinct permutations and how training effects the classifier’s hypothesis.

Chapter 10

Evaluation

10.1 Introduction

Chapter 7, chapter 8 and chapter 9 presented three complimentary disambiguation techniques. Each technique is supplied with seed data and a set of web resources to be disambiguated, and performs the disambiguation process in a unique manner. The requirements listed within chapter 4 stipulated that accuracy levels must be achieved by the investigated techniques which are comparable to human processing of the same data. This requirement addresses one of the claims of this thesis that automated techniques are able to replace the manual process performed by humans wishing to find web resources containing their personal information - so called *identity web references*. Therefore this chapter presents a detailed evaluation of the disambiguation techniques investigated within this thesis. Several baseline measures are used including an example state of the art unsupervised technique and human processing. The evaluation results present empirical evidence to justify several of the claims of this thesis.

The application of *semi-supervised* disambiguation techniques incorporates both labelled and unlabelled data into the decision process. As a result such techniques have the advantage of adapting to features present with the unlabeled data, thereby enhancing disambiguation decisions. Such techniques should therefore outperform *supervised* techniques by demonstrating their ability to detect additional identity web references. This chapter contrasts the performance of Inference Rules (*supervised*), Random Walks (*semi-supervised*) and Self-trained classifiers; before-training (*supervised*) and after-training (*semi-supervised*). The results of this study demonstrate the need to utilise unlabeled data and the ability of the *semi-supervised* techniques to learn from the features not present within the provided seed data in order to detect additional web references - contextualised by high recall levels.

This chapter is structured as follows: section 10.2 details the measures used within the evaluation for assessing the accuracy of each of the techniques. Section 10.3 describes the dataset used for the evaluation, detailing how it was constructed and the selection of participants. Section 10.4 describes the experimental setup for each of the disambiguation techniques including tuning of parameters. Section 10.5 discusses the selection of the baseline techniques that the presented disambiguation techniques are to be compared against. Section 10.6 discusses the construction of the gold standard used for evaluation. Section 10.7 presents the results from the evaluation, which is divided into subsections for each disambiguation

technique and their results are discussed. Section 10.8 contrasts the use of metadata models against a shallow approach using a bag-of-words model to represent web resources features, in doing so contrasting the differences in performance based on the production of rich semantic markup. Finally section 10.9 concludes the chapter with a summary of the findings.

10.2 Evaluation Measures

Each of the disambiguation techniques presented within this thesis divide a set of web resources into two distinct sets; those that refer to a given person (*positive*) and those that do not (*negative*). In order to assess the performance of each technique the accuracy of these derived sets must be assessed and information retrieval metrics [vR79] are perfectly suited to this task. The chosen information retrieval metrics feature two sets of documents: A denotes the set of relevant documents and B denotes the set of retrieved documents. Using these set definitions, precision and recall are defined as follows:

$$precision = \frac{|A \cap B|}{|B|} \quad (10.1)$$

$$recall = \frac{|A \cap B|}{|A|} \quad (10.2)$$

The sets produced by the disambiguation techniques are comparable to the above document sets, where the set of relevant documents is the set of relevant web resources (definitely citing the person) and the set of retrieved documents is the set of retrieved web resources. Precision is utilised to measure the proportion of web resources which are labelled as citing a given person and are true references. Meanwhile recall measures the proportion of true references that were retrieved. In the majority of classification tasks, both precision and recall are combined into a single measure defined as the harmonic mean between the two metrics known as f-measure, this is defined formally as follows:

$$f - measure_{\beta} = \frac{(1 + \beta^2) \cdot (precision \cdot recall)}{(\beta^2 \cdot precision + recall)} \quad (10.3)$$

Using the f-measure allows the priority of precision over recall to be adjusted using a suitable setting for β , where $\beta > 1$ weighs recall as being more important than precision and $\beta < 1$ weighs precision as being more important than recall, $\beta = 1$ weighs precision and recall as being of equal importance. Within the context of this thesis the need to find those web resources which contain a given person's personal information is vital, given that the information could have drastic consequences for the individual. Such a requirement therefore weighs recall as being more important than precision where a suitable value for β should be set. However each web user may weight recall over precision differently (i.e., a paranoid person may always want 100% recall and does not care about precision, whereas another person with a more liberal outlook may want to see what information is out there but does not want any incorrect citations returned), thus making a global setting for β hard to gauge.

To analyse differing performance levels of the presented disambiguation techniques precision/recall curves are used, and the best precision achieved at recall level 90% is used for assessment. To place this in context, this will assess the precision achieved by the technique

when at least 90% of the *identity web references* have been returned, the higher the precision, the better the performance of the technique. Recent work by [RYGR10] has used this same measure to good effect in order to evaluate the categorisation of web documents into distinct topics. In order to also provide additional measures for comparison between the techniques, f-measure is also used with incremental integer values for β : F_1 : where precision and recall are weighted equal; F_2 where recall is weighted twice as much as precision and F_3 where recall is weighted three times as important as precision. Using such a measure shows the alteration in performance of the differing methods as recall becomes more important than precision, this is particularly evident when comparing the performance of the techniques against human performance - the baseline of which is discussed shortly.

10.3 Dataset

The evaluation dataset was compiled using 50 members of the Semantic Web and Web 2.0 communities as participants. The participants were selected at random from conference attendee listings and based on their presence on both Twitter and Facebook - requiring each participant to be on both platforms. For each participant seed data was produced by generating individual social graphs from Facebook using the FOAF Generator (§5.5.1.1) and Twitter using the Twitter Exporter (§5.5.2.1). Each of these social graphs were then interlinked to produce a single social graph (§5.6.2) using the previously described methods. Although a linked data form of the graph is produced, the linked resource descriptions are dereferenced into a single graph for use by the presented disambiguation methods.

To gather a collection of possible web references, the search engines Google and Yahoo were queried, together with the Semantic Web search engines Sindice and Watson, each time using the name of the participant as the query input (e.g., *Matthew Rowe*). It is possible however that several web references are missed using this strategy, given that a set number of results are cached from each search engine (first 100), and the person in question may share the same name with another namesake who has a higher web presence (i.e., a celebrity). In such cases, humans often use query expansion to concatenate an additional piece of discriminative information to the query input, for example *Matthew Rowe Student*. However, this could bias the results to reduce the noise and citations of other namesake within the results. The presented techniques must be tested to ensure their ability to function over results which contain information about many disparate namesakes and perform disambiguation effectively, therefore the above query input - consisting of the person's name - is sufficient for this task.

Following querying of the search engines, the first 100 results were cached for the experiments, using the same number of results (100) as the dataset collected for the Web People Search Evaluation Campaign [AGV05]. It is worth noting that the first 100 results from the listed search engines differ - this is reflected in the percentage that was identified as being duplications (13.5%), therefore by using several distinct search engines a broader collection of possible web references will be collected. This should, in theory limit the detrimental effect that collecting the first 100 results will have on recall levels - given that the possibility of finding a larger amount personal information fragments is increased.

Within the returned web resources from querying the search engines, there were many duplicates (i.e., equivalent publication and homepages were returned by Google and Yahoo)

and several web resources were also noise (i.e., PDF files which cannot be parsed), these web resources were removed. Therefore the resulting dataset contained approximately 17300 web resources with ~ 346 web resources to be disambiguated for each participant. Metadata models were then built to represent information found within each of these web resources using the techniques described within chapter 6. Although the sample of participants was taken from computing communities, the level of available metadata in the collected web resources from both Google and Yahoo was below the levels defined in [MMZ09], where [MMZ09] found 2.6% of web resources contained either the hCard microformat or some form of RDFa, the collected dataset in this instance only contained 0.5%.

10.4 Experimental Setup

The dataset, following its compilation, was split into training and testing sets using a 10% to 90% split for training and testing respectively. Each of the explored techniques, where applicable, had parameters tuned on the 10% training split, and was then tested on the remaining 90%. Each technique was run for a single participant at a time: inputting the social graph for the person and the collection of possible web citations, and outputting the disambiguation results. Parameters for experiments over the 90% testing split were set as follows for the three distinct disambiguation techniques.

10.4.1 Inference Rules

The Inference Rule base is compiled from the seed data for a given participant, and the rules are then applied to the set of web resources for that subject. Inference Rules require no tuning or setting of parameters for the experiments.

10.4.2 Random Walks

For each participant an individual graph-space is compiled from the seed data (the inter-linked social graph) and the web resources to be disambiguated for that person (each modelled as an RDF model/graph). The presented graph-based disambiguation technique relies on agglomerative clustering to cluster web resources closest to the social graph together. Therefore a threshold must be derived, which when the distance between two clusters is less than, then the clusters are merged. Values of the threshold must be derived for both Commute Time and Optimum Transitions - given that these are the two measures that are explored. As stipulated in §8.4.5 the value of this threshold is defined as being $0 \leq \sigma \leq 1$. Chapter 8 also proposes an edge weighting proportion η such that unique and non-unique edges labels are assigned separate weights.

The derivation of optimum values parameters for σ and η was performed using the 10% training split of the dataset, the discussion and results of which are shown in Appendix J. For both Commute Time distances and Optimum Transitions the optimum parameters were found to be $\sigma = 0.8$ and $\eta = 1$. These values of σ and η were then used to perform the evaluation over the remainder of the dataset.

10.4.3 Self-training

The use of Self-training contrasts the performance of different combinations of machine learning classifier with feature similarity measures. Three classifiers; Perceptron, SVM and Naive Bayes are each combined with three measures; Jaccard similarity, Inverse Functional Property matching and RDF Entailment to produce nine individual permutations. Self-training relies on the use of Rocchio classification to generate negative training data via prototype vectors and cosine similarities. When applying Rocchio classification a suitable setting of ρ must be used for experimentation, therefore for each permutation of classifier and feature similarity measure tuning of ρ was performed using the 10% training split of the dataset and the parameter was chosen which optimised precision at recall level 90% - the discussion and findings from the tuning experiments are described in Appendix K. The results presented in this chapter are from enlargement of both the positive and negative set using the strongest instances following ranking. Experiments are also presented which compare the effects of enlargement of the negative set using the strongest instance against all negative labelled instances.

In order to test the accuracy of each of three machine learning classifiers the machine learning framework Aleph¹ was used. This framework contains built in classifier algorithms which require an input dataset for training, which consists of labelled instances, that can then be applied to a dataset for testing. As described in §9.4.2.3 the RDF models of each of the web resources are mapped into the dataset from which a classifier can be trained and tested.

10.5 Baseline Techniques

In order to provide a contextual assessment of each of the presented disambiguation techniques three baseline measures were used and are defined as follows.

10.5.1 Baseline 1: Person Name as Positive Classification

Comparing the performance of classifications where a web resource containing the participant's name is classed as an *identity web reference* demonstrates the imprecision of such a strategy, and the requirement for additional seed data to aid the classification process. This baseline technique forms a general rule (similar to the skeleton rule constructed when building rules in chapter 7) which matches the presence of a person's name in order to infer a web reference:

```
PREFIX foaf:<http://xmlns.com/foaf/0.1/>
CONSTRUCT { <http://www.dcs.shef.ac.uk/~mrowe/foaf.rdf#me> foaf:page ?url }
WHERE {
    <http://www.dcs.shef.ac.uk/~mrowe/foaf.rdf#me> foaf:name ?n .
    ?url foaf:topic ?p .
    ?p foaf:name ?n
}
```

¹<http://aleph-ml.sourceforge.net/>

10.5.2 Baseline 2: Hierarchical Clustering using Person Names

A latent semantic analysis method using hierarchical clustering is presented in [Mal05]. This method uses an unsupervised feature comparison technique where web resources are clustered based on the presence of person names within resources. Such an approach provides a suitable *unseeded* disambiguation technique for comparing the presented methods within this thesis against. The results from experiments using this technique provide an insight into the differing performance levels between *seeded* and *unseeded* disambiguation techniques. This technique uses a clustering strategy which produces k clusters, where k defines the number of distinct namesakes within the set of web resources. The cluster describing the participant is chosen as the *positive* set and the remaining clusters are merged to form the *negative* set. Tuning for this baseline is discussed within the Appendix I

10.5.3 Baseline 3: Human Processing

The third baseline measure compares the presented disambiguation techniques against human processing of the same data. To provide this baseline measure a group of 12 people, denoted as raters, were set the task of manually processing a portion of the dataset. Three different raters disambiguated web resources for each evaluation participant, thereby providing three distinct sets of results. Interrater agreement was then used to find the agreement of results between the raters according to techniques in [HR05], by using one rater's results as the gold standard and another rater's results as the test subject. This provides measures for precision and recall, and therefore f-measure. The average of the three separate rater agreement measures for each participant was then computed.

10.5.4 Baseline 4: Outlier Detection

As discussed within chapter 9 of this thesis, classification models are not limited to a binary classification scenario. Density estimators and proximity measures provide a suitable solution to avoiding the binary classification task and the subsequent need for generating negative training data to balance the hypothesis. However, within the problem setting of this thesis the availability of positive labelled data - describing true web references of a given person - is in short supply, thereby rendering density based techniques limited in their ability to build an estimation model over observed data.

To compare the binary classifiers discussed within chapter 9, a baseline implementation of using Cosine Angle distance is implemented. The prototype vector is calculated for the provided labelled data: comprising of the social graph and the linked web resources to the URI of the social graph owner, before calculating the maximum radius of the labelled data distribution. The unlabelled data - the collection of web resources which are to be disambiguated - are then converted into a feature vector representation, using the same features as the Self-training approach together with the variation in feature similarity measures, and compared with the centroid vector of the labelled data distribution. All data points that fall outside of the distribution are classed as outliers, the data distribution is then iteratively rebuilt using those data points that fall within the distribution - thus mimicking the nature of the Self-training algorithm while only considering positive set enlargement.

10.6 Constructing a Gold Standard

Evaluation of disambiguation techniques requires a gold standard to be constructed by labelling the dataset by hand. Classic methods of gold standard construction utilise multiple labellers and take the majority opinion to assign a given class label. For instance, imagine there are three domain experts A , B and C . Each expert must decide whether <http://www.google.com> is a web search engine page. A and B say yes, whereas C says no. As the majority opinion is that the page is a web search engine page then the URL is assigned the relevant label. Unfortunately performing this majority opinion strategy for the entire dataset (~ 17300 web resources) is an expensive task requiring time and raters willing to process a large amount of data. Instead a gold standard (GS') was created for a portion of the dataset (50 web resources for each participant, so 2500 web resources in total) using the majority opinion of the raters from Baseline 3.

Table 10.1: Rater agreement

GS	GS'	
	Positive	Negative
Positive	a	b
Negative	c	d

A single human was then set the activity of labelling the entire dataset by hand to create a second gold standard (GS) which the disambiguation techniques could then be compared against. This is essential as such techniques must be able to process large amounts of data given the size of the Web and the need to discover the maximum number of web references for a given person. One of the issues with constructing a gold standard using only a single domain 'expert' is the reliability of the gold standard, it is important to quantify the measure of reliability prior to applying it to evaluation tasks. A suitable quantifiable measure of reliability is the Kappa statistic [Fle81] which derives the chance corrected agreement between labelled sets. Using the set definitions from Table 10.1 the statistic is defined as follows:

$$\kappa = \frac{2(ad - bc)}{(a + c)(c + d) + (b + d)(a + b)} \quad (10.4)$$

Higher values of κ (upper bound is 1) indicate a greater level of agreement. The Kappa statistic has been widely used and explored within the field of medicine and research into treatment by calculating the agreed consensus with regards to medical decisions. Work by [KP03] stated that percentiles of the Kappa statistic correspond to agreement labels as shown in Table 10.2.

The Kappa statistic was calculated for the agreement between GS' and GS for each participant within the dataset and taken as an average of the 50 participants. Of course GS is a lot bigger but if it is reliable then it should contain a large agreement level with GS' by observing the same labels for the portion of the dataset which GS' was constructed from. The Kappa statistic was found to be 0.899 overall indicating, according to Table 10.2, almost perfect agreement between the two gold standards and suggesting that it is a suitable reliable gold standard to be used for evaluation. This is due to the subject labelling the

Table 10.2: Strength of agreement when using Kappa (κ) statistic

κ	Agreement Strength
< 0	Poor
$0.0 - 0.20$	Slight
$0.21 - 0.40$	Fair
$0.41 - 0.60$	Moderate
$0.61 - 0.80$	Substantial
$0.81 - 1.00$	Almost Perfect

entire dataset having prior knowledge and additional expertise of the evaluation participants (involved more within Semantic Web and Web 2.0 communities), meanwhile the raters used to construct GS' did not.

10.7 Results and Discussion

This section now presents the results from the experiments that were performed using each disambiguation technique. Each technique is compared against the previously described baseline measures. The experiments were also conducted over the portion of the dataset used for generating GS' , the results from these experiments are found within Appendix H.

10.7.1 Inference Rules

Inference Rules achieve high levels of precision with respect to the four baseline techniques (as shown in Table 10.3). This is due to the strict nature of the rules, requiring a web resource’s knowledge structure to exactly match the tripleset of a rule’s antecedent. If any information within the model is altered slightly (i.e., abbreviations of literal values), then the rule fails. A consequence of this strict matching of knowledge structures is poor levels of recall as indicated within Table 10.3. Moreover, unlike Self-training and Random Walks, rules do not learn from their disambiguation decisions in order to improve later classifications.

Although this technique achieves poor coverage of the whole set of web resources, it does achieve comparable F_3 score with respect to Baseline 2 (the unsupervised method). This suggests that a *supervised* disambiguation technique - given that labelled data is supplied and used to learn a labelling function - performs equally as well as an example *unsupervised* technique - given that no labelled data is supplied. By increasing β when calculating f-measure, levels achieved by inference rules reduces as recall becomes gains in importance over precision.

Table 10.3: Accuracy of Inference Rules compared to baseline techniques

	Precision	Recall	F_1	F_2	F_3
Inference Rules	0.955	0.436	0.524	0.501	0.482
Baseline 1	0.213	0.999	0.330	0.519	0.663
Baseline 2	0.239	0.748	0.288	0.404	0.492
Baseline 3	0.765	0.725	0.719	0.703	0.697
Baseline 4	0.226	0.956	0.324	0.487	0.614

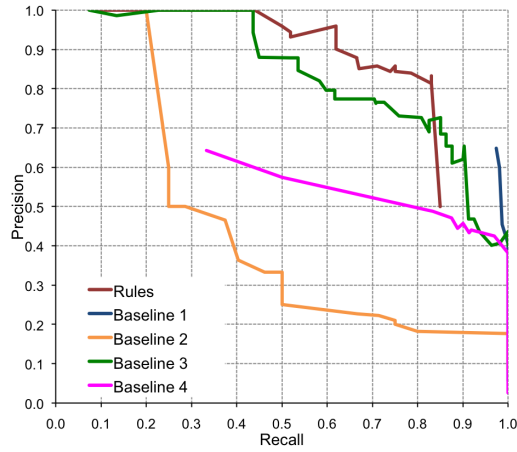


Figure 10.1: Precision/Recall curve for Inference Rules plotted against the baseline techniques

By plotting precision against recall levels, Figure 10.1 shows the differing performance levels achieved by each technique. Baseline 1, which uses the existence of a participant's name as being a positive indicator of a web reference, achieves high levels of recall for low-levels of precision. Baseline 2 achieve high levels of precision at low recall levels, however as recall increases, precision reduces. Baseline 3: human processing, achieves high levels of precision at lower levels of recall, however unlike baseline 2, as recall increases precision does not reduce as drastically. Rules are able to achieve higher levels of precision that humans as recall increases, characterised by the shape of the curve in Figure 10.1.

The goal of plotting the precision/recall curves is to compare the effects of differing techniques in situations where either precision or recall are considered important. As Figure 10.1 demonstrates with respect to baseline 4 - the application of Cosine Angle distance (CAD) as a proximity measure, up until recall 85%, rules outperforms this method, achieving noticeably higher precision than the outlier detection technique. However focussing on precision at recall level 90%, Inference Rules fails to achieve recall above 0.85, demonstrating its limitations in returning a large number of web references.

In situations where precision is considered more important than recall, Rules are better suited than all of the presented baselines. Using the Sign Test [NS00] ($p < 0.01$) precision levels with respect to human processing are found to be statistically different. Further analysis of the precision/recall curve shown in Figure 10.1 graphically demonstrates the applicability of rules in situations where precision is considered more important than recall - i.e., where the goal is to find information quickly, and noise in returned results should be kept to a minimum. However the poor recall levels achieved by the technique renders it limited in the context of detecting the optimum number of web references, this therefore inhibits its applicability in situations where recall is prioritised over precision. This technique could be extended however to be used as a rule-based classifier, which could be wrapped within a semi-supervised learning algorithm, similar to Self-training. This application is planned for future work, and is discussed in chapter 12 in more detail.

10.7.2 Random Walks

Table 10.4 presents the evaluation results of disambiguating using Random Walks and clustering using the two distance measures from chapter 8; Commute Time and Optimum Transitions. Both Commute Time and Optimum Transitions produce high levels of recall outperforming 3 (human processing), and precision levels which outperform Baseline 1 (person name as positive classification), baseline 2 and baseline 4 (outlier detection), however recall levels are lower than these methods. Assessing the different f-measure levels, where precision and recall are weighted equally (F_1) humans outperform the Random Walks based method. As the importance of recall increases, indicated by both F_2 and F_3 , both Commute Time and Optimum Transitions outperform humans.

Table 10.4: Accuracy of Random Walks and clustering using Commute Time and Optimum Transitions compared to baseline techniques

	Precision	Recall	F_1	F_2	F_3
Commute Time	0.683	0.793	0.667	0.723	0.751
Optimum Transitions	0.648	0.804	0.650	0.713	0.748
Baseline 1	0.213	0.999	0.330	0.519	0.663
Baseline 2	0.239	0.748	0.288	0.404	0.492
Baseline 3	0.765	0.725	0.719	0.703	0.697
Baseline 4	0.226	0.956	0.324	0.487	0.614

As mentioned previously, the tradeoff and prioritisation of precision over recall, and vice versa, is dependent on the user: given the independent nature of their personal information on the Web and the bespoke requirements for finding it. Therefore, in a similar manner to the analysis of Inference Rules, precision/recall curves of the Random Walks method using the two distinct distance metrics are presented in Figure 10.2. The curve for Commute Time almost completely subsumes the precision/recall curve of Optimum Transitions, indicating its superior performance. In comparison with human processing, Commute Time and Optimum Transitions outperform humans in terms of recall at precision 90%. For precision at recall 90% humans achieve 0.651, whereas Commute Time yields 0.566 and Optimum Transitions produces 0.514 at the same recall level. Statistical significance testing was performed using the sign test ($p < 0.01$) and clustering using Commute Time distances was found to be significantly different in F_3 levels with respect to human processing.

The inclusion of unlabelled data in the decision process demonstrates the increased performance in terms of recall levels with respect the previously discussed supervised disambiguation approach: Inference Rules. However, a notable by-product of the inclusion of unlabelled data is the reduction in precision, as false positives are included in disambiguation decisions. This graph-based technique constructs a graph-space by merging RDF models representing disparate web resources, and performing traversals through the graph-space from the social graph and measuring distances to web resource root nodes. This novel use of RDF provides many possible paths for traversal, edges are in turn weighted based on their ontological properties by *looking up* the concept's metadata description within the referenced ontology and assessing the provided description. This enables Random Walks to be applied within disparate domains, not solely the disambiguation of web references for people, given that the intrinsic methodology of the approach is generic - i.e., includes no ad-hoc strategies or domain-specific coding.

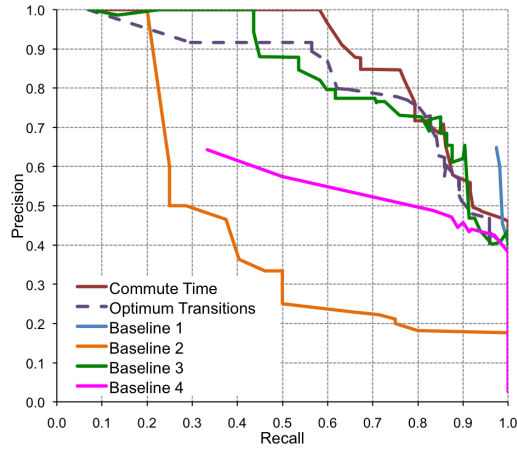


Figure 10.2: Precision/Recall curve for Random Walks using both Commute Time and Optimum Transitions to cluster, plotted against the baseline techniques

Humans maintain high levels of precision, yet achieve lower recall levels than Random Walks using the two explored distance measures. This is due to a “*needle in a haystack*” problem, where humans are able to ensure that the web references they return are accurate - contain a limited number of false positives - while often failing to detect the maximum number of web references. Surveying web resources for citations is a laborious and time-consuming task, information must be analysed thoroughly and the correct decision made, as a result this requires high-levels of concentration on the part of the human, should the individual tune-out then web references are missed. Automated techniques meanwhile do not suffer from this problem.

10.7.3 Self-training

Table 10.5 presents the results from performing Self-training using different permutations of machine learning classifiers and feature similarity measures. The results indicate that Jaccard yielded the highest precision levels by detecting strict graph equivalence between RDF instances. However recall levels were consistently lower than the two alternative feature measures as a consequence of this strict feature matching. As a result Jaccard is more accurate at matching features yet often misses *identity web references* where the two alternative measures do not. Higher levels of recall were achieved by Entailment due to the measure’s ability to match features where the descriptions vary in size, and the RDF graph of one feature subsumes that of another feature, yet contain equivalent information. This reduces overfitting to the training data and allows the method - when combined with differing classifiers - to generalise to new instances, causing additional classification of *identity web references* by learning features which are missed by Jaccard and IFP. Similarly IFP only relies on the matching of certain information within the instance descriptions thereby ignoring other information which is not defined by unique properties. This ensures that differences in information present within the instance description are ignored and that only a single value must be equivalent in separate instances to detect a match. The results confirm the intuition behind the strategies and their effects on precision - where the strict matches perform better - and recall - where measures which permit variation perform well.

Table 10.5: Accuracy of different permutations of Self-trained classifiers and feature similarity measures compared to baseline techniques

		Precision	Recall	F_1	F_2	F_3
Perceptron	Entailment	0.634	0.909	0.722	0.812	0.855
	IFP	0.619	0.807	0.699	0.693	0.739
	Jaccard	0.656	0.772	0.639	0.703	0.732
SVM	Entailment	0.610	0.905	0.727	0.801	0.847
	IFP	0.619	0.886	0.703	0.770	0.820
	Jaccard	0.747	0.674	0.695	0.684	0.646
Naive Bayes	Entailment	0.631	0.647	0.615	0.629	0.643
	IFP	0.639	0.653	0.544	0.579	0.608
	Jaccard	0.701	0.582	0.539	0.545	0.559
Baseline 1		0.213	0.999	0.330	0.519	0.663
Baseline 2		0.239	0.748	0.288	0.404	0.492
Baseline 3		0.765	0.725	0.719	0.703	0.697
Baseline 4		0.226	0.956	0.324	0.487	0.614

The results also demonstrate a variation between the performance of the machine learning classifiers. Perceptron and SVM achieve similar performance levels in terms of precision and recall, particularly when combined with Entailment as the feature similarity measure. Combining SVM with Entailment yields recall levels of 0.905 indicating that $\sim 90\%$ of *identity web references* are correctly retrieved using this permutation, similarly Perceptron yields 0.909 for recall, indicating that slightly more web references are identified using this method. SVMs learn a separating hyperplane within a vector space using optimization to enable the hyperplane to maximize the margin of separation between respective sets. SVM builds a model which does not overfit the training data - in this instance the social graph and generated negative examples - thereby leading to improved generalisation of new classifications, this is indicated by the high levels of recall achieved using this technique when combined with IFP.

The use of Naive Bayes as a classifier achieves lower levels for all evaluation measures than Perceptron and SVM. Building a generative model for classifying *identity web references* does not achieve performance levels to the same degree as discriminative models - highlighting the suitability of discriminative models over generative models for such a task. In the case of Perceptron and SVM a mapping function is learnt such that a given instance - the web resource to be classified in this case - is analysed and its inherent features are assigned weights, thereby allowing a value to be derived which denotes the membership of the instance to either of the two classes (*positive* or *negative*). In the case of generative classification models a given instance is analysed against a learnt density function describing the probability distribution of features, and as such generative models also take into account unobserved features. However in the context of applying such models to the problem setting addressed within this thesis, relying on only observed features - as Perceptron and SVM do - yields more accurate results. In particular this accounts for the ability of both discriminative models to classify new instances, denoted by the high levels of recall in Table 10.5 with respect to Naive Bayes.

The different permutations involving Naive Bayes yields differing performance levels which are comparable with the behaviour of Perceptron and SVM. For instance, Jaccard achieves high precision levels over the two alternative measures, similar to its performance

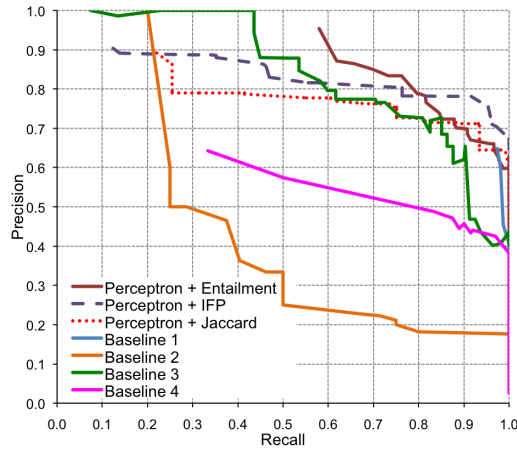


Figure 10.3: Precision/Recall curves for permutations of Self-trained Perceptron with differing feature similarity measures plotted against Baseline Techniques

when combined with Perceptron and SVM. IFP also achieves high levels of recall as it also does in the two alternative measures. The main limitation of Naive Bayes is its inability to generalise to new instances and therefore achieve high recall levels - given its comparable precision levels to the two alternative classifiers.

In terms of f-measure levels achieved as β is incremented, Table 10.5 indicates that where precision and recall are weighed equally, SVM and Entailment yields the optimum combination - outperforming all baselines techniques. As the weighting shifts to regard recall as being more important than precision - characterised by F_2 and F_3 scores - the optimum combination shifts to Perceptron with Entailment. Both the F_2 and F_3 scores in this case are found to be significantly different from human processing using the sign test ($p < 0.01$). This increase in weighting and the improvement of results provides an interesting insight into the effectiveness of humans at performing disambiguation with respect to automated techniques. As mentioned previously, the discovery of web resources containing personal information could have drastic consequences for the person which it describes (i.e., identity theft, discovery of sensitive personality traits, etc), thus making recall more important than precision. However, the results demonstrate the ineffectiveness of manual disambiguation in such a scenario. This therefore empirically establishes the utility of automated techniques, and self-trained classifiers in particular, over human processing of web content.

10.7.3.1 Perceptron

Analysis of the precision/recall curves produced when applying Self-training to a Perceptron classifier with differing feature similarity measures is shown in Figure 10.3. Perceptron combined with Entailment yields 0.7 for precision at recall 90%, a 0.15 increase over human processing. The two alternative feature similarity measures however produce higher levels of 0.712 and 0.781 for Jaccard and IFP respectively. As Entailment permits variations in features when building the input datasets for the classifiers it tends to reduce overfitting to the training data, this is evident at higher recall levels where precision degrades more sharply than both IFP and Jaccard.

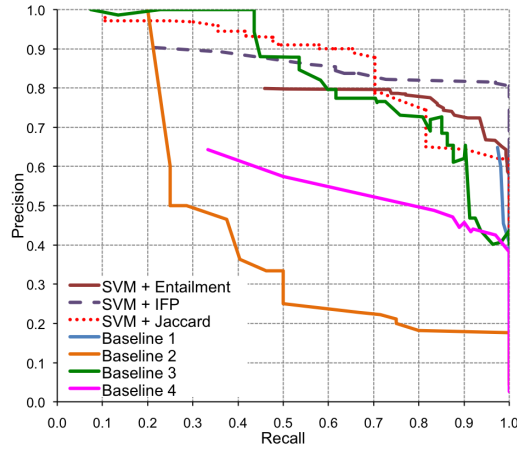


Figure 10.4: Precision/Recall curves for permutations of Self-trained SVM with differing feature similarity measures plotted against Baseline Techniques

10.7.3.2 SVM

SVM when combined with IFP performs similar to Perceptron, achieving 0.815 for precision at recall 90%, outperforming humans and all other baselines at this level of recall. SVM combined with Entailment produces 0.730 for precision at recall 90% and SVM with Jaccard yields 0.64 at the same recall level. At 100% recall, IFP maintains high levels of precision with a highest measure of 0.806 and comparing this to Baseline 1 (person name as positive classification) increases precision at recall 100% more than two-fold. For Jaccard combined with SVM, this permutation achieves 0.651 for recall at precision 90%, marginally outperforming Inference Rules at the same precision level (0.639).

10.7.3.3 Naive Bayes

Figure 10.5 presents the precision/recall curves achieved using different permutations of Naive Bayes and feature similarity measures. Similar to Perceptron and SVM, the combination of using a Naive Bayes classifier with IFP yields the highest level for precision at recall 90% of 0.591, outperforming the two alternative feature similarity measures. Human processing is more effective than Naive Bayes, particularly at higher precision levels, and when focussing on precision at recall 90%, all permutations are outperformed by humans.

10.7.3.4 Effects of Self-Training

Performing semi-supervised learning incorporates both labelled and unlabelled data into the decision process. The semi-supervised strategy explored within this thesis employs a Rocchio classification model to generate negative samples. A key requirement stipulated within chapter 9 of this thesis is for the selection of negative examples with caution, such that any false negatives do not reinforce themselves, and thereby lead to the misclassification of *identity web references* as not citing a given person - this is particularly important given the need to identify web resources containing possibly damaging personal information and the severe consequences of such information falling into the wrong hands.

Empirical assessments were therefore made of the performance of each self-trained clas-

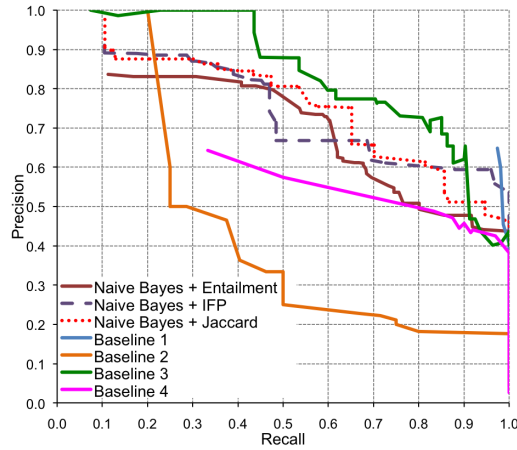


Figure 10.5: Precision/Recall curves for permutations of Self-trained Naive Bayes with differing feature similarity measures plotted against Baseline Techniques

sifier throughout its learning, and re-learning process. Evaluation was performed after each self-training iteration (i) against the gold standard to derive precision and recall values for the induced classifier. Figure 10.6 presents an overview of the precision/recall curves produced first by each classifier prior to training ($i = 0$) and then the same classifier, combined with a given feature similarity measure, once it has converged ($i = n$). As Figure 10.6 demonstrates, each classifier's curve moves away from the y-axis, essentially increasing the overall recall levels yielded by the classifier. For example, in the case of SVM combined with Jaccard as shown in Figure 10.6(b), there is a significant improvement in precision at recall level 90% produced by the classifier once converged in comparison within the same classifier with no training. The inclusion of unlabelled data within the learning process leads to the increase in recall levels, at the sake of precision as will be shown shortly, thereby leading to more web references being detected. This demonstrates the usefulness of incorporating unlabelled data into the disambiguation process, given that the initial provided seed data may only contain a portion of positive features, the rest of which are yet to be observed. To explore the effects of Self-training in closer detail, the following section now analyses the effects on each tested classifier in turn, examining the reduction in precision and the increase in recall which is produced.

Perceptron

Figure 10.7 presents the effects of Self-training on the Perceptron classifier when combined with different feature similarity measures. Maintaining the emphasis of prioritising recall over precision, Figure 10.7(a) shows the effects of Self-training on F_3 levels. The use of Entailment increases recall levels up to 1.0 before levels reduce and converge on a stationary value following the 29th self-training iteration. For all tested permutations with Perceptron precision decreases following training - as false positives are added to the training sets, however this is as a result of greater feature coverage over the positive set - characterised by the increase in recall levels. Comparison of the classifier's performance following the first training iteration against the converged performance levels, empirically demonstrates the improvement of Perceptron through self-training, using the strongest examples for set

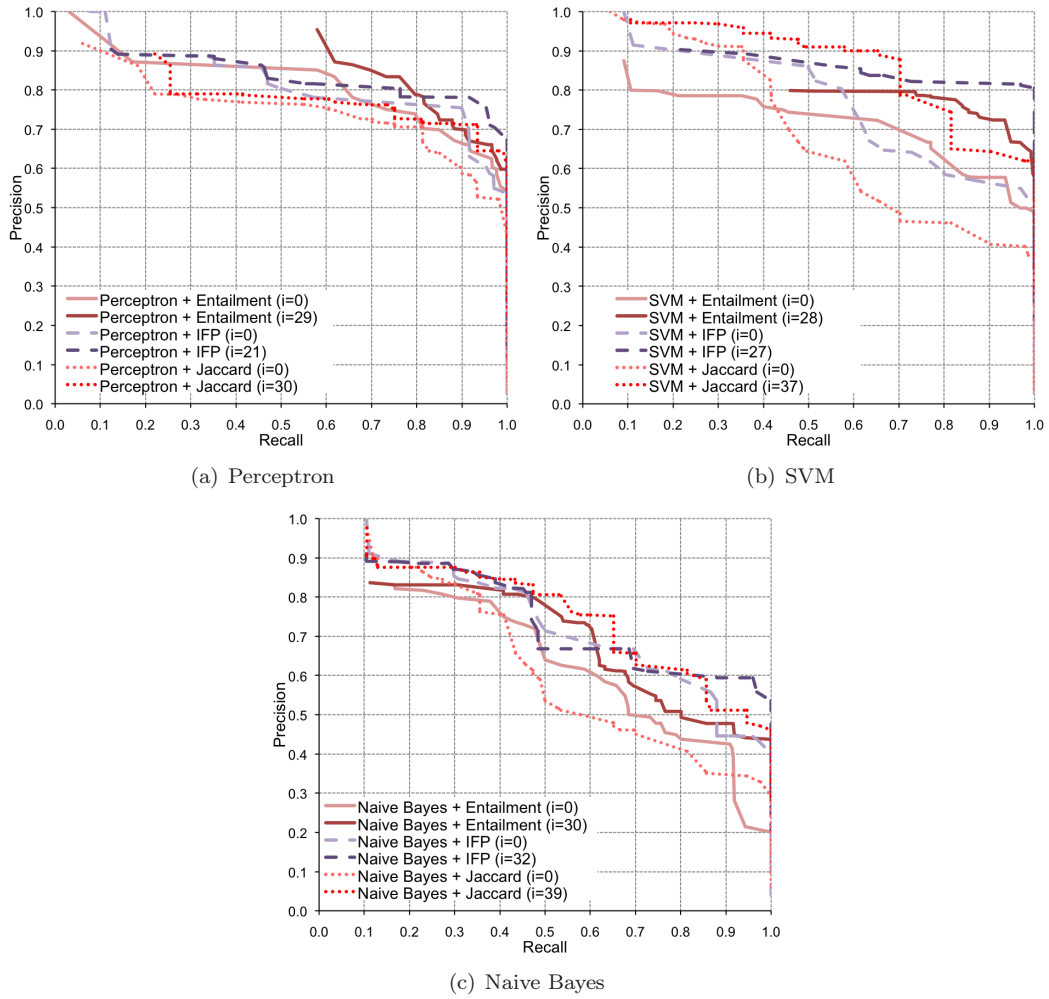
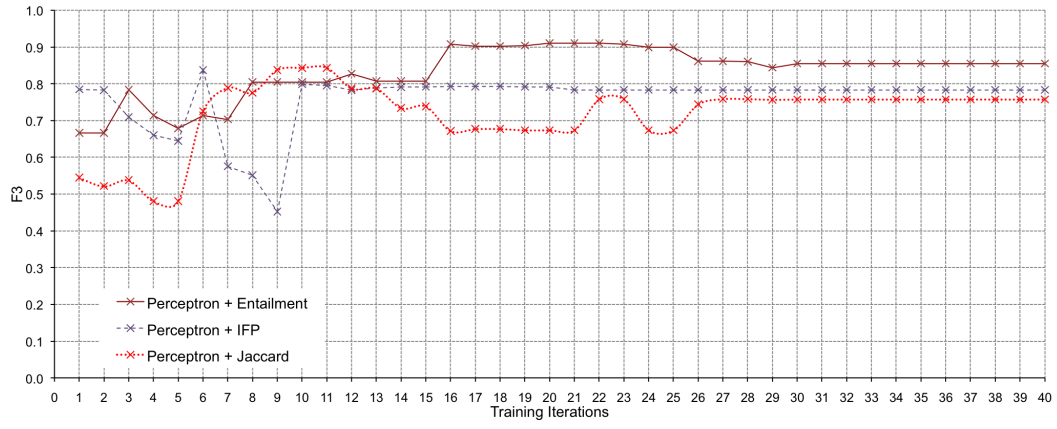


Figure 10.6: Precision/recall curves showing pre-training classifier ($i = 0$) against converged classifier ($i = n$)

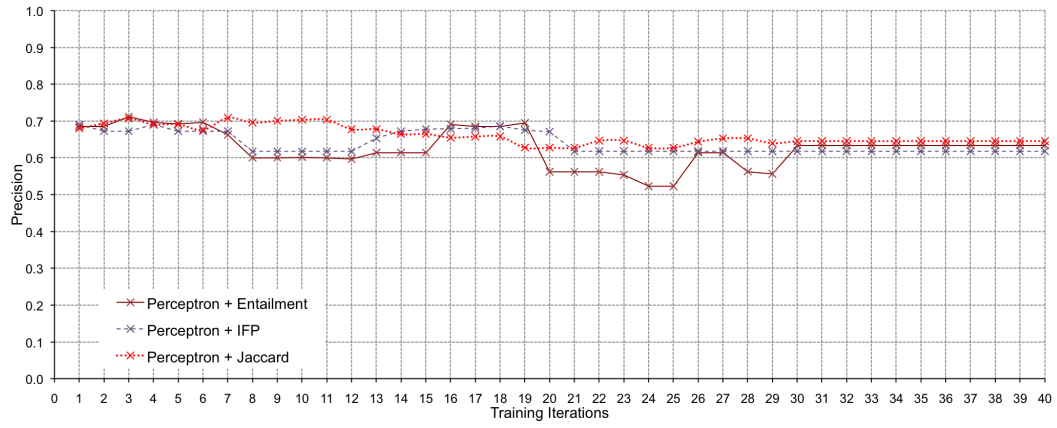
enlargement - full negative enlargement is discussed in the following section.

SVM

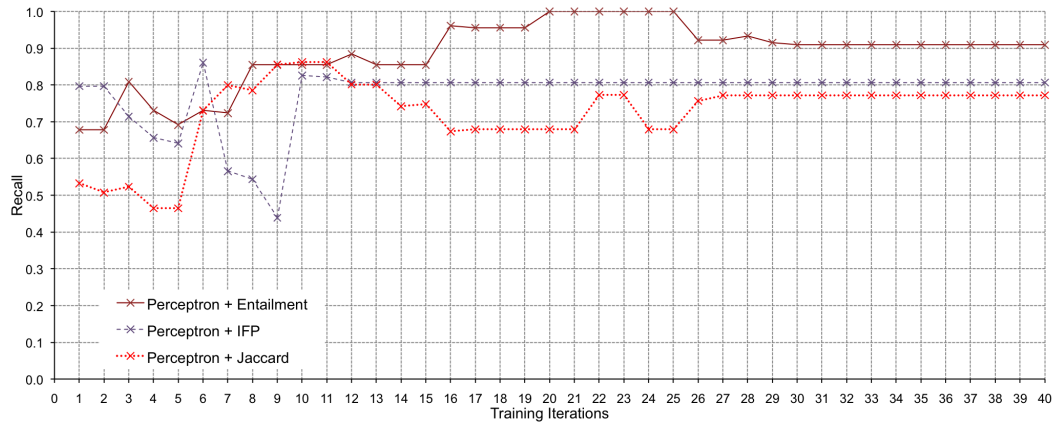
Like Perceptron, Self-training SVM leads to an increase in recall. However, the effects of training are not as drastic, or severe, as the effects of training on a Perceptron classifier. Utilising a supporting hyperplanes, SVM optimises the decision boundary to maximise the margin between support vectors from disparate sets, unlike Perceptron which chooses the first decision boundary which is found to separate the labelled datasets - using the Perceptron learning rule described in chapter 9. Figure 10.8 demonstrates the lack of oscillation occurring throughout the training process to the classifier, given the marginal alterations in both recall and precision levels between self-training iterations. Like Perceptron, SVM increases recall levels following Self-training, where the increases are more noticeable for the combinations of SVM with Entailment and IFP. Interesting the use of SVM with Jaccard is seen to increase precision levels throughout training, indicating that not only is feature coverage improved using this combination but also that there is a reduction in false positives produced as a result.



(a) F_3

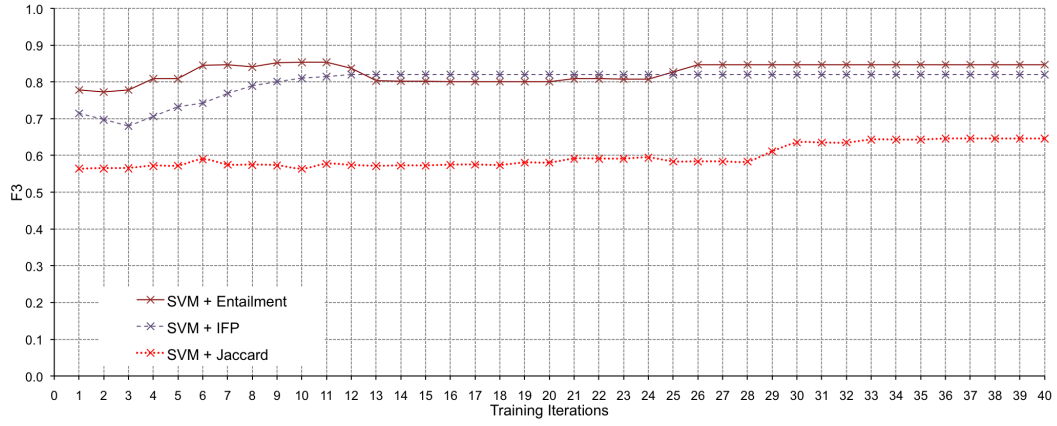


(b) Precision

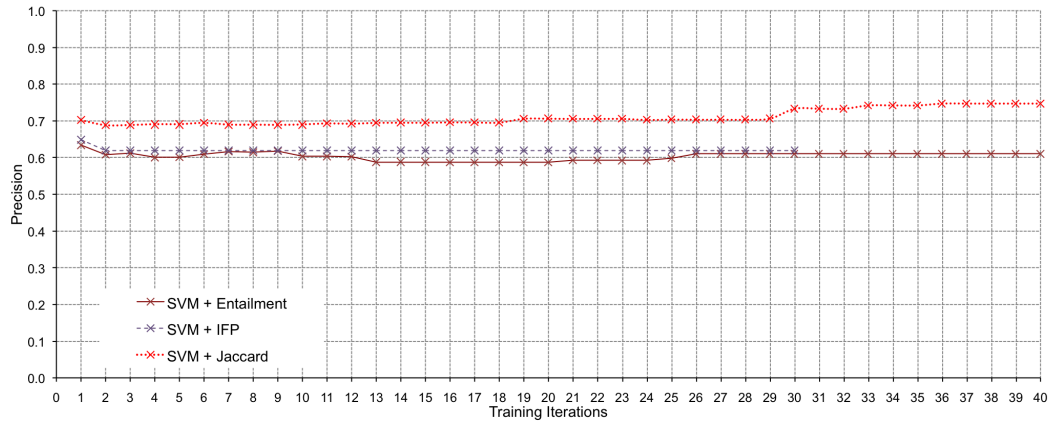


(c) Recall

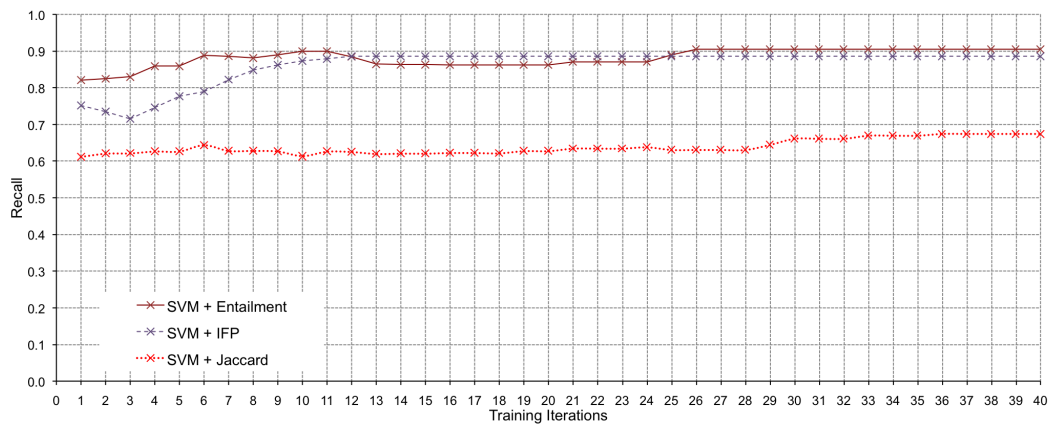
Figure 10.7: Effects of Self-training on different permutations of Perceptron and feature similarity measures



(a) F_3



(b) Precision



(c) Recall

Figure 10.8: Effects of Self-training on different permutations of SVM and feature similarity measures

Naive Bayes

Figure 10.9(b) demonstrates the reduction in precision for each of the self-trained permutations of Naive Bayes with differing feature similarity measures. The combination of Naive Bayes with IFP produces noticeable oscillations throughout the training process, shifting between varying values for precision before converging on a stable value after 30 self-training iterations. The increase in recall for the Naive Bayes classifier is not as significant as the increases produced when training Perceptron and SVM, Figure 10.9(a) shows how the F_3 level achieved for Naive Bayes with Entailment following the first training iteration is only marginally lower than the same permutation following convergence.

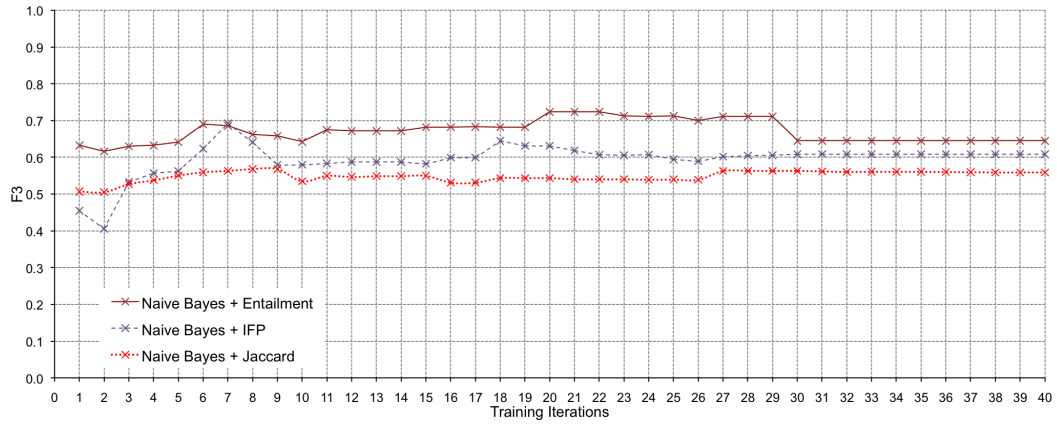
10.7.3.5 Effects of Negative Set Enlargement

The *growth size* of the training sets is controlled by the number of instances which are added to the training data following each iteration of the Self-training loop. In the context of this thesis, the growth size of the positive set is set to 1, and the growth size of the negative set is varied between two settings: 1 - where the strongest negative example is added to the negative set; and n - where all examples labelled as negative are added to the negative set. This contrasts two settings from the literature, where in [Mih04] it is argued that enlargement using the strongest instances yields the optimum results, and in [YHcC04] all negative examples are added to the negative training set. The results from experiments are now presented which contrast the difference in performance between enlarging the negative set using the strongest negative and all negative examples. For brevity, from hereafter enlarging the negative set with all negatives will be defined as FNE and enlarging the negative set with the strongest negative will be defined as SNE. Parameter tuning for Rocchio classification when generating negatives is explained in Appendix K, where the best performing parameters were selected from training over 10% of the dataset, and the same models were then run over the remaining 90% for testing.

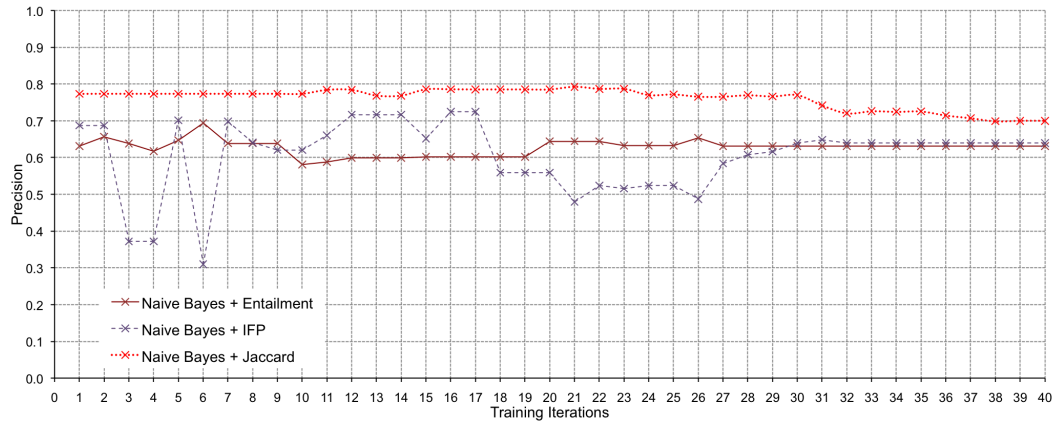
Table 10.6: Accuracy of different permutations of Self-trained classifiers using Full Negative Enlargement and feature similarity measures compared to baseline techniques

		Precision	Recall	F_1	F_2	F_3
Perceptron	Entailment	0.686	0.470	0.323	0.362	0.399
	IFP	0.774	0.485	0.300	0.321	0.316
	Jaccard	0.863	0.260	0.266	0.239	0.243
SVM	Entailment	0.724	0.616	0.447	0.485	0.530
	IFP	0.744	0.641	0.424	0.473	0.528
	Jaccard	0.755	0.335	0.337	0.313	0.318
Naive Bayes	Entailment	0.683	0.403	0.229	0.244	0.280
	IFP	0.705	0.329	0.248	0.245	0.265
	Jaccard	0.730	0.334	0.244	0.254	0.283
Baseline 1		0.213	0.999	0.330	0.519	0.663
Baseline 2		0.239	0.748	0.288	0.404	0.492
Baseline 3		0.765	0.725	0.719	0.703	0.697
Baseline 4		0.226	0.956	0.324	0.487	0.614

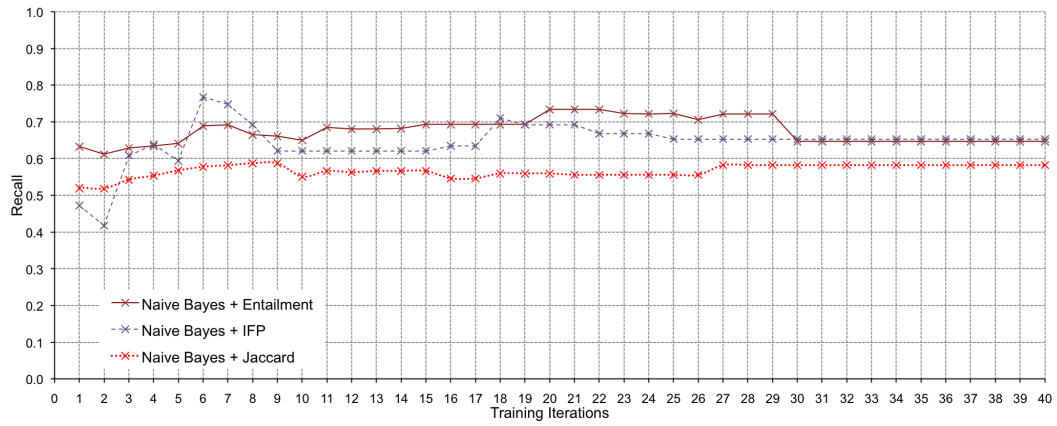
The results from experiments with full negative enlarged classifiers are shown in Table 10.6. There is a notable reduction in recall levels with respect to SNE classifiers, the results of which were shown previously in Table 10.5, and also an increase in precision levels.



(a) F_3



(b) Precision



(c) Recall

Figure 10.9: Effects of Self-training on different permutations of Naive Bayes and feature similarity measures

Enlargement of the negative set using all examples which are labelled as negative leads to the inclusion of false negatives. As a result future disambiguation decisions made by the trained classifier miss web references, and subsequently yield a reduced level of recall. Contextually this can be regarded in a scenario where a classifier is trained on the seed data leveraged from Social Web platforms, yet the features within this seed data may only capture a fragment of the person’s identity, leading to a lack of coverage of all possible discriminatory features - i.e., work colleagues, events attended with work, etc. Without primarily observing these features within the seed data, the classifier will misclassify *identity web references* as negatives. Enlargement of the negative set using all instances which are labelled as negative may lead to a reduction in computations - given that less instances have to be classified during the following Self-training iteration - yet recall levels are shown to reduce as a result.

Figure 10.10(a) presents the precision/recall curve produced when self-training a FNE Perceptron combined with differing feature similarity measures. In comparison with SNE Perceptron, as shown in Figure 10.3, FNE Perceptron yields poorer performance. Most notably the curve produced by SNE Perceptron with Entailment subsumes the curve produced by the same permutation for FNE - the same is also true for the permutations of Perceptron with IFP and Jaccard. Applying Self-training to FNE SVM shows a reduction in performance compared to SNE SVM. The differences are largely in the recall levels achieved when enlarging the negative set using the strongest instance, for instance in Figure 10.10(b) the curve produced by SVM with Entailment achieves precision at recall 90% of 0.241, whereas the same permutation when enlarging the negative set using the strongest negative instance yields 0.732. The same effects in performance are evident for FNE Naive Bayes, as shown where precision at recall 90% is significantly lower than permutations of SNE Naive Bayes.

The precision/recall curves shown in Figures 10.10(a), 10.10(b) and 10.10(c) and the results within Table 10.6 demonstrate the effects of enlarging the negative training set with all negative labelled instances. The Rocchio classification model used to generate negative examples does not guarantee the generation of negative instances to support the already provided positive seed data. Given that the collected seed data is often limited in size, approximation of negatives using Rocchio can produce false negatives - reflected by the limited influence which the known positives can have over the prototype vectors. Therefore by enlarging the negative set with all negative examples, false negatives are produced that contain features which are yet to be observed as belonging to the positive class. The results from this exploration show that where initial positive data is limited in size, enlarging the negative set with the strongest classification yields better performance - ensuring that high levels of precision are achieved for high recall levels.

10.8 Comparing Information Representations: Metadata vs Bag-of-words Models

The disambiguation techniques presented within this thesis use metadata model representations of web resources, where the process of constructing such models involves information extraction and metadata processing - both of which are computationally expensive. Conversely existing approaches [CDF⁺98, Coh02] employ a shallow approach to web resource feature representation, for instance by building a bag-of-words representation of a web re-

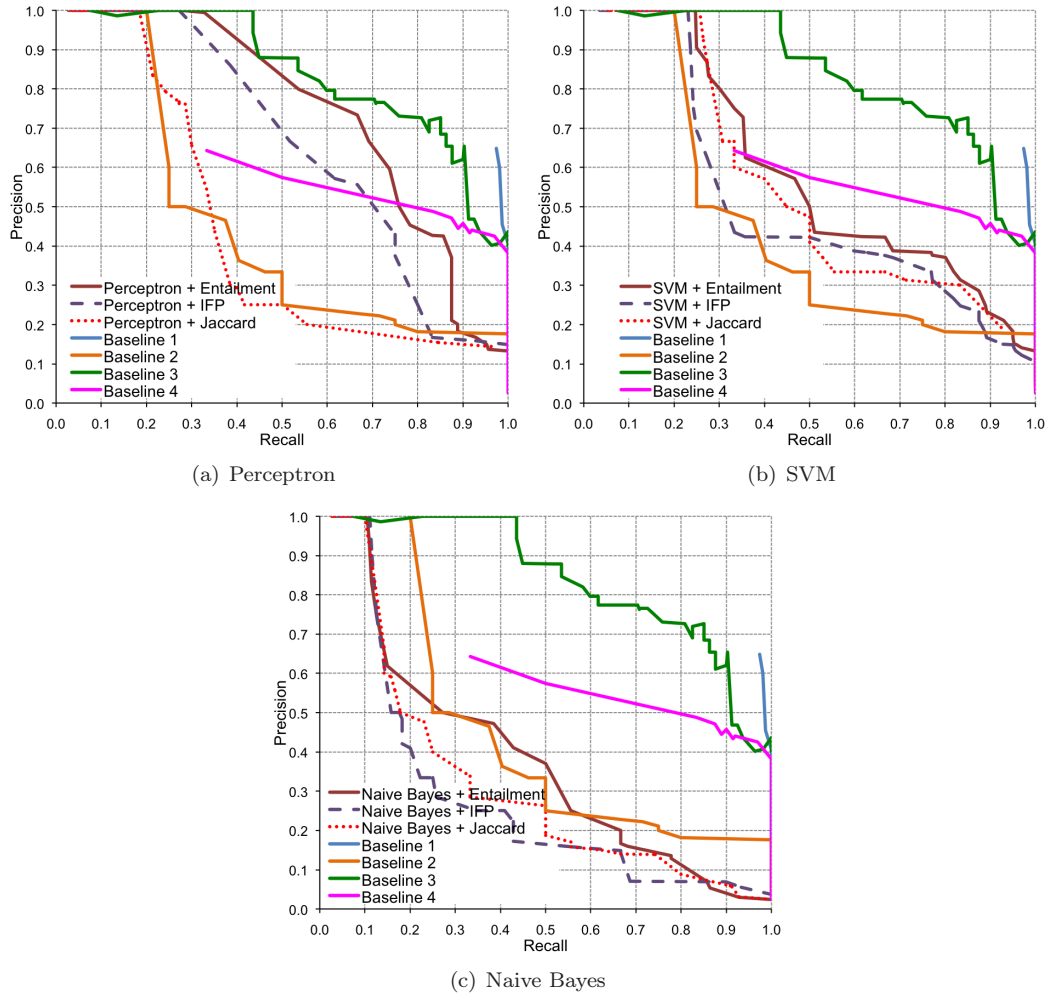


Figure 10.10: Precision/Recall curves for classifiers when enlarging the negative set with all negative samples after each training iteration

source and training a classifier from those features. To contrast the explored methods within this thesis, the Yarowsky algorithm - explained in chapter 9 - is provided with the same seed data as the explored techniques, and the same collection of web resources to be disambiguated for each participant. The algorithm uses the same classifiers as Self-training in its *inner* loop: Perceptron, SVM and Naive Bayes, however the feature representation used by each classifier differs by using a bag-of-words model, free of any semantic markup or processing into RDF.

Figure 10.11 presents an overview of the approach used to produce the bag-of-words model and train the classifiers using the Yarowsky algorithm from Algorithm 7. For each participant tested, the social graph exported from Social Web platforms is provided as input along with the collection of web resources which are to be disambiguated. The following SPARQL query is used to extract the literals and resources from the social graph, thereby producing a collection of string literals and hyperlinks from that document:

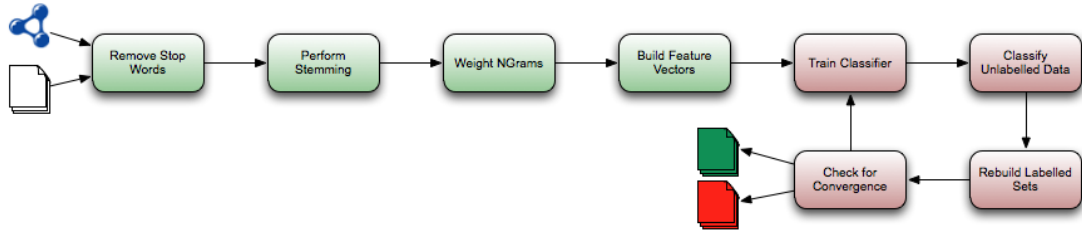


Figure 10.11: Pipeline architecture of the Yarowsky implementation

```

PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX foaf:<http://xmlns.com/foaf/0.1/>
SELECT ?y
WHERE {
    ?p rdf:type foaf:Person .
    ?p ?x ?y
}

```

Any web resource linked to the social graph owner also has a bag-of-words representation compiled from its inherent features using the same process by which the bag-of-words representation for each of the possible web citations are compiled. This functions as follows: first all HTML tags are removed from the web resources - preserving any links contained within the *href* attribute of a *<a>* element. Second all stop words are removed from the remaining content and the remaining words are then stemmed using the Porter Stemming algorithm [SJW97]. This produces the bag-of-words model for both the seed data - containing both the exported social graph and any web resource linked to the person in the social graph - and the collection of possible web references. Feature vectors are then compiled from each bag-of-words using unigrams as features and weighting the features based on TF/IDF weighting.

The Yarowsky algorithm takes the feature vectors as input and uses its *outer* loop to iteratively train the *inner* loop - a *supervised* classifier. The disambiguation process continues by retraining the classifier until convergence - where the classification labels from the previous training iteration are equivalent to the current round. The *outer* loop is then exited and the labelled web resources returned. As mentioned the interesting quality of this baseline is that the classifiers tested are equivalent to those used within the Self-training approach, while the algorithm and shallow feature representation differs. Rocchio classification was used to generate negative instances from which the bag-of-words classifiers were then trained, selection of ρ was performed using tuning over the 10% training split of the dataset, the results of which are shown in Appendix L.

Table 10.7 presents the results from the experiments using the bag-of-words classifiers. Recall levels are found to be comparable to that of same classifiers trained using metadata models, where Perceptron and SVM achieve higher levels of recall than Naive Bayes. Precision however is found to be significantly lower than the classifiers trained using metadata models. Incrementing β for f-measure indicates that as recall increases in importance, scores increase but are found to be significantly lower - using the sign test ($p < 0.01$) - than human processing for all tested measures. The use of an outlier detection method, using RDF instances from the metadata models, also outperforms all of the tested classifiers in terms of F_3 levels. Each of the tested classifiers are now examined in greater detail in terms of their

precision/recall curves.

Table 10.7: Accuracy of classifiers trained using Yarowky algorithm and a Bag-of-words model

	Precision	Recall	F_1	F_2	F_3
Perceptron	0.206	0.890	0.285	0.417	0.527
SVM	0.235	0.866	0.296	0.420	0.523
Naive Bayes	0.274	0.709	0.282	0.371	0.450
Baseline 1	0.213	0.999	0.330	0.519	0.663
Baseline 2	0.239	0.748	0.288	0.404	0.492
Baseline 3	0.765	0.725	0.719	0.703	0.697
Baseline 4	0.226	0.956	0.324	0.487	0.614

10.8.1 Perceptron

Figure 10.12(a) compares self-trained Perceptron using metadata model representations of web resources against a bag-of-words model in terms of precision/recall curves. The figure demonstrates that each of the tested permutations of Perceptron with feature similarity measure outperforms the bag-of-words model. Focussing on precision levels at recall 90%, one can see that precision levels achieved through using metadata models are almost two-fold greater than that of the bag-of-words model.

10.8.2 SVM

The precision/recall curves produced using self-trained SVM with metadata models show an improvement in performance over the same classifier trained using the Yarowsky algorithm and a bag-of-words model. Figure 10.12(b) shows how each of the permutations of SVM with feature similarity measures outperforms SVM using a bag-of-words model, in particular for precision at recall 90%. For recall at precision 70% the permutation of SVM with IFP yields 1.0, compared to 0.220 for SVM using a bag-of-words, other permutations also achieve higher recall at this precision level: 0.735 for SVM and Entailment and 0.651 for SVM with Jaccard.

10.8.3 Naive Bayes

Comparing the Naive Bayes classifier trained using a bag-of-words model against the previously discussed Self-training permutations, each of which use metadata models, indicates that at high recall levels metadata models yield higher levels of precision than the bag-of-words classifier, as presented in Figure 10.12(c) by the improvement in precision at recall 90% by the self-trained permutations of Naive Bayes. However, for recall at precision level 90% the use of a bag-of-words models shows similar precision levels as the rich semantic markup used by metadata models.

Overall the results from Table 10.7 and the precision/recall curves in Figures 10.12(a), 10.12(b) and 10.12(c) demonstrate the increased performance achieved through the use of metadata models by the tested classifiers. Using a bag-of-words representation of a web resource does not require the computationally expensive processes described within chapter 6 of this thesis, where features must be identified within HTML documents and extracted

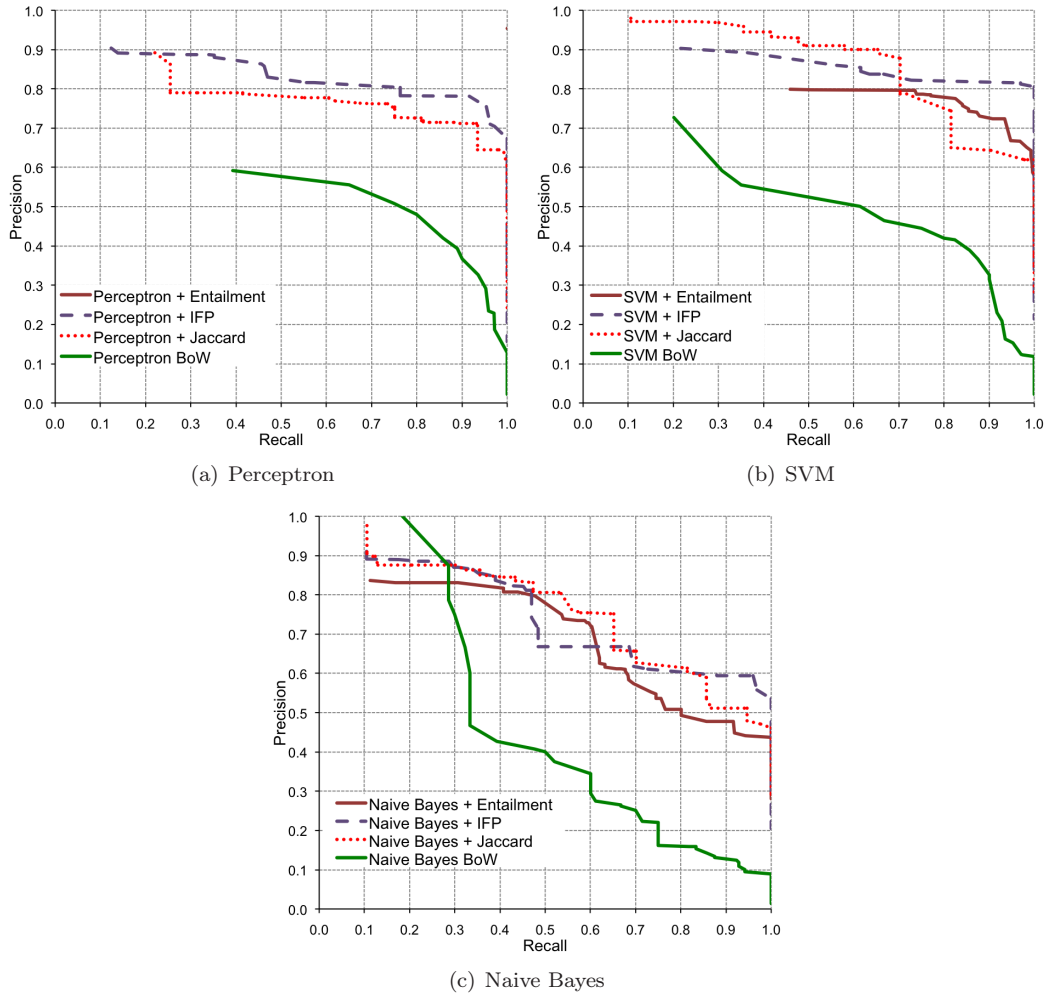


Figure 10.12: Precision/Recall curves for Classifiers using Metadata Models compared with a Bag-of-Words Model

with high levels of both precision and recall, however the use of a bag-of-words model leads to a noticeable reduction in precision levels in comparison with metadata models. Information found on the Web is, in general, noisy. Web pages do not conform to a strict formalisation and often contain words and terms which are ambiguous or included to bias search engine rankings (i.e., hidden content in blank frames containing an abundance of keywords). By performing information extraction from web resources, features are compiled which are relevant to the task at hand (e.g., person attributes). This process performs data cleaning of the web resources, reducing the possibility of features being included which could wrongly bias classification decisions. This is evident if one considers the difference in precision at recall 90% achieved by the Self-trained classifiers - which utilise metadata models - against the same bag-of-words model.

The use of metadata models to represent the underlying knowledge structure of web resources also enables the novel application of the techniques presented in chapters 7 and 8. Inferences Rules are applied using SPARQL over metadata models to infer web citations, matching the graph patterns to associate information within the seed data with information found within a given web resource. Random Walks are performed over a graph-space com-

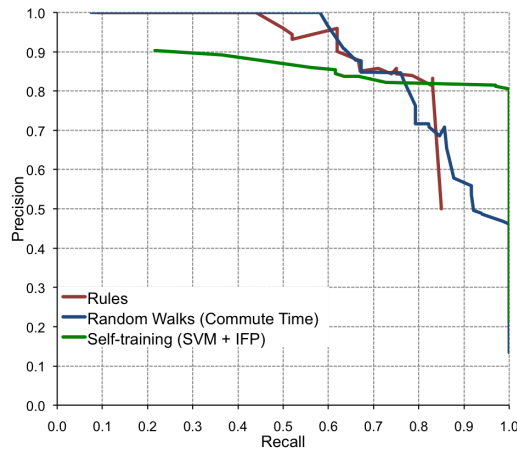


Figure 10.13: Precision/Recall curves for Rules, Random Walks using Commute Time distances and Self-trained SVM with IFP

piled from metadata models - utilising the graph-like structure of RDF - and weighting edges according to their ontological properties. At the time of writing this thesis, neither of these techniques have been applied in such a setting, and without representing web resources as metadata models these techniques would be unable to function in such a manner. Furthermore given the cited increase [MMZ09] in the use of metadata within web documents, future Web technologies are likely to allow metadata models to be gleaned automatically from web documents. Therefore techniques are required which are able to cope with such models, where shallow document representations lacking machine-readability will not be required.

10.9 Comparison of Disambiguation Techniques

Each of the explored disambiguation techniques yields high levels of either precision or recall. Inference Rules using their precise graph pattern matching technique achieves high precision ensuring that almost all of the web resources labelled as *identity web references* are correct. Random Walks using Commute Time distances achieves high levels of recall, however precision is found to be lower than Rules. Self-trained classifiers achieve higher recall than both Rules and Random Walks but with comparable precision levels to the latter method. To analyse the performance of each of these disambiguation techniques more closely, the optimum permutation of each technique was chosen and plotted on the same precision/recall graph, as shown in Figure 10.13 - the optimum permutation was selected based on the model optimising precision at recall 90% (similar to work by [RYGR10]).

Figure 10.13 shows the difference between the three methods. Where precision is considered as being more important than recall, for instance in a scenario where information is needed quickly and false positives must be minimised, then the use of Rules and Random Walks using Commute Time distances are well suited. The precision/recall curves show that for recall at precision 90% Inference Rules and Commute Time distances achieve comparable levels, with Random Walks with Commute Time distances achieving slightly higher recall levels due to the inclusion of unlabelled data within the disambiguation process, unlike Rules which uses only labelled data. Overall, evaluation of the presented disambiguation

techniques demonstrates the increased levels of recall achieved using semi-supervised techniques in comparison with the explored supervised technique, this is also evident in Figure 10.6 when the precision/recall curve of each tested classifier prior to Self-training is plotted against the same classifier following its convergence on a stationary hypothesis following training. In each of the tested cases, there is a noticeable improvement in recall, indicating the classifiers' incorporation of previously unknown features into their disambiguation processes.

The central motivation behind the techniques explored within this thesis is to provide an automated solution to identify web resources containing a given person's personal information, following which the individual would be alerted to the presence of such information, enabling them to take the correct action. This places a greater emphasis on recall than precision, given that the concern surrounding the identification of certain identity traits could have detrimental effects on the individual. When considering this context, and focussing on precision at recall 90%, figure 10.13 demonstrates how applying Self-training to SVM, with IFP as a feature similarity measure, achieves high precision levels, outperforming all of the tested baselines and other methods.

10.10 Summary

This chapter has evaluated each of the presented disambiguation techniques from chapters 7, 8 and 9. The findings show that each technique achieves high levels of either precision or recall, and are therefore better suited to differing settings and disambiguation needs. For instance where precision is more important than recall, Rules and Random Walks are better suited, whereas in situations where recall is considered more important than precision, Self-training is a more suitable technique. Evaluation against a number of different baselines provides empirical evidence of the high performance levels achieved by the techniques explored within this thesis. In particular the use of a baseline of human processing provides empirical evidence to justify one of the central claims of this thesis in that automated techniques are able to alleviate the job of humans. The strength of this thesis is evident in the analysis of precision at recall 90% where the application of Self-trained discriminative classifiers outperforms the human processing baseline, and also in terms F_3 scores, which are found to be statistically significant using the sign test ($p < 0.01$).

The following chapter presents a concluding discussion of the work presented within this thesis. It relates the approach presented in chapter 4, its various stages and the explored disambiguation techniques, back to the original motivation in chapter: the need to identify web citations in order to avoid identity theft and protect against lateral surveillance. In chapter 12 an approach is presented which is planned for future work, linking the approach presented within chapter 4 to the context of automatic risk assessment based on the presence of person attributes and the affects which the visibility of such information could have on an individual.

Part III

Conclusions

“I am looking at myself, at him, and I am absolutely certain that he, with his ruler straight eyebrows, is a stranger, somebody else, I just met him for the first time in my life. And I’m the real one. I AM NOT HIM.”

From “We” by Yevgeny Zamyatin

Chapter 11

Conclusions

11.1 Introduction

This thesis has explored automated techniques for disambiguating *identity web references* and how social data can be used to support such techniques. Chapter 2 presented an overview of the Social Web and digital identity. Chapter 3 reviewed current state of the art Semantic Web technologies, how such technologies can be used for knowledge representation and how the semantics of digital identity information can be described. Chapter 4 presented the architecture and design of a user-centric approach to identify web references by finding possible web citations and then applying the disambiguation techniques supported by social data to decide which resources referred to a given person. Chapter 5 described methodologies for collecting seed data by reconciling digital identity fragments distributed across the Social Web. Chapter 6 presented methods for generating metadata models representing the underlying knowledge structure of a web resource and how these models can be transformed to use consistent semantics with the provided seed data. Chapters 7, 8 and 9 each presented contrasting disambiguation techniques supported by digital identity information leveraged from Social Web platforms. Chapter 10 presented an evaluation of these disambiguation techniques using a real world dataset.

This chapter describes the conclusions learnt from the research presented within this thesis. Section 11.2 analyses the research questions and claims made in this thesis from chapter 1. Section 11.3 presents an analysis of the presented methodologies and approaches with respect to the various requirements set forth. Section 11.4 finishes the chapter with a summary of the conclusions learnt from this work.

11.2 Research Conclusions

11.2.1 Automated Web Reference Identification

Referring back to the motivations discussed within chapter 1 of this thesis based on the need to identify *identity web references*, this process is divided into two stages: filtering - searching the Web for possible citations, and vetting (disambiguation) - producing a unary set of *identity web references* which cite a given person. The techniques explored within this thesis concentrate on the latter of these stages, and through evaluation have been empirical

demonstrated as outperforming humans at the same disambiguation task. Mapping these findings back into the motivation of chapter 1: the need to detect web resources containing sensitive personal information, one can conclude that the presented techniques provide a viable solution to alleviate human processing of web content. The approach presented in chapter 4 attempts to replicate the entire cognitive process of web citation identification, by searching the Web using different web gateways - search engines including Google and Yahoo, and Sindice together with Watson on the Semantic Web - to gather a broad collection of possible web citations for a specific person. These web resources are then processed into a machine-readable form - representing the underlying knowledge structure of the web resources using RDF - before passing the possible web citations onto a given disambiguation technique for analysis.

In terms of preventing identity theft and protecting web users from lateral surveillance, the disambiguation techniques provide the means by which a web user can accurately (in terms of both precision and recall) retrieve those sources on the Web where their personal information appears. The techniques do not attempt to assess the risk posed to a given individual, this is instead planned for future work, a proposed approach to which is described in §12.2 of the following chapter. Instead the explored techniques provide the means to identify web citations so that the correct action can then be taken by the individual which they cite: i.e., removing the information, altering its visibility, enquiring how the information came to appear there by contacting the host of the web site. The last of these possible actions has recently become particular endemic on the Web, as person search engines become increasingly sophisticated and improve in their ability to accurately gather web references together in a central location, in short facilitating the collation of personal information for its reuse.

Although the motivation for exploring automated disambiguation within this thesis is based around people and their exposure on the Web, the techniques are not limited to this solitary domain. Each of the presented techniques could be adapted for application in alternative domains, thereby performing *entity disambiguation*. The approach presented in chapter 4 enables deployment in alternative domains, given the modular nature of its architecture, where modifications would be made to the approach's various stages: for instance, replacing the collection of seed data from Social Web platforms with some other means of producing seed data. Consider that the domain of application is no longer disambiguation of web references of people, but is instead the disambiguation of web references for companies or organisations - this is pertinent given the ambiguous names of companies. Seed data could be the list of contacts, employees, past contracts, and location information about a given organisation. The approach would function in the same manner when collecting possible web citations: searching the Web using the provided company's name - via a broad search across many web search engines. Compilation of metadata models for each of the returned web resources would require adaptation of the information extraction methods described in chapter 6 - given that the method is trained for the extraction of person attributes - where attributes for extraction would be defined and the HMM trained on the new domain.

The compilation of both the seed data and the collection of possible web references, each represented using metadata models, provides the disambiguation techniques with information represented using consistent semantics. Both Random Walks and Self-training were implemented in a generalised and non-domain specific form, where the former technique

builds a graph-space from the metadata models provided as input, weights distances according to the ontological characteristics of the edges and then clusters web resources based on distance measures. The latter technique, Self-training, uses the seed data, and any linked web resource to the URI of the entity, to first, generate negative training data, and then, train a classifier using the seed data and created negative data. Rules however would have to be adapted for alternative scenarios, given that the strategy used in Algorithm 5 includes triples containing concepts which are specific to identity ontologies (e.g, *foaf:name*). The rules produced using this approach are therefore domain specific, and would require either adaptation to an alternative domain, in which case the fundamentals of Algorithm 5 would have to be altered, or a suitable approach to rule generalisation could be implemented.

The applicability of the explored disambiguation techniques, together with the general architecture of the approach from chapter 4, is therefore not-domain specific. Utilising these techniques in other domains would require fresh training of the models using newly acquired seed data - specific to the domain of use - however the internal methodology of the presented techniques would require little (in the case of Inference Rules) or no alteration. Other domains where this approach is applicable includes reputation management on the Web, an increasingly important domain according to [MS10], where web references would be identified - for example on Social Media platforms such as Blog sites - and the information present assessed in terms of the sentiment of the citation. Performing this process over the entire Web would enable companies and organisations to keep track of how a market responds to their product, and the reputation it currently has online.

11.2.2 Analysis of Research Questions

The introduction of this thesis described a central key question which has been explored within this body of work:

How can identity web references be disambiguated automatically?

In answering this question research has been conducted into a range of fields through which novel means of automating the disambiguation process have been discovered. The underlying process by which disambiguation is performed is to process a given set of web resources into those which cite a person and those which do not, thereby generating a unary set of web citations for a given person. One of the underlying themes that was discovered in addressing this question was the need for consistency in the interpretation of information from disparate sources. The World Wide Web is made up of a heterogeneous collection of web documents described through a plethora of formats and representations. Interpreting information within such documents requires a common understanding of the meaning of the documents' inherent information. At a low-level this can be contextualised as knowing that a piece of information in a given web resource denotes the name of a person. To this end, the work explored in this thesis has been based on the foundations of semantics. The use of Semantic Web technologies, such as the Resource Description Framework, has provided a vital knowledge representation format, coupling this representation with web accessible ontologies allows information contained in distinct web documents to be interpreted by automated disambiguation techniques.

The central research question guiding the work presented within this thesis was broken down into several distinct research questions, outlined in chapter 1. The research presented

in this thesis has explored these research questions as follows:

1. *Is it possible to define automated techniques that alleviate the need for human processing?*

The cognitive process by which personal information is monitored on the Web forms a two-stage process: first candidate web resources are gathered which may cite a given person and then the set of web resources are analysed to identify which do cite the person. This cognitive process has been mimicked by the approach described in chapter 4, which collects seed data describing the digital identity of a given person, collects possible web references - and converts the information from those resources into a common format with the seed data - and uses the seed data to support disambiguation techniques. Throughout the approach each aspect of the pipeline has been evaluated extensively and shown to achieve high levels of performance, either by minimising the number of mistakes when aggregating social graphs or achieving high levels of precision and recall when generating metadata models from HTML documents. The evaluation from chapter 10 presents the culmination of these high performance levels, achieved by different techniques in the different stages, resulting in F_3 scores which surpass human processing for two of the presented disambiguation techniques.

The baseline of human processing used within the evaluation of the techniques provides an indication of the level of performance gauged by humans when performing a similar disambiguation task. Another suitable baseline measure would be for each evaluation participant to discover their own web references from a given set of web resources and then comparing the accuracy of the automated techniques against such findings. This is planned for future work, as the work presented within this thesis forms the basis of a deployed identity management system designed to alleviate the current manual process by which web users detect those web resources which cite them on the World Wide Web, and enable the automated assessment of the risk posed to those users based on the presence, nature and visibility of certain pieces of information - this approach is discussed as a proposal in the following chapter.

Emulation of the approach by which humans identify their *identity web references* - first discovering possible citations and then performing disambiguation to detect definite citations - has facilitated the exploration of techniques which are comparable in performance to humans when performing the same task, and for certain disambiguation techniques has been shown to surpass humans, especially in terms of recall. Moreover the suitability of automated techniques is clear to see when one considers the rate and enormous scale of information publication on the World Wide Web.

2. *Is it possible to define automated techniques that require minimal supervision and only the provision of initial seed data?*

This thesis has presented three disambiguation techniques, each utilising social data in a distinct manner: Inference Rules (chapter 7) use the seed data, given that this is modelled using RDF, to populate a data base from which a rule base is then constructed. The strategy for constructing this rule base utilises RDF instances and the semantics of information within the instance descriptions. These rules are then applied to the knowledge structures

of web resources to infer an *identity web reference*. Random Walks (chapter 8) uses the social data to construct the initial graph-space, given that this data is modelled using RDF which has a graph-like structure, and then compile a larger graph-space by adding the web resources which are to be disambiguated (see §8.4.1). Distances between nodes are measured within the graph space and the web resources which take the least time to reach from the social graph are clustered together. Self-training (chapter 9) uses the social data as positive training data to first generate negative training data using Rocchio classification, given that the disambiguation task is a binary classification problem, and then uses the training sets to learn an initial machine learning classifier. This classifier is then iteratively retrained and applied to the web resources to be disambiguated. In each technique no supervision is required, only the seed data is needed from which the disambiguation technique can then begin its process.

3. *Can seed data be gathered or constructed inexpensively?*

One of the hard problems of using *seeded* techniques (*supervised* and *semi-supervised*) for disambiguation is the generation of seed data, which is often expensive to produce manually. Instead this thesis explores the use of data leveraged from the Social Web to support such techniques. Chapter 2 described how digital identity information is distributed across several Social Web platforms where each platform contained a different piece, or facet, of a given person's identity. Such platforms restrict data access, thereby reducing data portability. Work presented within chapter 5 addressed the issues of identity fragmentation and data portability, and presented methods to generate RDF models from data within distinct Social Web platforms, where each model describes a facet of a person's digital identity within a given platform. This social data is then accurately gathered together by combining RDF models from separate platforms using a Graph Reasoning method which is supported by semantics used to describe information within the models. To gauge the accuracy of this Graph Reasoning method an evaluation was performed against two state of the art methods (§5.7). The results demonstrated that graph reasoning outperforms the state of the art methods thereby providing a novel means of accurately and effectively gathering seed data together from multiple Social Web platforms.

4. *How can information distributed across the World Wide Web and found in heterogeneous sources be interpreted by automated techniques?*

Semantic Web technologies have facilitated the interpretation of information from distinct sources. Chapter 5 presented techniques to leverage data from Social Web platforms as machine-processable RDF. Chapter 6 presented various methods for generating RDF models representing the underlying knowledge structure of web resources. In each case ontologies are used within the models to provide consistent metadata descriptions. By using consistent semantics a common understanding is achieved of the meaning of information from heterogeneous sources. This common interpretation therefore facilitates the integration of information - as shown in chapter 5 with the aforementioned Graph Reasoning method - and the ability to process information - as demonstrated in the chapters exploring various disambiguation techniques.

11.2.3 Analysis of Claims

Chapter 10 presented the evaluation of the automated disambiguation techniques from chapters 7, 8 and 9. The results from this evaluation have presented empirical evidence that under certain retrieval measures human processing is outperformed by the explored disambiguation techniques. For instance, disambiguation using Inference Rules presented in chapter 7 outperforms human processing in terms of precision but fails to outperform human processing in terms of recall. This indicates that a rule-based technique returns results which are accurate but fails to return all *identity web references*. The graph-based technique - referred to as Random Walks - explored within chapter 8 achieves higher recall than human processing while failing to exceed humans in terms of precision. The Self-training technique from chapter 9 explored several permutations of machine learning classifiers and feature similarity measures (defined as distinct RDF graph comparison techniques). Of the tested permutations, five permutations were found to outperform humans in terms of recall. Analysis precision/recall curves in chapter indicates the effectiveness of Self-trained classifiers in achieving higher values for precision at recall 90%, particularly for SVM and Perceptron combined with IFP.

- *Automated disambiguation techniques alleviate the need for human processing*

One can conclude that the explored disambiguation techniques surpass the performance of humans for several accuracy measures, and overall the use of Self-training as a learning strategy from which a classifier can be trained for the task of disambiguation yields the most promising results. Although Inference Rules are precise they fail to achieve high levels of recall, which within the context of this thesis makes the technique limited in its application - given the need to maximise the number of web references found over the the precision of the returned results. Random Walks sacrifices precision for recall and thereby achieves higher overall accuracy (in terms of F_1 , F_2 and F_3 scores) than Inference Rules. Self-training presents the only technique which is able to achieve high precision levels whilst also returning the maximum number of *identity web references* (defined by high recall levels in the results).

Within the introduction of this thesis the problem setting is described which web users are now faced with when detecting web resources which contain their personal information. The scale of the World Wide Web and the amount of information to be processed further compounds the complexity of this setting, such that manually handling and processing data on a regular basis is simply not feasible. The observations made within chapter 10 show the tendency of humans to miss *identity web references* in comparison with automated disambiguation techniques - emphasised by the lower recall levels. Applying the f-measure evaluation metric and increasing β empirically demonstrates the increase in performance of automating the disambiguation process over manual processing - as for both Random Walks and Self-training F_3 levels are higher than that of humans, where the differences are shown to be statistically significant using the sign test ($p < 0.01$).

- *Disambiguation techniques are aided by a semantic representation of web resources*

One of the central contributions and themes of this thesis is the application of Semantic Web technologies to represent information found within disparate web resources. Using rich semantic markup has enabled information integration in chapter 5 - where the heterogeneity

of XML schemas used by differing Social Web platforms was overcome through the exportation of social graphs using RDF and the FOAF and Geonames ontologies; and has supported disambiguation methods, in particular the use of SPARQL rules for inferring *identity web references* by matching triple patterns associating seed data with information found in web resources.

Existing web document classification approaches such as those described in [CDF⁺98, Coh02] have performed similar tasks without the process of constructing metadata model representations of web resources - and thereby avoiding the computationally expensive information extraction stage described in chapter 6. In such approaches a shallow approach to representing web documents is utilised that employs a bag-of-words model by stripping away any HTML markup in the document, removing stop words, and reducing the content found in the resource to keywords. Chapter 10 compared the same classifiers used within the Self-training technique documented in chapter 9: Perceptron, SVM and Naive Bayes, when provided with a shallow representation - i.e., a bag-of-words model, against metadata models. The findings indicate a significant improvement in performance when classifiers are trained using metadata models, in particular, precision levels are shown improve when metadata models are used.

One of the notable contributions of this thesis has been the novel use of data leveraged from the Social Web for use by the aforementioned disambiguation techniques (concluding remarks of how this data is used are to follow). Before using Social Web platforms as a source of identity information it was essential that the data, which is implicit to such sites, was analysed using the appropriate measures, thereby assessing the validity of one of the claims of this thesis:

- *Data found on Social Web platforms is representative of real identity information*

The user study presented within chapter 2 compared the social network which users of the Social Web platform Facebook maintained online with their real world equivalent. The results from this study demonstrated a large overlap in these networks ranging from 50% to 100% with an average overlap of 77%. By quantifying the measure of similarity between these networks, the user study results provide an insight into the degree to which the identity constructed by Social Web users is representative of their real identity. With an average overlap to the degree of 77% one can note the large extent to which a correlation exists, however it is worth considering that there is not a predefined threshold, which when exceeded, enables an observer to declare a web user as constructing their real identity in a virtual space. If one uses natural intuition then the figure indicates a strong level of agreement - which could be loosely based on agreement scales from the literature which explore agreement levels [KP03] - thereby suggesting that the above claim is supported.

Observing the degree to which the digital identities constructed by Social Web platforms mirror their real world equivalents indicates that Social Web platforms provide a useful source for identity information. Furthermore, the breakdown of the real network covered by the digital network demonstrates the demographics of this overlap and their coverage of a variety of relationship types: family, friends and colleagues. Identity information is leveraged from such platforms using the techniques outlined in chapter 5, and then used as seed data by the techniques outlined in chapters 7, 8 and 9. The application of this data and the subsequent results in chapter 10 support the following claim:

- *Social data provides the background knowledge required by automated disambiguation techniques*

One of the novel contributions, and indeed the underlying proposition of this thesis, is the use of social data to support disambiguation. The evaluation from chapter 10 demonstrated that leveraging data from Social Web platforms and using this data to support disambiguation techniques yields high accuracy levels. The results from the evaluation quantify the performance of the explored techniques against several baseline measures including the discussed unsupervised technique - included to provide a comparison against a technique which does not rely on seed data - and human processing. The use of social data obtained from Social Web platforms, including Facebook and Twitter, provides disambiguation techniques with information which describes a given person, including biographical and social network information. By utilising the features of this data, the explored techniques are aided with essential information from which to make accurate disambiguation decisions (reflected in the results obtained for the various techniques).

It is worth noting that seed data is only obtained from two Social Web platforms for use by the disambiguation techniques. Despite this limited number the results empirically demonstrate how disambiguation techniques perform well when supported by such data. The used platforms only expose a portion of an individual's digital identity, additional identity fragments could be obtained by leveraging seed data from additional platforms such as the business networking site LinkedIn. The architecture of the approach and the techniques for leveraging social graphs using RDF from chapter 5 function in a modular fashion, the inclusion of additional seed data would be a case of building another exporter for a different Social Web platform - e.g., LinkedIn.

The use of data leveraged from Social Web platforms as seed data can limit the approach if the identity which web users construct on such platforms is not consistent with their identity on the World Wide Web. Within the evaluation there were several cases where this occurred. However, in the overwhelming majority of cases, despite the seed data being minimal - due to the user exposing a limited amount of information within their profile - the semi-supervised disambiguation techniques (Random Walks and Self-training) were still able to cope with the lack of feature coverage provided by the seed data and achieve accurate results.

11.3 Analysis of Methodology Requirements

Part II of this thesis has presented a methodological discussion of techniques to leverage data from the Social Web, generate metadata models describing the underlying knowledge structure of web resources and automated means of disambiguation. Each of these methodology chapters included reviews of current state of the art work and requirements which were gleaned from these analyses. The concluding discussions of chapters 5 and 6 are found within the final section of those chapters and are not included here. This section analyses the previously specified requirements of chapters 4, 7, 8, and 9 with respect to the results of the evaluation from chapter 10.

Chapter 4 presented the requirements and design of an approach to identify *identity web references*. The approach architecture is divided into three stages: seed data gener-

ation, metadata model generation and identity disambiguation. Chapter 4 presented an overview of disambiguation techniques and their applicability to a user-centric approach to disambiguation. Such techniques were found to be divisible into two categories: *seeded* and *unseeded*, the former of which was identified as being the most suitable to the problem setting explored within this body of work. Based on this decision several requirements were listed which both *seeded* techniques and the provided seed data must fulfill. These are now analysed in relation to the presented work:

11.3.1 Requirements for Disambiguation

- *Perform the disambiguation process with minimal supervision*

The approach presented within this thesis alleviates the need for supervision by generating seed data from available sources of digital identity information, searching the Web to collect web resources which *may* cite a person and then using the seed data to disambiguate web references. Each of the explored disambiguation techniques requires no intervention, only the inputting of the compiled seed data and the collection of web resources which are then to be analysed.

- *Achieve disambiguation accuracy comparable to human processing of data*

Evaluating the disambiguation techniques, and the components of the overall approach for identifying web citations, has provided empirical evidence of the comparison of the explored techniques against human processing. A human processing baseline was created by allowing 12 humans to label a portion of the dataset used for evaluation - the entire dataset was not used due to the amount of information which would have been processed and time constraints on the labellers - and using the Interrater agreement between the human raters. The comparison of the disambiguation techniques from chapters 7, 8 and 9 with the human processing baseline indicate a comparable level of performance. The *supervised* technique explored within chapter 7 which uses Inference Rules to infer an *identity web reference* was found to perform poorly in terms of f-measure when compared against the baseline, but much better in terms of precision.

Conversely the Random Walks method from chapter 8 employed a *semi-supervised* strategy by combining both labelled and unlabeled data to support disambiguation. As a result precision levels were lower than the baseline, yet recall levels were higher, resulting in f-measure scores which were comparable yet insufficient to surpass human processing. Self-training a Perceptron classifier when combined with Entailment matching surpassed the baseline in terms recall and f-measure, as did Support Vector Machines aided with Entailment as a feature similarity measure. As mentioned previously, similar f-measure levels to humans, yet demonstrating higher recall levels, demonstrates the applicability of automated techniques to detect a large portion of *identity web references*. A desirable outcome given the context of application and the motivation for protecting a web user from falling victim to malevolent web practices such as identity theft and lateral surveillance.

- *Cope with web resources which do not contain any features present within the seed data*

Chapter 10 presented empirical observations of the improvement in performance of the self-trained classifiers, thereby demonstrating learning capabilities from limited seed data.

The evaluation results also demonstrated how Random Walks over an interlinked graph space allows web resources to be indirectly linked to the social graph, thus enabling the disambiguation of *identity web references* via transitive relations, the effectiveness of this strategy is demonstrated in the resultant high accuracy levels. By analysing the effects of the Self-training process on machine learning classifiers, empirical evidence is provided which demonstrates the ability of the technique to learn from features not present within the seed data. This improvement in feature coverage is shown in the recall levels which increase as the classifiers are iteratively retrained.

11.3.2 Requirements for Seed Data

Applying *seeded* disambiguation techniques places requirements on seed data. Chapter 4 listed the two central requirements which the provided seed data must fulfill, these were as follows:

- *Produce seed data with minimal cost*

One of the notable contributions of this thesis is the use of data obtained from Social Web platforms to aid disambiguation. Due to the widespread use and popularity of Social Web platforms, such platforms provide an ideal source to inexpensively leverage data from. In this thesis only two Social Web platforms are used to collect seed data from: Facebook and Twitter. As discussed previously, the advantage of the approach presented within chapter 5 for collecting seed data is its modular architecture. This allows additional exporters to be included within the approach so that social graphs can be combined from more than two platforms for use as seed data. In each case the production of seed data requires no manual data creation, as the digital identity information of a given person is leveraged from an existing source - this information is what constitutes the person's profile on the platform - thus overcoming the burden of seed data generation.

- *Generate reliable seed data*

Social Web platforms provide a reliable source of identity information which is consistent with the real identity of web users. The user study within chapter 2 investigated the extent to which the identity a web users constructs on the Social Web is similar to their real identity. The results from this study indicate that Social Web platforms provide a useful source for reliable seed data, and coupling these findings with the results from the evaluation of the explored disambiguation techniques demonstrates the reliability of the generated seed data.

It is worth noting that in this instance, the identity fragments reconciled from disparate Social Web platforms for use as seed data may only capture a single consistent identity of a given person. It may be that the identity that a person maintains on Facebook is the same as the identity they maintain on Twitter, for example by using both platforms to maintain business contacts and liaise with colleagues - this was very much observed with the evaluation participants. This seed data therefore provides the necessary means by which the disambiguation of *identity web references* can be performed for the provided identity representation of a person. Should the individual maintain multiple identities (i.e., a representation which mirrors their identity offline, and another which is their alter-ego) then the use of Facebook and Twitter may not be a suitable source for the provision of

seed data to capture alter-ego web references. However the techniques explored within this thesis are not limited to only the listed platforms. As seed data is modelled using Semantic Web technologies, the modular nature of the social graph interlinking technique in chapter 5 enables additional social graphs to be easily included from additional sources. Should a social graph depicting a person’s alterego be provided somehow - either via manual creation or exportation of the alter-ego’s social network from the domain in which they interact - then the techniques will still function regardless.

11.3.3 Disambiguation Techniques

Three different lines of research are explored to tackle the disambiguation problem within this thesis: rules, graph-based methods and machine learning. From each of these areas a disambiguation technique was researched and applied to the task of disambiguating *identity web references* supported by social data. As such each technique carried with it unique requirements gleaned from the literature which the technique must fulfill. These are now analysed for each of the proposed techniques:

11.3.3.1 Rule-based

Chapter 7 presented a disambiguation technique that constructs a rule base from the seed data. The construction technique provides a novel approach to forming rules from RDF instances. The requirements which the technique must fulfil are now discussed.

- *Rules must not overfit or underfit examples*

The rule construction strategy builds specific rules to match all the information within the RDF instance description. As mentioned, rules are also constructed to match data which is unique to the instance. Rules are built which are specific and may overfit instances within the data base, however the specificity of the rule depends on the number of triples included within the antecedent: if the instance description is limited then a more general rule will be built. The built rules are constructed as SPARQL rules, this allows a graph pattern to be placed within the rule’s antecedent - its WHERE clause. Variables are used to denote subjects and objects within the patterns and bind these variables to the social graph owner. As a consequence, rules are formed which can then be applied to the metadata models of the web resources by matching the graph pattern and relating found information to the social graph owner. In short, this strategy regards the seed data as containing features which are positive and should any of those features appear within a given web resource then a match is confirmed. This may seem to be a naive strategy yet the results from §10.7.1 demonstrate the high levels of precision obtained using this technique.

- *Induce rules from limited examples*

The technique presented within chapter 7 is able to build rules from only a limited amount of seed data. Existing rule induction strategies, such as those described within §7.2, are given a set of labelled examples from which a rule base is constructed. The technique in chapter 7 provides a novel induction approach to construct rules from RDF instances within the seed data. In effect one can consider the collection of instances to denote the example set referred to in the literature, thereby attempting to classify an *identity web reference*

by matching information which uniquely defines the RDF instance, and therefore a given resources, from within the provided seed data. The evaluation results demonstrate the high levels of precision of these Inference Rules when applied to disambiguation tasks.

- *Ensure suitable features are used to construct rules*

The rule induction strategy presented within §7.4.1 exploits the semantics of the seed data to induce several rules for each RDF instance within the seed data. For instance, a specific rule is generated covering the features of the instance specified by properties and information within the instance description. Features within the instance description which are unique identifiers of the instance are used to construct separate rules (i.e., A rule is constructed to match the existence of an email address within a given web resource).

11.3.3.2 Graph-based

Random Walks, a graph-based disambiguation technique is presented within chapter 8. By exploiting the graph-like structure of RDF a graph-space is compiled from the seed data and the set of web resources to be disambiguated. Common literals and resources between the metadata models provide traversable paths through the space. From a review of the literature pertaining to graph-based disambiguation techniques several requirements were established which the application of such a technique must adhere to. These are discussed as follows:

- *Provide an interlinked graph-space*

One of the novel contributions to the state of the art in chapter 8 is the method for constructing a graph-space. Although graph-based techniques place emphasis on the need for a graph-space interlinked by common features between relevant elements, there has been little work which applies existing knowledge structures into such a setting. Knowledge representation technologies such as RDF provide a traversable graph-based representation which depicts the relations between statements in a machine readable form. By capitalising on this structure a graph-space is built with ease from the social graph - describing the seed data - and the metadata models of the web resources.

- *Ensure suitable features are used to build a graph space*

The feature set used from the available RDF models (social graph and web resources) is, in this instance, comprised of the literals and resources from those models. This is in contrast to the use of RDF instances as features by the techniques in chapters 7 and 9. The use of Random Walks as a traversal mechanism relies on a multitude of paths and possible routes of traversal being available within the graph space. The use of RDF instances by this technique would also require the selection of an appropriate feature similarity measure, as explored within chapter 9. Instead the knowledge models are kept in tact and the inclusion of additional paths for walks are created.

- *Accurately partition the graph space*

The Random Walks model takes advantage of the properties of the graph space by traversing along possible paths using a stochastic process until a desired node is reached.

By forming edges within the graph-space which refer to ontological concepts, the properties of the graph space can be leveraged to effect how Random Walks are performed. For instance, by weighting edges that refer to ontology relations which denote unique instance properties, the transition probabilities are thus effected. A novel edge-weighting heuristic (η) is presented in chapter 8, and is tuned together with σ , to produce optimum results when clustering with Commute Time and Optimum Transitions. Appendix J presents the results from this tuning and discusses the effects of the variance of edge weighting on the use of RandomWalks, these findings confirm the intuitions from work by [SFYD04, Mal05], which state that edges should be weighted more which denote significance. However further empirical evidence is sought to strengthen the above hypothesis, this is planned for future work described in the following chapter.

11.3.3.3 Semi-supervised Machine Learning

Chapter 9 presented a semi-supervised machine learning technique called Self-training which is applied to train and improve a machine learning classifier. This learnt classifier is then used to disambiguate *identity web references* by labelling a web resource as either citing a given person or not. Using semi-supervised machine learning requires several requirements to be fulfilled - as explained in chapter 9 - these were addressed as follows:

- *Maximise the accuracy of the seed data*

The disambiguation problem addressed within this thesis is set as a binary classification task. Learnt classifiers must be able to distinguish between web resources which cite a specific person and those that do not. Therefore classifiers must be given both positive and negative training data, the initial seed data generated from the approach within chapter 5 provides known positive training data. §9.4.1 describes the method used to generate negative training data by using Rocchio classification to construct a vector space representation of the universal set and positive set. Prototype vectors are then formed to denote centroids within the vector space of the positive and negative set - given that the universal set is treated as the negative set in this instance. Web resources - in their vector form - are compared against each of the prototype vectors, both negative and positive, and the web resources which have a higher cosine similarity with the negative vector than the positive vector are found to be closer to the negative prototype vector in the vector space.

This method of negative set generation - without the provision of pure negative examples - has been explored within the field of text classification, however no work has applied this same approach to negative set generation in the context of web page classification. However given the general notion of feature vectors, it is applicable in this scenario. Applying Rocchio classification requires a careful selection of values for α and β , which impact the prototype vectors. Appendix K describes the findings from tuning $\rho = \frac{\beta}{\alpha}$, and choosing the values that optimise precision at recall 90% for different permutations of self-trained classifiers and feature similarity measures. In cases where the classifier uses Jaccard similarity, precision at recall 90% is optimised where $\rho = \frac{1}{4}$, the standard setting for α and β used within the test classification community. Conversely for both Entailment and IFP, the classifiers function better when using a higher setting of ρ , where generally $\rho > 1$. This demonstrates that a careful selection of ρ is required, often depending on the features selected and the matching heuristic if any that is used.

- *Ensure accurate feature matching*

The variance in feature similarity measures explores the effects of different measures on the resultant classifications and the inherent learning process. Of the three measures explored, Jaccard yielded the highest precision levels - given its requirement for strict graph equivalence when comparing features - and Entailment was found yield high recall levels - given its function of matching features by graph subsumption. Although precise, Jaccard fails in many cases to match features which Entailment and IFP detect. In terms of the above requirement, ensuring that features are matched accurately leads to high precision levels at the cost of recall - indicated by the case of Jaccard similarity. By permitting variations in the matching of RDF instances, recall levels are shown to increase, such that features are detected without the need for exact matching instances descriptions - in the case of Jaccard - or the presence of definitive values - in the case of IFP matching. In machine learning terms, such an affect is attributed to the reduction in overfitting to the provided training data by Entailment.

- *Accuracy must improve as the classifier is trained*

Self-training functions by first learning a classification model from labelled training data: seed data collected from the Social Web as the positive set and web resources generated using Rocchio classification for the negative set. Unlabelled data is incorporated into the learning process by enlarging the training sets with the strongest positive and negative classifications, retraining the classifier and then applying it to the unlabelled data. As mentioned previously the experimental results from chapter 10 present empirical evidence demonstrating the improvement of classifiers as training is performed. No permutation of machine learning classifier and feature similarity measure degrades in terms of f-measure through Self-training. In most cases permutations are found to reduce in precision as more instances are added to the training data following a cycle of the Self-training process, but while precision decreases, recall is found to increase significantly. This demonstrates the behaviour of Self-training to learn from classifications and adapt the hypothesis to unseen features which were not present within the seed data.

- *Enlarge training data with suitable instances*

The number of instances which the training sets are enlarged with is controlled by the parameter k . The reliance on Rocchio classification to generation negative examples increases the likelihood of reducing false negatives but does not guarantee it, therefore it is necessary to employ Self-training in a setting which enables the classification mistakes to be overcome. To demonstrate this decision the enlargement of the negative training set was performed using: a) the strongest negative; and b) all negative examples. The findings in chapter 10 presents empirical evidence demonstrating the effectiveness of strong negative enlargement on recall levels. Where the negative set is enlarged with all negatives, there is a tendency for the classifier to produce many false negatives - reflected in the precision/recall curves presented in chapter 10 by recall at precision 90%. Enlargement using the strongest negative each time leads to an increase in recall, as only the instance which contains the fewest known positive features will be selected.

11.4 Summary

The research explored within this thesis has demonstrated that automated means of disambiguation can yield accuracy levels which are comparable to human processing, and therefore be deployed to replace humans in their current task of identifying web citations. Given the size of the Web and its propensity to grow at a staggering rate, the need for automating the process of detecting personal information online is evident. Social Web platforms now provide a rich set of features and functionalities with which users can construct a digital identity deployed as a user profile. Empirical evidence from the evaluation of the explored disambiguation techniques indicates that using data leveraged from the Social Web as seed data yields accurate results and overcomes the burden imposed on automated techniques of building this seed data from scratch.

The process of disambiguation is important for the future of the Web. As information is published then the need to monitor and interpret this information grows. The increase in use of technologies such as RDFa and Microformats, and their adoption by large and established organisations such as Google, has lead to an increase in the embedding of metadata descriptions in web pages. Such embedding provides gleanable knowledge representations describing information in a machine-readable form. Being able to analyse this information is essential in discovering information through automated means. The techniques presented within this thesis go some way to providing a basis through which relevant information can be discovered describing a given entity - which in the context of this thesis is a person. The following chapter discusses the future work which is planned based on the explored techniques, and the outlook of the future of identity and the Internet.

Chapter 12

Future Work and Outlook

12.1 Introduction

The work presented within this thesis has explored a variety of automated methods designed to alleviate the manual process by which web users detect their web citations. The presented techniques, namely the disambiguation techniques in chapters 7, 8 and 9, are not constrained to only the domain of personal information discovery and the disambiguation of *identity web references*. The term *identity* has been used throughout this thesis to refer to a set of attributes, or properties, which are able to uniquely identify a given person. The collection of such attributes into a single setting provides sufficient evidence to reduce the ambiguity of a given person's citation into a single entity. One can imagine that any given entity, thing or object has an identity, it is what makes that entity unique, be it the unique stripe pattern and scars of a Tiger which separates the animal from others, or the model, make and serial number of a food processor which makes that object unique. This chapter discusses plans for future work which have arisen as a result of the research presented within this thesis. This chapter also presents the outlook for the future development of the World Wide Web and the role which identity will play, particularly on the Internet of Things and the Web of Linked Data.

This chapter is structured as follows: section 12.2 presents a proposed approach to perform risk assessment for web users as an extension of the approach presented within chapter 4 of this thesis. Section 12.3 describes future work based on the methodologies explored within this thesis, mainly concerning extensions to the Random Walks and Self-training techniques, and how the disambiguation techniques could be combined together. Section 12.4 discusses the future of the World Wide Web and the current progression towards an Internet of Things, and how identity of objects and real world identity may lead to ambiguity issues, and the need to resolve such issues. Section 12.5 discusses additional work currently being performed by the author of this thesis based on the techniques described in this thesis. Section 12.6 discusses the future of web content with the next revision of HTML containing metadata functionalities and the how the techniques within this thesis are applicable to such a setting. Section 12.7 describes the outlook of personal information dissemination influenced by the findings from this work and possible means of alleviating such problems. Section 12.8 summarises the key points raised within this chapter.

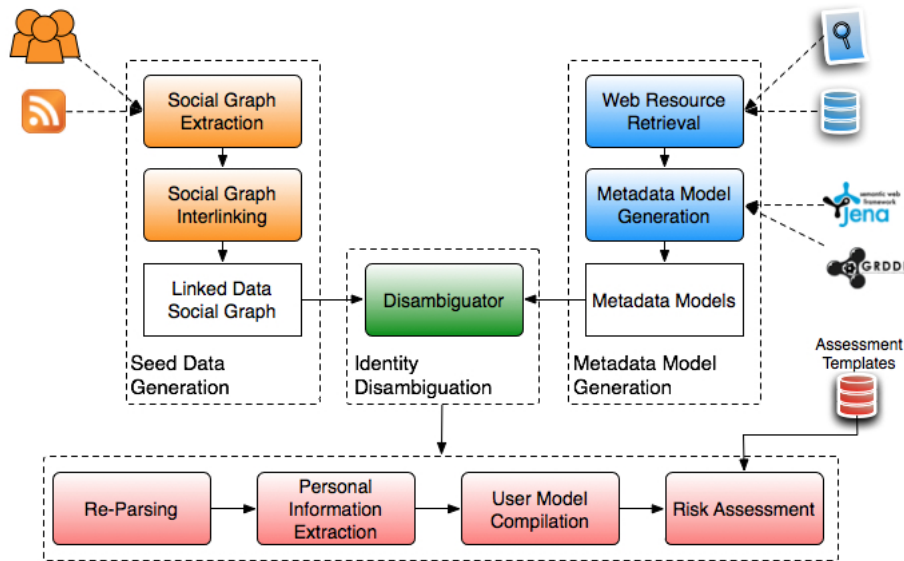


Figure 12.1: Architecture of an approach to assess the risk of identity theft/lateral surveillance posed to web users

12.2 A Proposed Approach to Risk Assessment

The work presented within this thesis is designed to alleviate the manual process by which web users monitor the Web for web resources containing their personal information. Future work based upon the techniques and methods discussed within this body of work will extend the architecture presented within chapter 4 to not only discover *identity web references* but also perform *risk assessment* automatically, thereby judging whether a given person is susceptible to lateral surveillance or identity theft.

Figure 12.1 presents the architecture of the proposed approach and how it will fit into the current approach utilised within this thesis. The general mechanism for disambiguation is the same: collecting seed data from Social Web platforms, generating metadata model representations of possible web citations, and then passing the collection of web resources and the seed data onto a given identity disambiguation technique. The evaluation in chapter 10 empirically demonstrates the variance in performance of the three explored techniques: Inference Rules for high levels of precision, and Random Walks and Self-training for high recall levels - the latter showing significantly higher F_3 scores. Based on these differences, the optimum application would be to combine the techniques together, for instance by first enlarging the seed data using Inference Rules - given that very few false positives are generated this way - and then performing Self-training to learn from the unlabelled web resources and thus detect more web references - increasing recall levels.

A key problem with the current techniques however is their feature representation for the web resources. At present only a limited number of features are used for HTML documents: name, email, web address and location, from which a metadata model is then built to represent the web resource in a machine-processable format. The goal therefore is to improve this current method, following disambiguation, to extract additional features from a given web resource. This is presented in the lower portion of Figure 12.1 where the disambiguation

results are passed onto a separate pipeline for risk assessment.

The pipeline begins by re-parsing the collection of identified web references - using their original web page form, not the metadata model representation - and extracting any pieces of personal information that are found within the web page. The latter of these stages is analogous to the attribute extraction challenge which forms part of the Web People Search workshop.¹ Existing approaches presented at the workshop cover a range of methods from pattern recognition to regular expression matching. The use of hidden Markov models from chapter 6 could be adapted for this task, by providing additional training data for distinct person attributes.

Once the personal information has been extracted then a model is built from the extracted information in order to compile as similar an identity representation as possible to the given person. This model is then assessed for risk using a set of templates, where the templates stipulate a different requirement upon different attributes being more important than others. This is to reflect the different scenarios and contexts which the user may exist in, for example in the UK, the appearance of a person's social security number, given that they are not used in the UK, should carry less weight than the national insurance number of the person being visible. The assessment stage of the process will consider the collation of the various person attributes within the given user model and the risk which this culmination poses to the individual. *Person Attribute* is used as a general term to refer to any given piece of information that could be attributed to a person, the following subsection now presents results from a manual assessment of these attributes using web references returned using the combination of SVM with IFP - given its optimal performance for precision at recall 90% - thereby presenting an overview of the severity of such information being visible on the Web.

Table 12.1: Person attributes visible in web resources returned using Self-trained SVM with IFP

Attribute	Percentage Visible
Date of birth	16%
Birth place	16%
Occupation	97%
Address	22%
Affiliation	100%
School	38%
Nationality	49%
Relatives	6%
Phone	45%
FAX	69%
Email	100%
Web site	100%

12.2.1 The Severity of Personal Information Dissemination

Using the approach proposed in Figure 12.1, and choosing the optimum automated technique presented within this thesis: Self-trained SVM with IFP, for each of the evaluation participants the returned web resources were manually assessed for the presence of person attributes. The results from this manual assessment are shown in Table 12.1, where the per-

¹<http://nlp.uned.es/weps/weps-3/guidelines/42-guidelines-for-the-weps-3-attribute-extraction-subtask>

son attributes are based around those used for the Attribute Extraction Challenge together with the *address* attribute. The goal of this task was to empirically observe the availability of certain person attributes which could then be extracted from the disambiguated pages using the above approach.

The results indicate that certain attributes are more widely available than others, for instance *web site*, *email* and *affiliation* are found in all of the participants' disambiguated web resources. Attributes which pose a greater risk if found: *date of birth*, *birth place* and *relatives* appear less frequently, potentially reflecting the importance that the participants attach to such information and the need for it to be hidden from view. Proposing an approach to identify web citations and then extract the relevant attributes would then enable an assessment to be made of the risk which the visibility of these attributes pose to a given individual. The UK government's initiative: *Get Safe Online*² provides a succinct quote describing the risk posed from identity theft and the uncontrolled dissemination of personal information:

"Identity thieves can piece together your identity from public information piece by piece like putting together a jigsaw."

In effect assessing the risk of a web user falling victim to identity theft, or being susceptible to lateral surveillance, is replicating the actions of an identity thief: collating person attributes together into a single context. Within the wider context of Personal Information Management, deploying the technologies explored within this thesis requires careful consideration of the anonymisation of data and security. Collecting personal information into a single location requires that the information is not lost or stolen, and cannot be accessed by any person other than the individual it describes.

12.3 Future Methodologies

The research presented within this thesis covers a broad spectrum of work, however in certain cases the presented methods could be explored further, either in new and novel settings, or using benchmarking against similar methods via community utilised datasets. This section details planned future work based on the techniques described in Part II of this thesis.

12.3.1 Random Walks

12.3.1.1 Spectral Clustering with Commute Time Kernel

The use of Random Walks provides a means of traversing a graph-space - constructed from metadata models - measuring the distances between nodes and applying agglomerative clustering. Work by [YFD⁺07] refers to the use of Commute Time distances as applying a *Commute Time Kernel* to a graph-space in order to transform it into a separate feature space. The kernel works by removing the nodes which link the web resources with the social graph and instead replace them with direct links - where these links are labelled with distances between nodes. In mathematical terms this step reduces the dimensionality of the graph-space and the resultant transition probability matrix. In this thesis agglomerative

²http://www.getsafeonline.org/nqcontent.cfm?a_id=1491

clustering has been used as the strategy by which similar nodes are grouped together by clustering nodes which take less time to reach or are closer together. One of the issues with applying agglomerative clustering is the reliance on tuning a threshold (σ) by which clusters are merged. In cases where a person has a high web presence level, the strongly connected component which is found to contain the *social graph root node* tends to be entirely composed of web resources which cite the person. As a consequence web resources which take longer to reach - i.e., outliers from the social graph - are classed as not citing the individual, resulting in poorer recall levels.

Spectral decomposition of a graph-space could provide a solution to this problem by using the graph's spectrum to cluster nodes within the graph space. In work by [HZG05, OL07] graph-spaces are compiled from similar features between web resources (which denote nodes) and creating edges between web resources based on a given similarity heuristic. This setting is similar to the graph-space which is produced following the application of a *Commute Time Kernel*, by connecting web resources using their traversal time and weighting those edges by the time taken to reach the node and return. Therefore this suggests that Spectral Clustering [BJ05] could then be applied to the graph space in order to segment the data points.

12.3.1.2 Resolution of Random Walks

The Commute Time measure used within chapter 8 functions by using the Random Walk model and the derived transition probability matrix to calculate the Commute Time from leaving a given node i reaching another node j and returning again to node i . Using Commute Time requires the resolution of the walk to be set to a suitable value, which in this case was 2, in order to preserve local information in the graph-space and permit the exploration of the effects of other parameters on the Random Walks process, and the accuracy of the resulting clustering technique. Existing work by [SJ02] has experimented with differing values of t - denoting the number of steps used by the Markov process - and found smaller values to be more suited to graphs where class membership changes over a path which is short in length - i.e., leaving the *social graph root node* and reaching a web resource that does not cite the person over a low number of steps.

Further work within this area will explore the effects of the resolution of the walk over the graph-space based on the parameter setting for the proportion of edges weights, defined by η , and the clustering threshold, defined by σ , before moving on to expand this exploration for other parameter values. Experiments will be performed on the same datasets as [YFD⁺09] in order to assess how Commute Time is effected by the resolution of the walk over the IMDB dataset.

12.3.1.3 Edge Weights and their Effects on Random Walks

Building a graphs-space from metadata models allowed edges to be weighted based on the ontology concepts which they denoted and their inherent characteristics. The proportion of edge weights - controlled by the value of η - provides a novel investigation into the effects of weighting edges on the Random Walks process. Additional work is to be performed within this area to investigate the applicability of this theory in different domains and therefore provide additional evidence to support the assumptions made in work by [SFYD04] that

edges which denote uniqueness should carry *more* weight. At present this statement does not seem to be sufficiently backed up with empirical evidence.

12.3.2 Semi-supervised Machine Learning

The evaluation presented in chapter 10 compares the three explored disambiguation techniques against one another. The most promising technique was found to be the use of semi-supervised machine learning from chapter 9. By exploring the use of such a technique, several avenues of work have been raised which are to be investigated in detail, these are now discussed.

12.3.2.1 Discovering an Optimal Hypothesis during Self-training

The results from applying Self-training to learn and iteratively improve a machine learning classifier, presented within §10.7.3.2, demonstrates that the classifier improves from its initial hypothesis for all combinations with feature similarity measures. However, in several cases the effects of training indicate that an earlier hypothesis, not the primary established hypothesis, achieves higher levels of f-measure than the final hypothesis. This therefore suggests that stopping the training cycle at this stage within the learning process would yield a more optimal outcome with respect to disambiguation accuracy levels.

One of the primary reasons for utilising Self-training combining with an effective negative set generation strategy is its ability to avoid a local maxima. This is in comparison to similar semi-supervised learning strategies such as Expectation-Maximisation (see §9.2.1.2 for a discussion of this technique). Therefore a natural progression of the exploration of Self-training would be to investigate techniques and conditions in order to detect a global maxima and thus enable the learnt hypothesis to halt at this peak within the learning process. In effect this investigation will explore possible stopping conditions and heuristics which would lead to the selection of an optimal hypothesis during the Self-training process.

12.3.2.2 Combining the Disambiguation Techniques

A product of the evaluation results detailed within chapter 10 was the high level of precision achieved using the rule-based technique discussed within chapter 7. Due to the strict nature of Inference Rules, requiring knowledge structures to match the exact graph pattern within the antecedent of a rule, once a rule matches the metadata model of a given web resource the likelihood of an *identity web reference* is almost definite - indicated by the high level of precision obtained when evaluating Inference Rules.

Self-training: Positive Set Enlargement One of the limitations of the Self-training method explored within chapter 9 is the lack of initial seed data of both positive and negative classes. The highly precise nature of the rule-based technique suggests that this approach could be leveraged to enlarge the initial positive training set, thus providing additional instances, and hence features, from which a classifier can be trained.

A by-product of this enlargement allows alternative techniques to be utilised when constructing the negative set, namely the technique presented within work by [FL06] which forms a statistical distribution from positive features. The problem setting discussed throughout this thesis only has a limited amount of positive seed data to work with. Moreover the

existing negative set generation technique, as discussed within §9.4.1, which employs Rocchio classification to approximate outliers with respect to the positive set via prototype vectors could also be enhanced: the initial positive prototype vector which is constructed only contains a portion of the total positive features within the feature space and is therefore a weak approximation of the entire positive set - this causes web resources which cite the person to lie away from the centroid despite being true references. By enriching this vector representation with additional instances the construction of the negative prototype vector, given that this vector is built from the universal set, would yield more reliable negative instances.

Outlier Detection: Positive Set Enlargement Another planned avenue of work is to utilise a single-class classification approach. Baseline 4 used within the evaluation presented in chapter 10 uses a naive approach to detecting outliers within a feature space. As a consequence it tends to produce many false positives - characterised by the low level of precision yielded using this technique. One of the central factors influencing this low level of precision is the original feature distribution which is used for the positive set. As in the majority of cases labelled data is limited, the use of a proximity measure, or even single-class SVM which was found to have insufficient training data to separate its hyperplane from the origin, has very few features to learn a distribution from. This sparseness therefore renders single-class classification techniques limited, unless the distribution of the positive set was enlarged somehow. This is where the use of Inference Rules would be beneficial: given its high-precision nature.

Co-training The application of Co-training as a semi-supervised learning strategy is a natural progression of the work described in chapter 9. The ethos upon which Co-training is based is the use of two classifiers to simultaneously apply labels to unlabelled data, followed by adjustments of the training sets for each classifier using the classifications made by the alternative classifier. The basis for this strategy is that each classifier has a unique perspective of the datasets; either by performing classification using a distinct approach, or by considering different feature sets. Future work will therefore combine a high-recall disambiguation technique with a high-precision disambiguation technique by co-training using Inference Rules for high-precision, and Perceptron/SVM with Entailment for high-recall.

12.4 Identity and the Internet of Things

Identity provides a notion of what a thing or entity is through the provision of a set of attributes or properties. This combination reduces the ambiguity of an entity with respect to other similar entities of the same creed. The Internet, at the time of writing this thesis, has begun to expand into the real world to provide an Internet of Things: the original Semantic Web vision didn't limit resources to the World Wide Web, but defined resources in the real world identified by a URI. The Internet of Things encapsulates this vision by allowing objects and "things" in real world environments to be identified by Radio Frequency Identifiers (RFID) [WBC⁺09] and IP addresses. This identification gives way to ubiquitous computing where things become referenceable and ultimately pervasive. For instance, work

by [GKC04] described a house where every intrinsic object and thing was identified by an IP address. The alarm clock upon waking the habitant of the house would be able to turn the lights on and boil the kettle in the kitchen. The Semantic Web provides the grounding for this reasoning and interaction to happen, by offering common mediation of information through ontologies, reasoning mechanisms through rules, and handling metadata through formats capable of expressing meaning.

At the time of writing the Internet of Things is very much in its infancy. That said there has been substantial work recently within the field of semantic sensor networks³ focussing on how Semantic Web technologies can be used to interface and manage data produced by real world sensors. One can hypothesise of the problems that will be faced if one considers the current Web of Linked Data and the associated issues concerning the identity of resources. One of the research challenges associated with this field of work has been the use of multiple URIs to identify the same resource, often referred to as the problem of coreference resolution [GJM09] and URI disambiguation [JGM08]. The need to find all the URIs which refer to a given resource fall neatly into the problem setting explored within this thesis. The identity of the resource therefore must be utilised in some way to enable all additional identity references to be found, thus capturing the complete set of URIs which denote the resource. Within the domain of electronic products, a given resource may resemble a unique product listing where the product is listed in multiple locations on multiple suppliers' websites. The background knowledge for finding all these references of the product would incorporate a set of properties which make the product unique (model, manufacturer, cost, year of manufacture). One can now see how the techniques presented within chapters 7, 8 and 9 are sufficiently generalised that applying them to a disparate domain would be feasible.

This notion of resource ambiguity is therefore one which must also be addressed within the Internet of Things. How a resource is labelled, or identified, will vary between domains of discourse. This is already true of languages where famous landmarks are referred to using different names depending on the language of use. The techniques explored within this thesis provide a basis from which disambiguation can be achieved to a relatively high level of accuracy.

12.5 Data.dcs

At the time of documenting the research presented within this thesis, the author has begun work on a project to produce Linked Data describing people, publications and research groups within the University of Sheffield's Department of Computer Science. Such legacy data is stored within HTML documents and RSS feeds, which in both cases, lacks metadata. The *Data.dcs* project aims to develop techniques to *triplify* this data, thus creating a rich metaspace providing consistent interpretations of information from distributed documents. Coreferring entities (i.e., the same people who appear on different pages) are discovered which, in turn, allows people to be linked to their publications and the research groups which they are members of. This dataset is then linked to the Web of Linked Data, such that members of the department are linked to the publications they have authored, the details of which are hosted in external databases.

³<http://www.ict.csiro.au/conferences/ssn/ssn09>

The techniques explored within this thesis form the basis for the project. For instance the information extraction technique used in chapter 6 for building metadata models describing HTML documents is applied to leverage information about people in the department from pages within the web site. SPARQL rules are applied to enable person disambiguation and link members of the department with the papers they have authored. Such work demonstrates the applicability of the techniques presented in this thesis to real world problems. An example Linked Dataset produced as part of the project can be found on the Data.dcs web site⁴, where it can be browsed and queried, taking the explorer into the wider Web of Linked Data.

12.6 HTML5: The next revision of the World Wide Web

Throughout this thesis metadata plays a crucial role in describing information in such a way that machines and automated processes can look up the metadata, and its meaning, within a web accessible ontology. It allows a machine to know what a piece of information is, how it is associated with other pieces information and how it can be used. At the time of writing this thesis the major Web technologies adopted to describe information in a machine-readable form were RDFa and Microformats. As highlighted in chapter 6, the uptake of such technologies has risen in recent years as web developers see the benefits of providing machine-readable labels to their content, content which was previously intended only for human consumption.

Development of HTML5 is currently underway and will provide similar machine-readable tags embedded within web content. The techniques and methods explored within this thesis function using Semantic Web technologies including RDF and ontologies. Already web developers and Semantic Web evangelists have constructed RDF exportation scripts for web documents containing markup conforming to HTML5. Such markup allows resources and their properties and attributes to be included within human-readable web documents alongside original content. Issues will emerge which must be solved, such as resource disambiguation. The techniques explored within this thesis provide the necessary means to address such issues. For instance, the use of Inference Rules constructed from background knowledge could easily be adapted to match information within RDF models gleaned from web documents containing HTML5 markup.

12.7 Summary

This chapter has presented the plans for future work and the outlook of the future of identity. The investigated techniques described within chapters 7, 8 and 9 are not limited to a single domain of application, the disambiguation of *identity web references* of people. Generalising the definition of an *identity web reference* to incorporate the disambiguation of entities opens up a variety of possible avenues of work. This chapter has only suggested several examples which came to mind throughout the research conducted within this thesis.

⁴<http://data.dcs.shef.ac.uk/>

Bibliography

- [ABK⁺08] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: A nucleus for a web of open data. In *6th International Semantic Web Conference, Busan, Korea*, pages 722–735, 2008.
- [Abn04] Steven Abney. Understanding the yarowsky algorithm. *Computational Linguistics*, 30(3):365–395, 2004.
- [Ack89] Russell Ackoff. From data to wisdom. *Journal of Applied Systems Analysis*, 16:3–9, 1989.
- [AGM03] Arvind Arasu and Hector Garcia-Molina. Extracting structured data from web pages. In *SIGMOD '03: Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 337–348, New York, NY, USA, 2003. ACM.
- [AGV05] Javier Artiles, Julio Gonzalo, and Felisa Verdejo. A testbed for people searching strategies in the www. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 569–570, New York, NY, USA, 2005. ACM.
- [Ai05] Somjit Arch-int. Web document clustering using semantic link analysis. *Computational Intelligence for Modelling, Control and Automation, International Conference on*, 2:13–18, 2005.
- [AKKP08] Waseem Akhtar, Jacek Kopecky, Thomas Krennwallner, and Axel Polleres. Xsparql: Traveling between the xml and rdf worlds – and avoiding the xslt pilgrimage. In Manfred Hauswirth, Manolis Koubarakis, and Sean Bechhofer, editors, *Proceedings of the 5th European Semantic Web Conference*, LNCS, Berlin, Heidelberg, June 2008. Springer Verlag.
- [And05] Mark Andrejevic. The work of watching one another: Lateral surveillance, risk and governance. *Surveillance and Society*, 2 (4):479–497, 2005.
- [And06] Mark Andrejevic. The discipline of watching: Detection, risk, and lateral surveillance. *Critical Studies in Media Communication*, 23(5):392–407, 2006.
- [AvH04] Grigoris Antoniou and Frank van Harmelen. *A Semantic Web Primer*. The MIT Press, April 2004.

- [AW06] Ralitsa Angelova and Gerhard Weikum. Graph-based text classification: learn from your neighbors. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 485–492, New York, NY, USA, 2006. ACM.
- [Bar07] Naomi Baron. My best day: Presentation of self and social manipulation in facebook and im. In *Proceedings of Eighth International Conference, Association of Internet Researchers, Vancouver,, 2007*.
- [BB08] Christian Becker and Chris Bizer. Dbpedia mobile: A location-enabled linked data browser. In *Linked Data on the Web (LDOW2008)*, 2008.
- [BBL08] Dave Beckett and Tim Berners-Lee. Turtle - terse RDF triple language, W3C team submission, 2008.
- [BDKW06] Horst Bunke, Peter Dickinson, Miro Kraetzl, and Walter D Wallis. *A Graph-Theoretic Approach to Enterprise Network Dynamics*. Birkhauser, 2006.
- [BG04a] Indrajit Bhattacharya and Lise Getoor. Deduplication and group detection using links. In *ACM SIGKDD Workshop on Link Analysis and Group Detection (LinkKDD)*, 2004.
- [BG04b] Dan Brickley and Ramanathan Guha. RDF Vocabulary Description Language 1.0: RDF Schema, W3C Working Draft. Technical report, World Wide Web Consortium, <http://www.w3.org/TR/rdf-schema/>, 2004.
- [BHAR07] Chris Bizer, Tom Heath, Danny Ayers, and Yves Raimond. Interlinking open data on the web. In *European Semantic Web Conference, Poster Track*, Innsbruck, Austria, 2007.
- [BHBD05] John G. Breslin, Andreas Harth, Uldis Bojars, and Stefan Decker. Towards semantically-interlinked online communities. In *2nd European Semantic Web Conference*, pages 500–514, 2005.
- [BHLT06] Tim Bray, Dave Hollander, Andrew Layman, and Richard Tobin. Namespaces in XML 1.0 (second edition). World Wide Web Consortium, Recommendation REC-xml-names-20060816, August 2006.
- [Bis95] Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, November 1995.
- [BJ05] Francis R. Bach and Michael I. Jordan. Learning spectral clustering. In *Advances in Neural Information Processing Systems 16*. MIT Press, 2005.
- [BL06] Tim Berners-Lee. Notation 3: A readable language for data on the web, March 2006.
- [BLHL01] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web: Scientific american. *Scientific American*, May 2001.
- [BM98] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Eleventh Annual Conference on Computational learning theory*, pages 92–100. Morgan Kaufmann Publishers, 1998.

- [BM05] Ron Bekkerman and Andrew McCallum. Disambiguating web appearances of people in a social network. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 463–470, New York, NY, USA, 2005. ACM.
- [BM07] Dan Brickley and Libby Miller. FOAF vocabulary specification. Technical report, FOAF project, May 2007. Published online on May 24th, 2007 at.
- [BPCB08] Uldis Bojars, Alexandre Passant, Richard Cyganiak, and John Breslin. Weaving SIOC into the Web of Linked Data. In *Proceedings of the WWW 2008 Workshop Linked Data on the Web (LDOW2008), Beijing, China*, Apr 2008.
- [BPM⁺08] Diego Berrueta, Jon Phipps, Alistair Miles, Thomas Baker, and Ralph Swick. Best practice recipes for publishing rdf vocabularies, August 2008.
- [BPSW70] Leonard E. Baum, Ted Petrie, George Soules, and Norman Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *The Annals of Mathematical Statistics*, 41(1):164–171, 1970.
- [Bri03] Dan Brickley. Basic geo (wgs84 lat/long) vocabulary, January 2003.
- [BSA94] Chris Buckley, Gerard Salton, and James Allan. The effect of adding relevance information in a relevance feedback environment. In *SIGIR '94: Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 292–300, New York, NY, USA, 1994. Springer-Verlag New York, Inc.
- [Bur98] Christopher J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2:121–167, 1998.
- [Car07] Jorge Cardoso. The semantic web vision: Where are we? *Intelligent Systems*, 22(5):84–88, 2007.
- [CBK09] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. Technical report, University of Minnesota, 2009.
- [CCDW04] Fabio Ciravegna, Sam Chapman, Alexiei Dingli, and Yorick Wilks. Learning to harvest information for the semantic web. In *Proceedings of the 1st European Semantic Web Symposium (ESWS-2004)*, May 2004.
- [CDF⁺98] Mark Craven, Dan DiPasquo, Dayne Freitag, Andrew McCallum, Tom Mitchell, Kamal Nigam, and Seán Slattery. Learning to extract symbolic knowledge from the world wide web. In *AAAI '98/IAAI '98: Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence*, pages 509–516, Menlo Park, CA, USA, 1998. American Association for Artificial Intelligence.
- [CEH⁺08] Caterina Caracciolo, Jérôme Euzenat, Laura Hollink, Ryutaro Ichise, Antoine Isaac, Véronique Malaisé, Christian Meilicke, Juan Pane, Pavel Shvaiko, Heiner Stuckenschmidt, Ondrej Sváb-Zamazal, and Vojtech Svátek. Results of the ontology alignment evaluation initiative 2008. In *OM*, 2008.

- [CLH09] Ying Chen, Sophia Yat Mei Lee, and Chu-Ren Huang. Polyuhk: A robust information extraction system for web personalnames. In *2nd Web People Search Evaluation Workshop (WePS 2009), 18th WWW Conference*, 2009.
- [CMHJ⁺09] Philippe Cudré-Mauroux, Parisa Haghani, Michael Jost, Karl Aberer, and Hermann De Meer. idmesh: Graph-based disambiguation of linked data. In *18th International World Wide Web Conference*, pages 591–591, April 2009.
- [CN89] Peter Clark and Tim Niblett. The cn2 induction algorithm. In *Machine Learning*, pages 261–283, 1989.
- [CNC05] Sam Chapman, Barrington Norton, and Fabio Ciravegna. Armadillo: Integrating knowledge for the semantic web. In *Proceedings of the Dagstuhl Seminar in Machine Learning for the Semantic Web*, February 2005.
- [Coh02] William W. Cohen. Improving a page classifier with anchor extraction and link analysis. In *Advances in Neural Information Processing Systems 15*, pages 1481–1488. MIT Press, 2002.
- [Cro07] Charlene Croft. Facebook vs. myspace – reclaiming and reinventing identity online, April 2007.
- [CS07] Nick Craswell and Martin Szummer. Random walks on the click graph. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 239–246, New York, NY, USA, 2007. ACM.
- [CSZ06] Olivier Chapelle, Bernard Schölkopf, and Alexander Zien. *Semi-Supervised Learning*. MIT Press, Cambridge, MA, 2006.
- [Cyg08] Richard Cyganiak. Expressing the XFN microformat in RDF. Technical report, Deri, 2008.
- [Dar90] Richard Darlington. *Regression and Linear Models*. McGraw-Hill, 1990.
- [DB04] Judith Donath and Danah Boyd. Public displays of connection. *BT Technology Journal*, 22(4):71–82, October 2004.
- [DHM05] Xin Dong, Alon Halevy, and Jayant Madhavan. Reference reconciliation in complex information spaces. In *SIGMOD '05: Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 85–96, New York, NY, USA, 2005. ACM.
- [DLGT03] Francois Denis, Anne Laurent, Rémi Gilleron, and Marc Tommasi. Text classification and co-training from positive and unlabeled examples. In *Proceedings of the ICML 2003 Workshop: The Continuum from Labeled to Unlabeled Data*, pages 80–87, 2003.
- [DLLH03] Anhai Doan, Ying Lu, Yoonkyong Lee, and Jiawei Han. Object matching for information integration: A profiler-based approach. In *Proceedings of the IJCAI-03 Workshop on Information Integration on the Web. (2003)*, pages 53–58, 2003.

- [DLR77] Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.
- [Dom95] Pedro Domingos. Rule induction and instance-based learning: A unified approach. In *4th international joint conference on Artificial intelligence*, pages 1226–1232. Morgan Kaufmann, 1995.
- [Dow05] Stephen Downes. Semantic networks and social networks. *The Learning Organization: An International Journal*, 12(5):411–417, May 2005.
- [DPHS98] Susan Dumais, John Platt, David Heckerman, and Mehran Sahami. Inductive learning algorithms and representations for text categorization. In *CIKM '98: Proceedings of the seventh international conference on Information and knowledge management*, pages 148–155, New York, NY, USA, 1998. ACM.
- [dPSF09] Cesar de Pablo Sanchez and Paloma Martinez Fernandez. UC3M at WePS2-AE: Acquiring patterns for people attribute extraction from webpages. In *2nd Web People Search Evaluation Workshop (WePS 2009), 18th World Wide Web Conference*, 2009.
- [ECD⁺05] Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. Unsupervised named-entity extraction from the web: an experimental study. *Journal of Artificial Intelligence*, 165(1):91–134, 2005.
- [EMS08] Jérôme Euzenat, Adrian Mocan, and François Scharffe. Ontology alignments. In *Semantic Web and Beyond*, pages 177–206. Springer, 2008.
- [ES04] Marc Ehrig and Steffan Staab. QOM - Quick Ontology Mapping. In *Third International Semantic Web Conference (ISWC 2004)*, November 7-11 2004.
- [ES07] Jérôme Euzenat and Pavel Shvaiko. *Ontology matching*. Springer-Verlag, Heidelberg (DE), 2007.
- [ESL07] Nicole B. Ellison, Charles Steinfield, and Cliff Lampe. The benefits of facebook friends: Social capital and college students’ use of online social network sites. *Journal of Computer-Mediated Communication*, 12:1143–1168, 2007.
- [EV04] Jérôme Euzenat and Petko Valtchev. Similarity-based ontology alignment in OWL-Lite. In *European Conference on Artificial Intelligence ECAI-04 (2004) European Conference on Artificial Intelligence ECAI-04 (2004) European Conference on Artificial Intelligence ECAI-04 (2004) European Conference on Artificial Intelligence ECAI-04 (2004) European Conference on Artificial Intelligence*, Valencia, España, 2004.
- [FDZJ05] Tim Finin, Li Ding, Lina Zhou, and Anupam Joshi. Social networking on the semantic web. *The Learning Organization: An International Journal*, 12(5):418–435, May 2005.

- [FH04] Michael Ben Fleischman and Eduard Hovy. Multi-document person name resolution. In *Annual Meeting of the Association for Computational Linguistics (ACL), Reference Resolution Workshop*, pages 66–82, 2004.
- [FHLW03] Dieter Fensel, James A. Hendler, Henry Lieberman, and Wolfgang Wahlster, editors. *Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential [outcome of a Dagstuhl seminar]*. MIT Press, 2003.
- [FL06] Gabriel Pui Cheong Fung and Hongjun Lu. Text classification without negative examples revisit. *IEEE Trans. on Knowl. and Data Eng.*, 18(1):6–20, 2006. Member-Yu, Jeffrey X. and Fellow-Yu, Philip S.
- [Fle81] Joseph Fleiss. *Statistical Methods for Rates and Proportions*. Wiley-Interscience, 1981.
- [FM99] Dayne Freitag and Andrew Kachites McCallum. Information extraction with hmms and shrinkage. In *Proceedings of the AAAI-99 Workshop on Machine Learning for Information Extraction*, pages 31–36, 1999.
- [For73] David G. Forney. The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, 1973.
- [GAH05] Ralph Gross, Alessandro Acquisti, and H. John Heinz, III. Information revelation and privacy in online social networks. In *WPES '05: Proceedings of the 2005 ACM workshop on Privacy in the electronic society*, pages 71–80, New York, NY, USA, 2005. ACM.
- [Gal90] Simon I. Gallant. Perceptron-based learning algorithms. *Neural Networks, IEEE Transactions on*, 1(2):179–191, 1990.
- [GEP08] Gunnar Grimnes, Peter Edwards, and Alun Preece. Instance based clustering of semantic web resources. In *Proceedings of the 5th European Semantic Web Conference*, LNCS. Springer-Verlag, 2008.
- [GJM09] Hugh Glaser, Afraz Jaffri, and Ian Millard. Managing co-reference on the semantic web. In *WWW2009 Workshop: Linked Data on the Web (LDOW2009)*, April 2009.
- [GKC04] Neil Gershenfeld, Raffi Krikorian, and Danny Cohen. The internet of things. *Scientific American*, 291(4):46–51, 2004.
- [GP03] Charles F. Goldfarb and Paul Prescod. *XML Handbook, Fifth Edition*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2003.
- [Gro08] W3C Working Group. RDFa primer: Bridging the human and data webs, October 2008.
- [HA04] Victoria J. Hodge and Jim Austin. A survey of outlier detection methodologies. *Artificial Intelligence Review*, 22:2004, 2004.
- [Hau09] Michael Hausenblas. Exploiting linked data to build web applications. *IEEE Internet Computing*, 13(4):68–73, 2009.

- [Het08] Erik Hetzner. A simple method for citation metadata extraction using hidden markov models. In *JCDL '08: Proceedings of the 8th ACM/IEEE-CS joint conference on Digital libraries*, pages 280–284, New York, NY, USA, 2008. ACM.
- [HF04] Nicholas Holden and Alex A. Freitas. Web page classification with an ant colony algorithm nicholas holden and alex a. freitas. In *Parallel Problem Solving from Nature - PPSN VIII, LNCS 3242*, pages 1092–1102. Springer-Verlag, 2004.
- [HGZ⁺04] Hui Han, Lee Giles, Hongyuan Zha, Cheng Li, and Kostas Tsioutsoulis. Two supervised learning approaches for name disambiguation in author citations. In *JCDL '04: Proceedings of the 4th ACM/IEEE-CS joint conference on Digital libraries*, pages 296–305, New York, NY, USA, 2004. ACM.
- [HHRH08] Michael Hausenblas, Wilhelm Halb, Yves Raimond, and Tom Heath. What is the size of the semantic web? In *I-Semantics 2008: International Conference on Semantic Systems*, Graz, Austria, 2008.
- [HPB⁺04] Ian Horrocks, Peter, Harold Boley, Said, and Mike Dean. Swrl: A semantic web rule language combining owl and ruleml, May 2004.
- [HR05] George Hripcsak and Adam S Rothschild. Agreement, the f-measure, and reliability in information retrieval. *Journal of American Medical Informatics Association*, 12(3):296–298, 2005.
- [HRH08] Wolfgang Halb, Yves Raimond, and Michael Hausenblas. Building linked data for both humans and machines. In *Linked Data on the Web (LDOW2008)*, 2008.
- [HRT⁺08] Jennefer Hart, Charlene Ridley, Faisal Taher, Corina Sas, and Alan Dix. Exploring the facebook experience: a new approach to usability. In *NordiCHI '08: Proceedings of the 5th Nordic conference on Human-computer interaction*, pages 471–474, New York, NY, USA, 2008. ACM.
- [HSW07] Harry Halpin, Brian Suda, and Norman Walsh. An ontology for vcards. Technical report, World Wide Web Consortium, 2007.
- [HZG05] Hui Han, Hongyuan Zha, and C. Lee Giles. Name disambiguation in author citations using a k-way spectral clustering method. In *JCDL '05: Proceedings of the 5th ACM/IEEE-CS joint conference on Digital libraries*, pages 334–343, New York, NY, USA, 2005. ACM.
- [HZS06] Jiayuan Huang, Tingshao Zhu, and Dale Schuurmans. Web communities identification from random walks. In *PKDD*, pages 187–198, 2006.
- [IS09] Antoine Isaac and Ed Summers. Skos simple knowledge organization system primer. Technical report, W3C, 2009.
- [IXZ07] Jose Iria, Lei Xia, and Ziqi Zhang. Wit: Web people search disambiguation using random walks. In *Proceedings of 4th International Workshop on Semantic Evaluations*, Prague, Czech Republic, 2007.

- [Jef89] Richard Jeffrey. *Formal Logic*. McGraw-Hill, 1989.
- [JGM08] Afraz Jaffri, Hugh Glaser, and Ian Millard. Uri disambiguation in the context of linked data. In *Linked Data on the Web Workshop, International World Wide Conference*, 2008.
- [JWA⁺09] Lili Jiang, Jianyong Wang, Ning An, Shengyuan Wang, Jian Zhan, and Lian Li. Two birds with one stone: A graph-based framework for disambiguating and tagging people names in web search. In *18th International World Wide Web Conference*, April 2009.
- [Kay07] Michael Kay. Xsl transformations (xslt) version 2.0 (w3c recommendation 23 january 2007). Technical report, W3C, 2007.
- [KCMNT08] Dmitri V. Kalashnikov, Zhaoqi Chen, Sharad Mehrotra, and Rabia Nuray-Turan. Web people search via connection analysis. *IEEE Transactions on Knowledge and Data Engineering*, 20(11):1550–1565, 2008.
- [Kha06] Rohit Khare. Microformats: The next (small) thing on the semantic web? *IEEE Internet Computing*, 10(1):68–75, 2006.
- [KM01] Svetlana Kiritchenko and Stan Matwin. Email classification with co-training. In *CASCON '01: Proceedings of the 2001 conference of the Centre for Advanced Studies on Collaborative research*, page 8. IBM Press, 2001.
- [KNTM08] Dmitri V. Kalashnikov, Rabia Nuray-Turan, and Sharad Mehrotra. Towards breaking the quality curse.: a web-querying approach to web people search. In *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 27–34, New York, NY, USA, 2008. ACM.
- [Koz06] Zornitsa Kozareva. Bootstrapping named entity recognition with automatically generated gazetteer lists. In *EACL '06: Proceedings of the Eleventh Conference of the European Chapter of the Association for Computational Linguistics: Student Research Workshop*, pages 15–21, Morristown, NJ, USA, 2006. Association for Computational Linguistics.
- [KP03] Harold Kundell and Marcia Polansky. Measurement of observer agreement. *Radiology*, 228(2):303–308, 2003.
- [LBGT05] Levon Lloyd, Varun Bhagwan, Daniel Gruhl, and Andrew Tomkins. Disambiguation of references to individuals. Technical report, HP, 2005.
- [LES06] Cliff Lampe, Nicole Ellison, and Charles Steinfield. A face(book) in the crowd: social searching vs. social browsing. In *CSCW '06: Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work*, pages 167–170, New York, NY, USA, 2006. ACM.
- [LL03] Xiaoli Li and Bing Liu. Learning to classify texts using positive and unlabeled data. In *IJCAI'03: Proceedings of the 18th international joint conference on Artificial intelligence*, pages 587–592, San Francisco, CA, USA, 2003. Morgan Kaufmann Publishers Inc.

- [Lov93] Laszlo Lovasz. Random walks on graphs: A survey, 1993.
- [LZ06] Sheng Liu and Jian Zhang. Retrieving and matching rdf graphs by solving the satisfiability problem. In *WI '06: Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 510–513, Washington, DC, USA, 2006. IEEE Computer Society.
- [LZL⁺09] Man Lan, Yu Zhe Zhang, Yue Lu, Jian Su, and ChewLim Tan. Which who are they? people attribute extraction and disambiguation in web search results. In *2nd Web People Search Evaluation Workshop (WePS 2009), 18th WWW Conference*, 2009.
- [Mal05] Bradley Malin. Unsupervised name disambiguation via social network similarity. In *Workshop on Link Analysis, Counterterrorism, and Security*, 2005.
- [Mar99] Gary T. Marx. What’s in a name? some reflections on the sociology of anonymity. *The Information Society*, 15(2), 1999.
- [mAYLL⁺09] Ching man Au Yeung, Ilaria Liccardi, Kanghao Lu, Oshani Seneviratne, and Tim Berners-Lee. Decentralization: The future of online social networking. In *W3C Workshop on the Future of Social Networking Position Papers*, 2009.
- [Mih04] Rada Mihalcea. Co-training and self-training for word sense disambiguation. In *CoNLL-2004*, 2004.
- [Mit97] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.
- [MLK06] Alex Markov, Mark Last, and Abraham Kandel. Model-based classification of web documents represented by graphs. In *WEBKDD*, Philadelphia, Pennsylvania, USA., 2006.
- [MM04] Frank Manola and Eric Miller, editors. *RDF Primer*. W3C Recommendation. World Wide Web Consortium, February 2004.
- [MMZ09] Peter Mika, Edgar Meij, and Hugo Zaragoza. Investigating the semantic gap through query log analysis. In *8th International Semantic Web Conference (ISWC2009)*, October 2009.
- [MNU00] Andrew McCallum, Kamal Nigam, and Lyle H. Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. In *KDD '00: Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 169–178, New York, NY, USA, 2000. ACM.
- [Mos03] Alessandro Moschitti. A study on optimal parameter tuning for rocchio text classifier. In *Proceedings of the 25th European Conference on Information Retrieval Research (ECIR'03)*, pages 420–435. Springer Verlag, 2003.
- [MRS02] Christoph Müller, Stefan Rapp, and Michael Strube. Applying co-training to reference resolution. In *ACL '02: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 352–359, Morristown, NJ, USA, 2002. Association for Computational Linguistics.

- [MS10] Mary Madden and Aaron Smith. Reputation management and social media. Technical report, Pew Research Centre, 2010.
- [MT93] Sean P. Meyn and Richard L. Tweedie. *Markov chains and stochastic stability*. Springer-Verlag, 1993.
- [MY02] Larry M. Manevitz and Malik Yousef. One-class svms for document classification. *Journal of Machine Learning Research*, 2:139–154, 2002.
- [NG00] Kamal Nigam and Rayid Ghani. Analyzing the effectiveness and applicability of co-training. In *CIKM '00: Proceedings of the ninth international conference on Information and knowledge management*, pages 86–93, New York, NY, USA, 2000. ACM.
- [NGM08] Natalya F. Noy, Nicholas Griffith, and Mark A. Musen. Collecting community-based mappings in an ontology repository. In *ISWC '08: Proceedings of the 7th International Conference on The Semantic Web*, pages 371–386, Berlin, Heidelberg, 2008. Springer-Verlag.
- [NL10] Roberto Navigli and Mirella Lapata. An experimental study of graph connectivity for unsupervised word sense disambiguation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32:678–692, 2010.
- [NMTM00] Kamal Nigam, Andrew Kachites McCallum, Sebastian Thrun, and Tom Mitchell. Text classification from labeled and unlabeled documents using em. *Mach. Learn.*, 39(2-3):103–134, 2000.
- [NS00] Deborah Nolan and Terry Speed. *Stat labs: mathematical statistics through applications*. Springer, 2000.
- [OL07] Byung-Won On and Dongwon Lee. Scalable name disambiguation using multi-level graph partition. In *Proceedings of the Seventh SIAM International Conference on Data Mining, April 26-28, 2007, Minneapolis, Minnesota, USA, 2007*.
- [OM05] Marek Obitko and Vladimír Marík. Integrating transportation ontologies using semantic web languages. In *Second International Conference on Industrial Applications of Holonic and Multi-Agent Systems*, pages 99–110, 2005.
- [O’R05] Tim O’Reilly. O’reilly network: What is web 2.0, September 2005.
- [OY08] Seog-Chan Oh and Shang-Tae Yee. Manufacturing interoperability using a semantic mediation. *The International Journal of Advanced Manufacturing Technology*, 39(1-2):199–210, October 2008.
- [Pas08] Alexandre Passant. :me owl:sameAs flickr:33669349@N00. In *Proceedings of the WWW 2008 Workshop Linked Data on the Web (LDOW2008), Beijing, China, Apr 2008*.
- [PDGM08] Panagiotis Papadimitriou, Ali Dasdan, and Hector Garcia-Molina. Web graph similarity for anomaly detection (poster). In *WWW '08: Proceeding of the*

- 17th international conference on World Wide Web*, pages 1167–1168. ACM, 2008.
- [Pee03] Joachim Peer. A logic programming approach to rdf document and query transformation. In *Knowledge Transformation for the Semantic Web*, pages 122–136, 2003.
- [PS06] Eric Prud’hommeaux and Andy Seaborne. SPARQL Query Language for RDF. Technical report, W3C, 2006.
- [QCj93] Ross J. Quinlan and Mike J. Cameron-jones. Foil: A midterm report. In *Proceedings of the European Conference on Machine Learning*. Springer-Verlag, 1993.
- [QH07] Huaijun Qiu and Edwin R. Hancock. Clustering and embedding using commute times. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 29(11):1873–1890, 2007.
- [QSGP04] Gang Qian, Shamik Sural, Yuelong Gu, and Sakti Pramanik. Similarity between euclidean and cosine angle distance for nearest neighbor queries. In *SAC ’04: Proceedings of the 2004 ACM symposium on Applied computing*, pages 1232–1237, New York, NY, USA, 2004. ACM.
- [Qui83] Ross J. Quinlan. Learning efficient classification procedures and their application to chess end-games. *Machine Learning: An Artificial Intelligence Approach*, 1983.
- [RB01] Erhard Rahm and Philip A. Bernstein. A survey of approaches to automatic schema matching. *The VLDB Journal*, 10(4):334–350, 2001.
- [RHS05] Chuck Rosenberg, Martial Hebert, and Henry Schneiderman. Semi-supervised self-training of object detection models. In *Seventh IEEE Workshop on Applications of Computer Vision*, pages 29–36, 2005.
- [Ris01] Irina Rish. An empirical study of the naive bayes classifier. In *IJCAI-01 workshop on "Empirical Methods in AI"*, 2001.
- [RJ99] Ellen Riloff and Rosie Jones. Learning dictionaries for information extraction by multi-level bootstrapping. In *AAAI ’99/IAAI ’99: Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence conference innovative applications of artificial intelligence*, pages 474–479, Menlo Park, CA, USA, 1999. American Association for Artificial Intelligence.
- [RSS08] Yves Raimond, Christopher Sutton, and Mark Sandler. Automatic inter-linking of music datasets on the semantic web. In *Linked Data on the Web Workshop, International World Wide Web Conference*, Apr 2008.
- [RYGR10] Suju Rajan, Dragomir Yankov, Scott Gaffney, and Adwait Ratnaparkhi. A large scale active learning system for topical categorization on the web. In *International World Wide Web Conference*, April 2010.

- [SB08] Christian D. Sigg and Joachim M. Buhmann. Expectation-maximization for sparse and non-negative pca. In *Proceedings of the International Conference on Machine learning*, pages 960–967, New York, NY, USA, 2008. ACM.
- [SBF⁺08] Lian Shi, Diego Berrueta, Sergio Fernandez, Luis Polo, and Silvino Fernandez. Smushing rdf instances: are alice and bob the same open source developer? In *ISWC2008 workshop on Personal Identification and Collaborations: Knowledge Mediation and Extraction (PICKME 2008)*, October 2008.
- [Sch98] Bernhard Scholkopf. Support vector machines: A practical consequence of learning theory. *IEEE Intelligent Systems*, 13(4), July/August 1998.
- [SCV07] Leo Sauermann, Richard Cyganiak, and Max Voelkel. Cool uris for the semantic web. Technical report, Deutsches Forschungszentrum fuer Kuenstliche Intelligenz GmbH, 2007.
- [SFYD04] Marco Saerens, Francois Fouss, Luh Yen, and Pierre Dupont. The principal components analysis of a graph, and its relationships to spectral clustering. In *Proceedings of the 15th European Conference on Machine Learning (ECML 2004)*. *Lecture Notes in Artificial Intelligence*, pages 371–383. Springer-Verlag, 2004.
- [SHC⁺07] Yang Song, Jian Huang, Isaac G. Councill, Jia Li, and C. Lee Giles. Efficient topic-based unsupervised name disambiguation. In *JCDL '07: Proceedings of the 7th ACM/IEEE-CS joint conference on Digital libraries*, pages 342–351, New York, NY, USA, 2007. ACM.
- [SHH02] Heiner Stuckenschmidt, Jens Hartmann, and Frank van Harmelen. Learning structural classification rules for web-page categorization. In *Proceedings of the Fifteenth International Florida Artificial Intelligence Research Society Conference*, pages 440–444. AAAI Press, 2002.
- [SJ02] Martin Szummer and Tommi Jaakkola. Partially labeled classification with markov random walks. In *Advances in Neural Information Processing Systems*, pages 945–952. MIT Press, 2002.
- [SJW97] Karen Sparck Jones and Peter Willett. *Readings in information retrieval*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997.
- [SLBK03] Adam Schenker, Mark Last, Horst Bunke, and Abraham Kandel. Classification of web documents using a graph model. *Document Analysis and Recognition, International Conference on*, 1:240, 2003.
- [SM00] Sean Slattery and Tom Mitchell. Discovering test set regularities in relational domains. In *Proceedings of International Conference on Machine Learning*, pages 895–902. Morgan Kaufmann, 2000.
- [SMB⁺08] Andy Seaborne, Geetha Manjunath, Chris Bizer, John Breslin, Souripriya Das, Ian Davis, Steve Harris, Kingsley Idehen, Olivier Corby, Kjetil Kjernsmo, and Benjamin Nowack. SPARQL/Update A language for

- updating RDF graphs. W3C Member Submission, W3C, July 2008. <http://www.w3.org/Submission/2008/04/>.
- [SMR99] Kristie Seymore, Andrew McCallum, and Ronald Rosenfeld. Learning hidden markov model structure for information extraction. In *AAAI 99 Workshop on Machine Learning for Information Extraction*, pages 37–42, 1999.
 - [SPR09] Fatiha Saïs, Nathalie Pernelle, and Marie-Christine Rousset. Combining a logical and a numerical method for data reconciliation. *Journal on Data Semantics XII*, pages 66–94, 2009.
 - [SPST⁺99] Bernhard Scholkopf, John Platt, John Shawe-Taylor, Anthony Smola, and R C Williamson. Estimating the support of a high-dimensional distribution. Technical report, Microsoft Research, 1999.
 - [SRWE08] K. Subrahmanyam, S. Reich, N. Waechter, and G. Espinoza. Online and offline social networks: Use of social networking sites by emerging adults. *Journal of Applied Developmental Psychology*, 29(6):420–433, November 2008.
 - [SS83] Carl E. Speicher and Jack W. Smith. *Choosing Effective Laboratory Tests*. WB Saunders, 1983.
 - [SSK05] Giorgos Stoilos, Giorgos Stamou, and Stefanos Kollias. A string metric for ontology alignment. In Yolanda Gil, Enrico Motta, V. Richard Benjamins, and Mark A. Musen, editors, *International Semantic Web Conference*, volume 3729 of *Lecture Notes in Computer Science*, pages 624–637, Berlin/Heidelberg, 2005. Springer-Verlag.
 - [SvED07] Leo Sauermann, Ludger van Elst, and Andreas Dengel. Pimo - a framework for representing personal information models. In K. Tochtermann, W. Haas, F. Kappe, A. Scharl, T. Pellegrini, and S. Schaffert, editors, *Proceedings of I-MEDIA '07 and I-SEMANTICS '07*, 2007.
 - [SWM04] Michael Smith, Chris Welty, and Deborah McGuinness. OWL web ontology guide., 2004.
 - [Tar72] Robert Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972.
 - [TCS⁺05] Dussadee Thamvijit, Hutchatai Chanlekha, Chalermpon Sirigayon, Trajul Permpool, and Asanee Kawtrakul. Person information extraction from the web. In *6th Symposium on 6th Symposium of Natural Language Processing*, 2005.
 - [TR02] Michael Thelen and Ellen Riloff. A bootstrapping method for learning semantic lexicons using extraction pattern contexts. In *EMNLP '02: Proceedings of the ACL-02 conference on Empirical methods in natural language processing*, pages 214–221, Morristown, NJ, USA, 2002. Association for Computational Linguistics.

- [TT03] Yoshimasa Tsuruoka and Jun'ichi Tsujii. Training a naive bayes classifier via the em algorithm with a class distribution constraint. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003*, pages 127–134, Morristown, NJ, USA, 2003. Association for Computational Linguistics.
- [vR79] Cornelis Joost van Rijsbergen. *Information Retrieval*. Butterworths, London, 2nd edition, 1979.
- [W3C07] W3C. Gleaning resource descriptions from dialects of languages (GRDDL). W3c recommendation, W3C, September 2007.
- [WBC⁺09] Evan Welbourne, Leilani Battle, Garret Cole, Kayla Gould, Kyle Rector, Samuel Raymer, Magdalena Balazinska, and Gaetano Borriello. Building the internet of things using rfid: The rfid ecosystem experience. *IEEE Internet Computing*, 13(3):48–55, 2009.
- [WBMM09] Keigo Watanabe, Danushka Bollegala, Yutaka Matsuo, and Mitsuru Ishizuka. A two-step approach to extracting attributes for people on the web. In *2nd Web People Search Evaluation Workshop (WePS 2009), World Wide Web Conference*, 2009.
- [WF94] Stanley Wasserman and Katherine Faust. *Social network analysis : methods and applications*. Structural analysis in the social sciences, 8. Cambridge University Press, 1 edition, November 1994.
- [WGLD05] Xiaojun Wan, Jianfeng Gao, Mu Li, and Binggong Ding. Person resolution in person search results: Webhawk. In *CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 163–170, New York, NY, USA, 2005. ACM.
- [Win05] Phillip Windley. *Digital Identity*. O'Reilly Media, Inc., August 2005.
- [WMPH04] Ben Wellner, Andrew McCallum, Fuchun Peng, and Michael Hay. An integrated, conditional model of information extraction and coreference with application to citation matching. In *UAI '04: Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pages 593–601, Arlington, Virginia, United States, 2004. AUAI Press.
- [WYZ07] Gang Wang, Yong Yu, and Haiping Zhu. Pore: Positive-only relation extraction from wikipedia text. In *Proceedings of the International and Asian Semantic Web Conference*, pages 580–594, 2007.
- [Yar95] David Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 189–196, 1995.
- [YFD⁺07] Luh Yen, Francois Fouss, Christine Decaestecker, Pascal Francq, and Marco Saerens. Graph nodes clustering based on the commute-time kernel. In *Proceedings of the 11th Pacific-Asia Conference on Knowledge Discovery and*

Data Mining (PAKDD 2007). Lecture notes in Computer Science, LNCS, 2007.

- [YFD⁺09] Luh Yen, Francois Fouss, Christine Decaestecker, Pascal Francq, and Marco Saerens. Graph nodes clustering with the sigmoid commute-time kernel: A comparative study. *Data and Knowledge Engineering*, 68(3):338 – 361, 2009.
- [YHeC04] Hwanjo Yu, Jiawei Han, and Kevin Chen chuan Chang. Pebl: Web page classification without negative examples. *IEEE Transactions on Knowledge and Data Engineering*, 16:70–81, 2004.
- [YVW⁺05] Luh Yen, Denis Vanvyve, Fabien Wouters, François Fouss, Michel Verleysen, and Marco Saerens. Clustering using a random walk based distance measure, 2005.
- [ZLT07] Jie Zou, Daniel Le, and George R. Thoma. Structure and content analysis for html medical articles: a hidden markov model approach. In *DocEng '07: Proceedings of the 2007 ACM symposium on Document engineering*, pages 199–201, New York, NY, USA, 2007. ACM.
- [ZLY⁺08] Baoyao Zhou, Wei Liu, Yu Yang, Weichun Wang, and Ming Zhang. Effective metadata extraction from irregularly structured web content. Technical report, HP Laboratories, 2008.

Appendices

Appendix A

Graph Reasoning Algorithm

This appendix contains the algorithm in pseudocode form of the graph reasoning technique presented within chapter 5. Person instances from separate graphs are compared for a match.

Algorithm 10 $\text{sim}(g, h)$: Compares two instances of *foaf:Person* represented as graphs (g and h) for similarity based on information within the graphs. Returns *true* if the instances are equivalent, otherwise *false*.

Input: g and h

Output: *true* or *false*

```
1:  $n_g = \text{foaf:name}$  from  $g$ 
2:  $n_h = \text{foaf:name}$  from  $h$ 
3: if  $\text{strsim}(n_g, n_h) > \sigma$  then
4:    $mbox_g = \text{foaf:mbox}$  from  $g$ 
5:    $mbox_h = \text{foaf:mbox}$  from  $h$ 
6:   if  $mbox_g == mbox_h$  then
7:     return true
8:   end if
9:    $url_g = \text{foaf:homepage}$  from  $g$ 
10:   $url_h = \text{foaf:homepage}$  from  $h$ 
11:  if  $url_g == url_h$  then
12:    return true
13:  end if
14:   $geo_g = \text{geo:Feature}$  from  $g$ 
15:   $geo_h = \text{geo:Feature}$  from  $h$ 
16:  if  $geo_g == geo_h$  then
17:    return true
18:  else
19:     $loc_g = \text{geo:name}$  from  $g$ 
20:     $loc_h = \text{geo:name}$  from  $h$ 
21:    if  $\text{checkLocations}(loc_g, loc_h)$  then
22:      return true
23:    else
24:      return  $\text{checkLocations}(loc_h, loc_g)$ 
25:    end if
26:  end if
27: end if
```

Algorithm 11 `checkLocations( $loc_g, loc_h$ )` : Takes two locations loc_g and loc_h and checks for a relation between them. If one exists then *true* is returned, otherwise *false*.

Input: loc_g and loc_h

Output: *true* or *false*

```

1: relation = query ( $\langle loc_g \rangle, \mathbf{relation}, \langle loc_h \rangle$ )
2: if relation == (dbprop:metropolitanBorough || dbprop:redirect) then
3:   return true
4: else
5:   return false
6: end if

```

Appendix B

Linked Data Social Graph

Below is a snippet from a produced Linked Data social graph. It combines digital identity information exported from Facebook and Twitter into an interlinked RDF model. Individual social graphs were exported using the techniques described in §5.5 and combined into a Linked Data social graph using the technique presented in §5.6.

```
<?xml version="1.0" encoding="utf-8" ?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:foaf="http://xmlns.com/foaf/0.1/"
  xmlns:owl="http://www.w3.org/2002/07/owl#">
  <foaf:Person rdf:ID="me">
    <foaf:name>Matthew Rowe</foaf:name>
    <owl:sameAs rdf:resource="http://namespace.com/fb.rdf#me"/>
    <owl:sameAs rdf:resource="http://namespace.com/twitter.rdf#me"/>
    <owl:sameAs rdf:resource="http://namespace.com/myspace.rdf#me"/>
    <foaf:knows>
      <foaf:Person rdf:about="#harryhalpin">
        <foaf:name>Harry Halpin</foaf:name>
        <owl:sameAs rdf:resource="http://namespace.com/fb.rdf#1047709463"/>
        <owl:sameAs rdf:resource="http://namespace.com/twitter.rdf#harryhalpin"/>
      </foaf:Person>
    </foaf:knows>
    <foaf:knows>
      <foaf:Person rdf:about="#SamChapman">
        <foaf:name>Sam Chapman</foaf:name>
        <owl:sameAs rdf:resource="http://namespace.com/fb.rdf#617555567"/>
        <owl:sameAs rdf:resource="http://namespace.com/twitter.rdf#samchapman"/>
      </foaf:Person>
    </foaf:knows>
  </foaf:Person>
</rdf:RDF>
```

Automated processes are able to *lookup* the URIs in the social graph and dereference the instance descriptions, returning a complete social graph containing the necessary details for each instance. Such a process is performed by each of the disambiguation techniques presented in chapters 7, 8 and 9. For clarity below is an example of the dereferenced graph from the above snippet:

```

<?xml version="1.0" encoding="utf-8" ?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:foaf="http://xmlns.com/foaf/0.1/"
  xmlns:geo="http://www.geonames.org/ontology">
  <foaf:Person rdf:ID="me">
    <foaf:name>Matthew Rowe</foaf:name>
    <foaf:homepage rdf:resource="http://www.dcs.shef.ac.uk/~mrowe"/>
    <foaf:mbox rdf:resource="m.rowe@dc.shef.ac.uk">
    <foaf:based_near>
      <geo:Feature rdf:about="http://www.geonames.org/2638077">
        <geo:name>Sheffield</geo:name>
        <geo:inCountry>UK</geo:inCountry>
      </geo:Feature>
    </foaf:based_near>
    <foaf:knows>
      <foaf:Person rdf:about="#harryhalpin">
        <foaf:name>Harry Halpin</foaf:name>
        <foaf:based_near>
          <geo:Feature rdf:about="http://www.geonames.org/33329">
            <geo:name>Edinburgh</geo:name>
            <geo:inCountry>UK</geo:inCountry>
          </geo:Feature>
        </foaf:based_near>
      </foaf:Person>
    </foaf:knows>
    <foaf:knows>
      <foaf:Person rdf:about="#SamChapman">
        <foaf:name>Sam Chapman</foaf:name>
        <foaf:homepage rdf:resource="http://www.dcs.shef.ac.uk/~sam"/>
        <foaf:based_near>
          <geo:Feature rdf:about="http://www.geonames.org/2638077">
            <geo:name>Sheffield</geo:name>
            <geo:inCountry>UK</geo:inCountry>
          </geo:Feature>
        </foaf:based_near>
      </foaf:Person>
    </foaf:knows>
  </foaf:Person>
</rdf:RDF>

```

Appendix C

Tuning Levenshtein Measure

The Levenshtein distance measure is used throughout the methods presented within this thesis to match person names. The normalised Levenshtein distance measure is used from the SimMetrics package [chapman:armadillo] which confirms a match between two strings when a threshold (σ) has been exceeded. Therefore this threshold must be derived based on the dataset used. The threshold was tuned using 10% of the dataset from §5.7.2 by creating a gold standard of person names which should be matched in either graph and increasing σ where $0 \leq \sigma \leq 1$. Given the gold standard the information retrieval measures precision, recall and f-measure were used to derive the optimum threshold value where the f-measure was maximised. The following plot presents the derivation of the threshold where $\sigma = 0.75$.

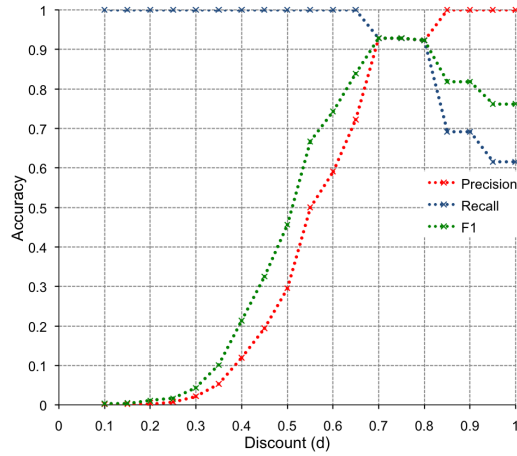


Figure C.1: Tuning Levenshtein distance for matching person names

Appendix D

Tuning Node/Edge Overlap

Chapter 5 presents several state of the art graph matching techniques applied to the task of interlinking distributed social graphs. Node/Edge Overlap (§5.2.2.1) confirms a match between two graphs when the Jaccard similarity exceeds a given threshold. Tuning the threshold (σ) was performed using the 10% dataset collected for the evaluation in chapter 5. The following graphs plots the False Positive Rate vs False Negative Rate, indicating that the best performing threshold ($\sigma = 0.7$) minimised both error rates and appears closest to the origin. The graph also indicates that tuning σ is sensitive to extremes of either False Positive or False Negative Rates at both higher (> 0.8) or lower (< 0.3) values for σ : At lower σ FNR is minimised, variability is only seen in FPR, whereas at higher σ FPR is minimised and variance is only seen in FNR.

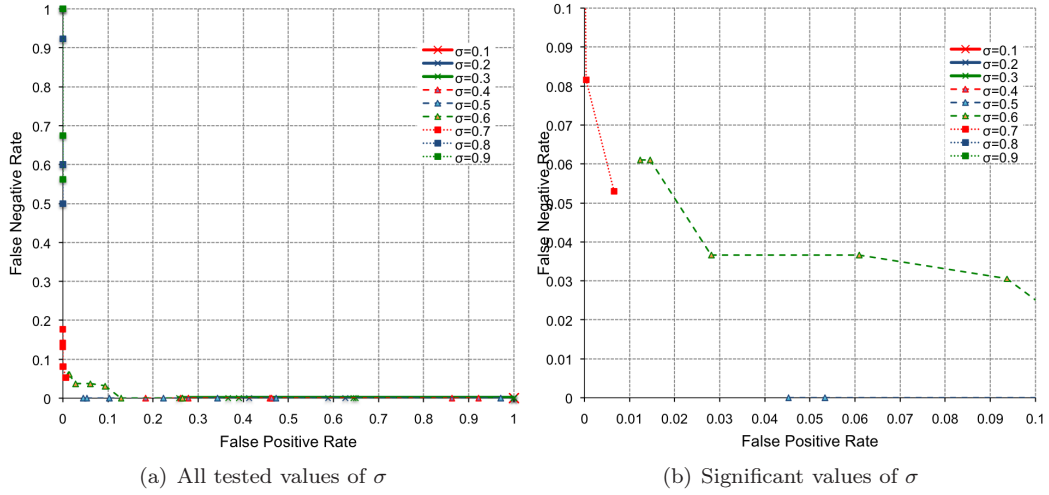


Figure D.1: False Positive Rate vs False Negative Rate when Tuning Node/Edge Overlap threshold

Appendix E

Tuning Node Mapping

Similar to Appendix D, Node Mapping (§5.2.2.2) is used as a state of the art graph matching technique for interlinking social graphs. Node Mapping confirms a match between two graphs when a mapping combination between two graph exceeds a given threshold (σ). Tuning of σ was performed using the 10% dataset collected for the evaluation in chapter 5. The following graphs plots the False Positive Rate (FPR) vs False Negative Rate (FNR) as the threshold (σ) is varied. The graph indicates that the optimum setting for σ is 0.75 where the curves appears closest to the origin.

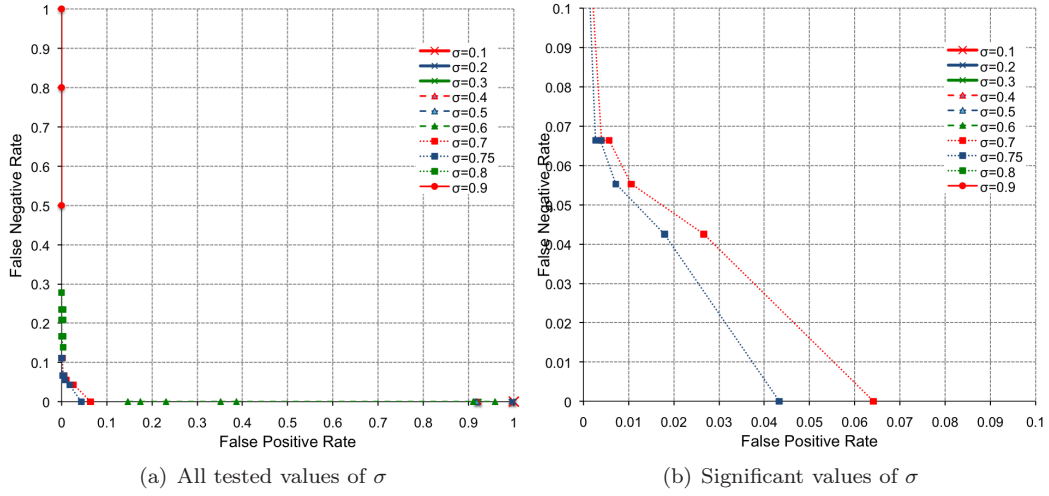


Figure E.1: False Positive Rate vs False Negative Rate when Tuning Node Mapping threshold

Appendix F

Person Attribute Patterns

This appendix contains the various patterns used to facilitate the matching of information describing different person attributes: name, email, and URL. These are as follows:

F.1 Name Patterns

```
<first-name> [A-Za-z]+  
<first-name> [A-Za-z]-[A-Za-z]  
<first-name> [A-Z]. [A-Za-z]+  
<first-name> [A-Z]. [A-Za-z]-[A-Za-z]  
[A-Za-z]+ ([A-Z].)+  
[A-Za-z]-[A-Za-z] ([A-Z].)+  
[A-Za-z]+, ([A-Z].)+  
[A-Za-z]-[A-Za-z], ([A-Z].)+  
[A-Za-z]+ ([A-Z].)+  
[A-Za-z]-[A-Za-z] ([A-Z].)+  
[A-Za-z]+, ([A-Z].)+  
[A-Za-z]-[A-Za-z], ([A-Z].)+  
([A-Z].)+ [A-Za-z]+  
([A-Z].)+ [A-Za-z]-[A-Za-z]  
([A-Z].)+ [A-Za-z]+  
([A-Z].)+ [A-Za-z]-[A-Za-z]
```

F.2 Email Patterns

```
[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,4}
```

F.3 URL Patterns

```
https?://([-\w.]+)+(:\d+)?(/[\\w/_-~.]*?(?\\S+)?)?
```

Appendix G

Experiments with Absolute Discounting

Chapter 6 of this thesis discusses the use of Hidden Markov Models using different smoothing methods from the literature. One such method, known as Absolute Discounting [FM99], discounts a decimal constant from non-zero probabilities and uses a uniform distribution coupled with the discount constant to smooth zero probabilities. The value of the discount d is such that $0 < d < 1$. Therefore a suitable value must be chosen for accurate smoothing. Figure G.1(a) presents results from a range of discount values. As $d \rightarrow 0$ all evaluation measures improved, therefore to explore the further effects of reducing d , the discount was set as $d = 10^{-n}$, $n = \{1, 2, 3, \dots, 8\}$. As these discounts are hard to see in Figure G.1(a), Figure G.1(b) shows the results in closer detail. As n increases F_1 peaks where $n = 4$ before reducing again.

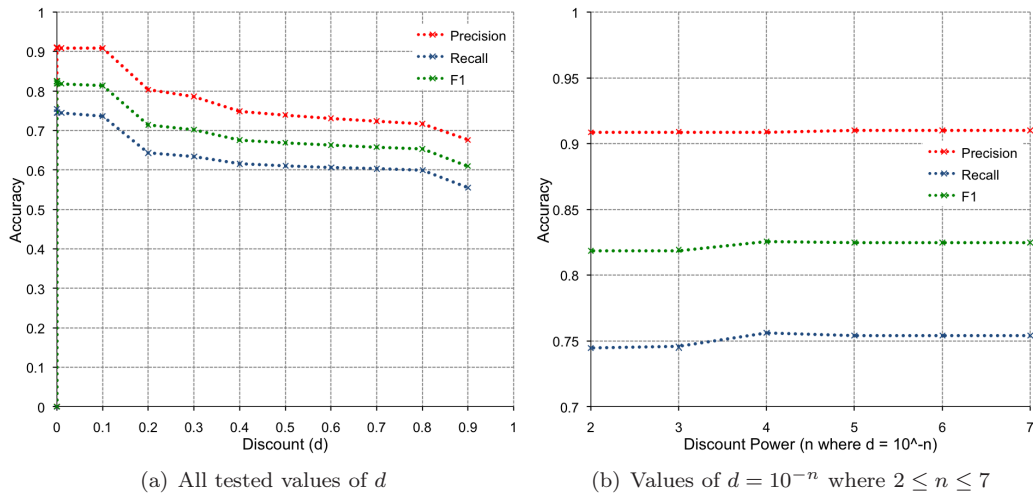


Figure G.1: Tuning Discount when smoothing using Absolute Discounting

Appendix H

Experiments using Interrater Gold Standard

The evaluation presented in chapter 10 uses a dataset containing ~ 346 web resources to be disambiguated for each participant. The scale and sheer number of web resources made labelling the dataset - and thus generating a gold standard - using multiple raters was simply not feasible - given that there are 50 participants who are evaluated. Instead 12 raters labelled a portion of the dataset - 50 web resources per participant which was then used as a gold standard to judge the reliability of the gold standard over the entire dataset - given that this was created by a single person. To provide additional evidence of the accuracy levels obtained by the disambiguation techniques described within this thesis, the experiments detailed in chapter 10 were performed using only a portion of the entire dataset and using the relevant gold standard for evaluation - derived from using majority opinion of web resource labels. The same experimental setup was also used from chapter 10. The results from these experiments are now presented.

Table H.1: Accuracy of Inference Rules with respect to Baseline techniques using GS'

	Precision	Recall	F_1	F_2	F_3
Inference Rules	0.95	0.444	0.566	0.487	0.464
Baseline 1	0.188	0.998	0.294	0.467	0.607
Baseline 2	0.242	0.742	0.278	0.390	0.489
Baseline 3	0.765	0.725	0.719	0.703	0.697
Baseline 4	0.259	0.888	0.310	0.439	0.546

Table H.2: Accuracy of applying Random Walks with Commute Time and Optimum Transitions with respect to Baseline techniques using GS'

	Precision	Recall	F_1	F_2	F_3
Commute Time	0.667	0.797	0.707	0.749	0.770
Optimum Transitions	0.645	0.807	0.701	0.752	0.777
Baseline 1	0.188	0.998	0.294	0.467	0.607
Baseline 2	0.242	0.742	0.278	0.390	0.489
Baseline 3	0.765	0.725	0.719	0.703	0.697
Baseline 4	0.259	0.888	0.310	0.439	0.546

Table H.3: Accuracy of Different Permutations of Machine Learning Classifier and Feature Similarity Measure with respect to Baseline techniques using GS'

		Precision	Recall	F_1	F_2	F_3
Perceptron	Entailment	0.660	0.881	0.731	0.804	0.837
	IFP	0.645	0.735	0.718	0.653	0.687
	Jaccard	0.683	0.796	0.666	0.729	0.756
SVM	Entailment	0.621	0.890	0.753	0.805	0.848
	IFP	0.631	0.733	0.692	0.652	0.686
	Jaccard	0.769	0.638	0.684	0.612	0.622
Naive Bayes	Entailment	0.638	0.667	0.651	0.639	0.651
	IFP	0.644	0.653	0.656	0.600	0.622
	Jaccard	0.696	0.587	0.545	0.559	0.570
	Baseline 1	0.188	0.998	0.294	0.467	0.607
	Baseline 2	0.242	0.742	0.278	0.390	0.489
	Baseline 3	0.765	0.725	0.719	0.703	0.697
	Baseline 4	0.259	0.888	0.310	0.439	0.546

Appendix I

Tuning Baseline 2: Hierarchical Clustering

The second baseline technique deployed within chapter 10 utilises an *unseeded* approach to disambiguation through the use of unsupervised clustering over web resources, where person names are employed as features of the web resources. For a given web resource a bag-of-words model is built composed of person names, extracted from the RDF model representation of the web resource using the following SPARQL query:

```
PREFIX foaf:<http://xmlns.com/foaf/0.1/>
SELECT ?name
WHERE {
    ?url foaf:topic ?p .
    ?p foaf:name ?name
}
```

In order to apply an agglomerative clustering strategy however, the clustering similarity threshold must be tuned on the training split of the dataset prior to applying it to the remaining 90% for testing. Experiments were performed by varying the threshold t between a range of values from 0 to 1 with increments of 0.1: $\sigma = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$. Figure I.1 shows the precision/recall curves for all the tuning thresholds, where $\sigma = 0.2$ and $\sigma = 0.3$ each achieve 0.33 for recall at precision 0.9. In the context of this thesis however recall is more important than precision, given that the majority of *identity web references* must be found in order to avoid personal information being discovered and reused (e.g., stealing the person's identity). Focussing on this requirement, Figure I.1 shows that using a threshold setting of $\sigma = 0.9$ achieves significantly higher levels for precision at recall 90% than the other threshold settings. Based on the results the threshold of $\sigma = 0.9$ is chosen.

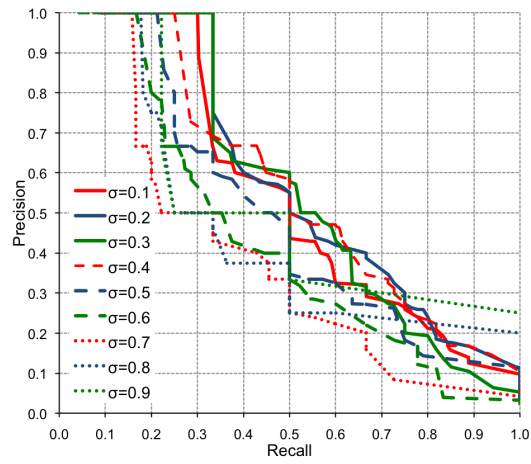


Figure I.1: Precision/Recall curve for various clustering thresholds for Hierarchical Clustering

Appendix J

Tuning Random Walks: Effects of Edge Weighting

The explanation of the Random Walks method within chapter 8 uses the context of *unique* and *normal* edges. The graph-space is compiled from metadata models describing the seed data and the collection of web resources to be analysed. Metadata models describe information in a machine-readable form using metadata descriptions linked to concepts from web accessible ontologies. This permits edges within the graph-space to be assigned weights based on the characteristics of the concepts which they denote. Within the literature the notion of edge weighing is briefly mentioned, but not extensively studied, in fact no heuristics are given for weighting edges, only a basic criteria is suggested that edges which denote uniqueness should carry more weight.

Edges within a graph-space are assigned a given label l_j from the set of labels L . Given that L contains labels for both unique and non-unique edges it can be divided into two subsets: $L = L^U \cup L^N$. Each edge in the graph is given a weight based on the concept it denotes and therefore its label from L using a weight function $w(l)$. A probability distribution is used such that the sum of all distinct edge weights is 1. Splitting the edge weights into weights for unique and non-unique edges using λ^U and λ^N respectively, allows the following equation to be derived (see chapter 8 for more details of its derivation):

$$\lambda^N \cdot (\eta \cdot |L^U| + |L^N|) = 1 \quad (\text{J.1})$$

Using Equation (J.1) allows $\eta = \frac{\lambda^U}{\lambda^N}$ to be defined as a proportion between the unique and non-unique edge label weights. To explore the effects of edge weighting on the resultant accuracy levels when performing Random Walks and clustering using Commute Time and Optimum Transitions different values of η ($\eta = \{\frac{1}{4}, \dots, 1, \dots, 4\}$) were tested when combined with different clustering thresholds for σ ($\sigma = \{0.1, \dots, 0.5, \dots, 0.9\}$). The experiments were performed over the 10% split of the dataset and the optimum setting for both σ and η were then used for experiments over the remaining 90%, as shown in chapter 10. Optimum parameters were identified as those which optimized precision at recall 90%, the following now documents the derivation of these parameters for both Commute Time distances and Optimum Transitions.

J.1 Commute Time

Figure J.1 shows the precision/recall curves obtained for settings of $\eta = \{\frac{1}{4}, 1, 4\}$ and $0.3 \leq \sigma \leq 0.9$ - it is worth noting that $\eta = \frac{1}{4}$ and $\eta = 4$ are included to show the extremes of setting η . At lower values of η normal edges are weighted more than unique edges, thereby leading to a decrease in precision. This is evident when comparing Figure J.1(a) with Figure J.1(c) where recall at precision 90% is lower. The goal of these experiments is to identify the parameters which will perform best over the remaining 90% test split of the dataset. Figure J.1(b) shows experiments performed where $\eta = 1$ for various threshold settings of σ , the graph identifies the use of $\sigma = 0.8$ as being the optimum parameter for this setting of η as it yields the highest level of precision at recall 90%. In comparison with the other experiments performed for different settings of η the curves identify different trends in performance, future work, as discussed in chapter 12, will utilise the same graph dataset as used in [YFD⁺07] to explore the effects of edge weights further on Commute Time distances.

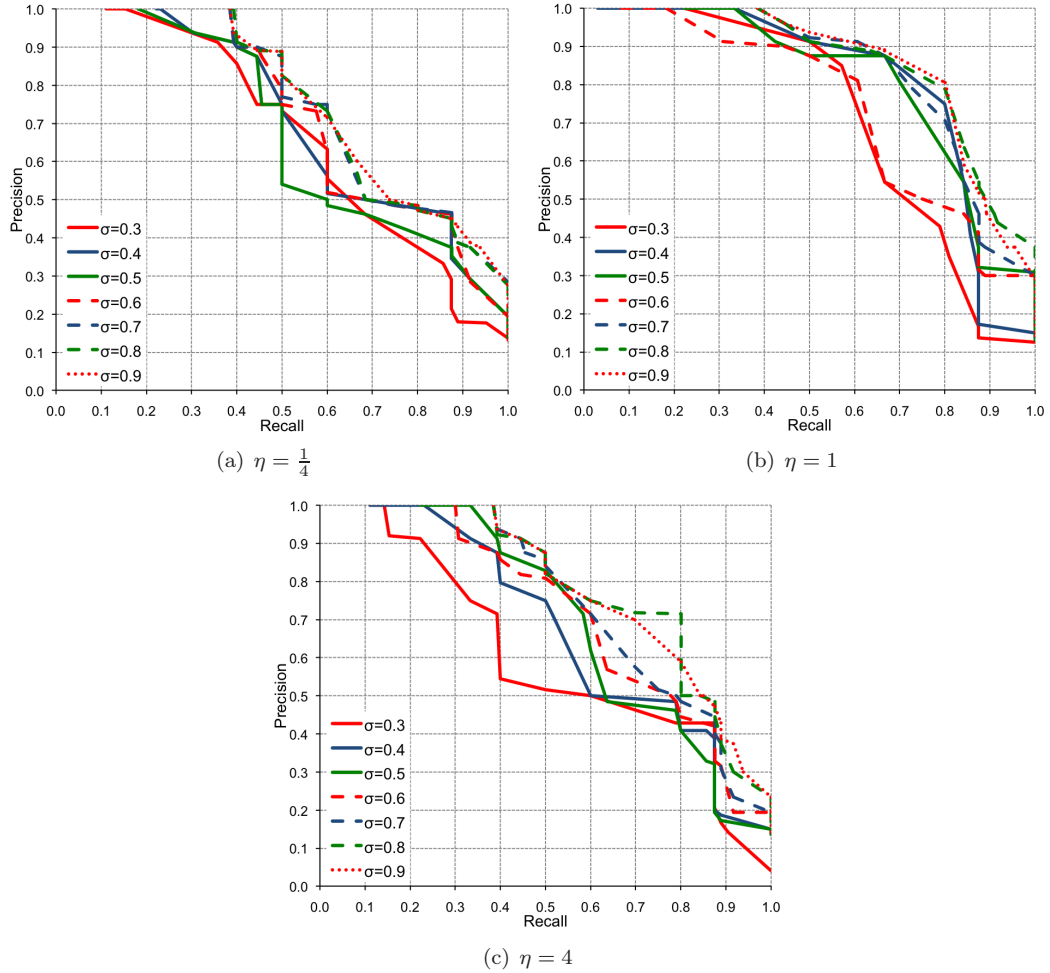


Figure J.1: Precision/Recall curves when tuning σ and η for Random Walks with Commute Time distances

J.2 Optimum Transitions

Experiments with values of η and σ with Optimum Transitions yields similar effects to Commute Time distances, as Figure J.2 demonstrates. As σ is increased recall increases when clustering, due to the inclusion of nodes that are further away and thus take longer to reach - by definition this implies that the dissimilarity between the nodes is therefore greater. As both experiments for Commute Time and Optimum Transitions demonstrate the effects of σ are greater on clustering than η . However there are noticeable differences in the curves produced for differing values of η , especially when comparing $\eta = \frac{1}{4}$ and $\eta = 4$. Assessment of the precision/recall curves in Figure J.2 identifies $\eta = 1$ and $\sigma = 0.8$ as being the parameters which optimise precision at recall 90%. These parameters were chosen for experiments over the remaining 90% of the dataset, discussion of these experiments is presented in chapter 10 of this thesis.

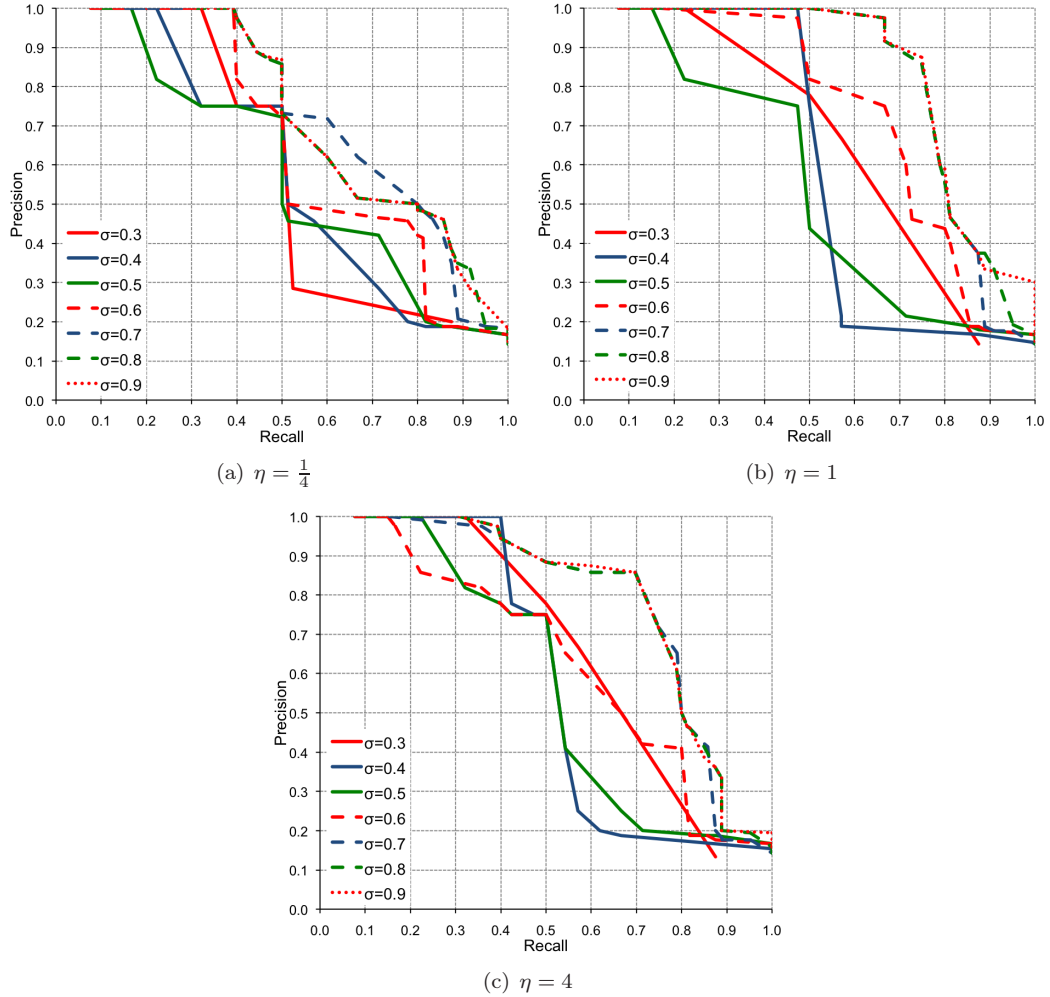


Figure J.2: Precision/Recall curves when tuning σ and η for Random Walks with Optimum Transitions

Appendix K

Tuning Rocchio Classification for Self-Training

This section details tuning experiments to derive an optimum setting for ρ when applying Rocchio classification for negative set generation. Self-training is applied in two settings in chapter 9: enlargement of the negative set using only the strongest negative (referred to hereafter as SNE for brevity) and enlargement of the negative training set using all samples classed as negative (referred to hereafter as FNE). Below describes the results and findings from experiments conducted using the 10% split of the dataset from chapter 10 for tuning the parameters. Values of ρ were selected that maximised precision at recall 90%.

K.1 Strongest-Negative Enlargement

SNE enlarges the negative training set following each self-training iteration with the negative labelled sample with the largest classification confidence. The following explains the findings obtained from testing different values of $\rho = \{\frac{1}{4}, \dots, 1, \dots, 4\}$ with the three different classifiers (Perceptron, SVM and Naive Bayes) coupled with the three differing feature similarity measures (Entailment, IFP and Jaccard).

K.1.1 Perceptron

Figure K.1 presents the precision/recall curves obtained for the different feature similarity measures when tested using a Perceptron classifier. Higher values of ρ provide greater feedback to the negative prototype vector than lower values, thereby reducing the likelihood of false negatives being produced - given that known positive features effect the placement of the negative set's hyperplane. Figure K.1(a) indicates that $\rho = 2$ optimises precision at recall 90% when Perceptron is combined with Entailment, Figure K.1(b) indicates that $\rho = 1$ achieves the optimum setting for precision at recall 90% when Perceptron is combined with IFP and Figure K.1(c) also indicates that Perceptron with Jaccard yields the best values for precision at recall 90% when $\rho = 1$.

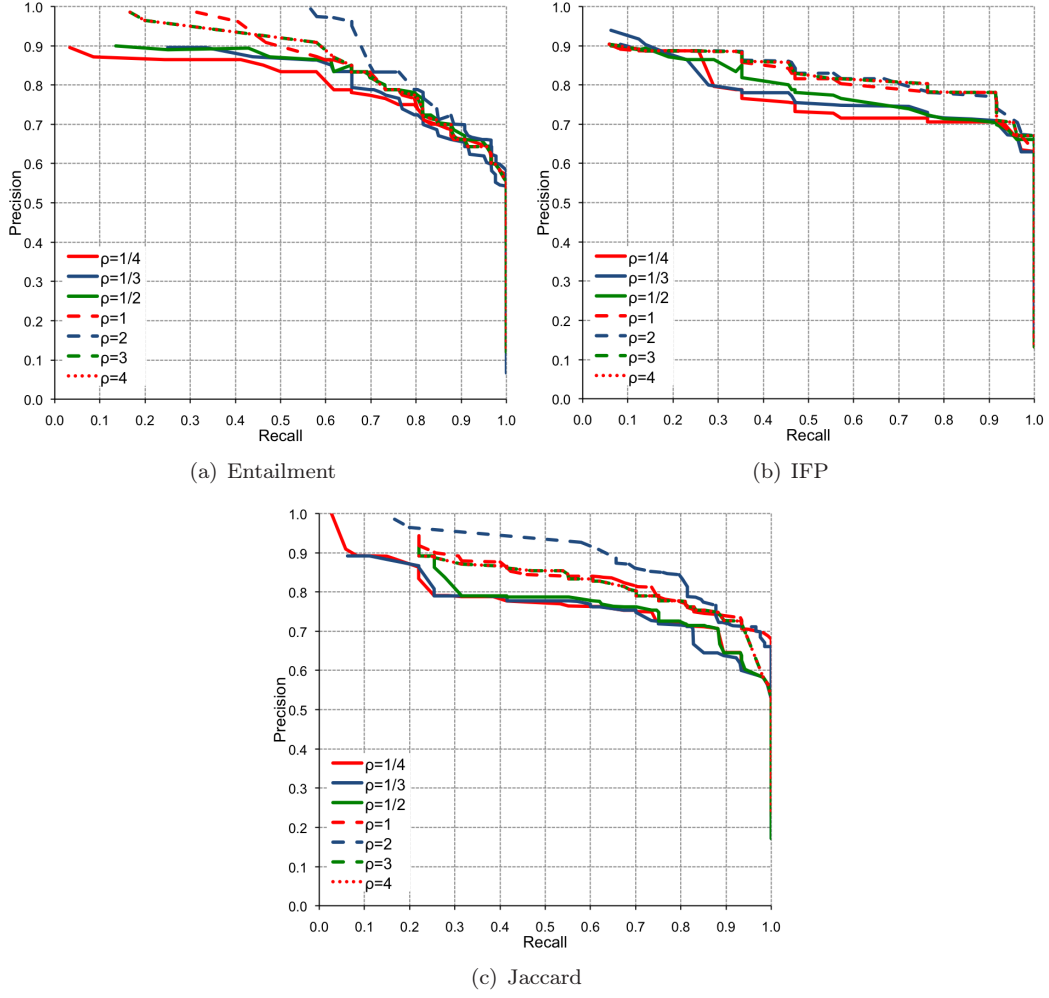


Figure K.1: Precision/Recall curves when tuning ρ for SNE Self-trained Perceptron with different feature similarity measures

K.1.2 SVM

Figure K.3 presents the tuning of SVM and the subsequent derived precision/recall curves. For SVM combined with Entailment, $\rho = 2$ yields the best setting for optimising precision at recall 90%, where this setting also increases recall at precision 90% over other parameter values. For SVM combined with IFP, Figure K.2(b) indicates that $\rho = 4$ achieves the best setting for optimising precision at recall 90%. Figure K.2(c) shows the precision/recall curves produced for SVM with Jaccard similarity, where $\rho = 2$ yields the optimum setting for generating negatives training data.

K.1.3 Naive Bayes

Self-training Naive Bayes with SNE also requires different settings of ρ for negative set generation when combined with different feature similarity measures. For Naives Bayes combined with Entailment $\rho = 1$ achieves the optimum value for precision at recall 90%, for Naive Bayes with IFP $\rho = 2$ performs best, and for Naive Bayes with Jaccard, $\rho = \frac{1}{4}$ yields the best performance.

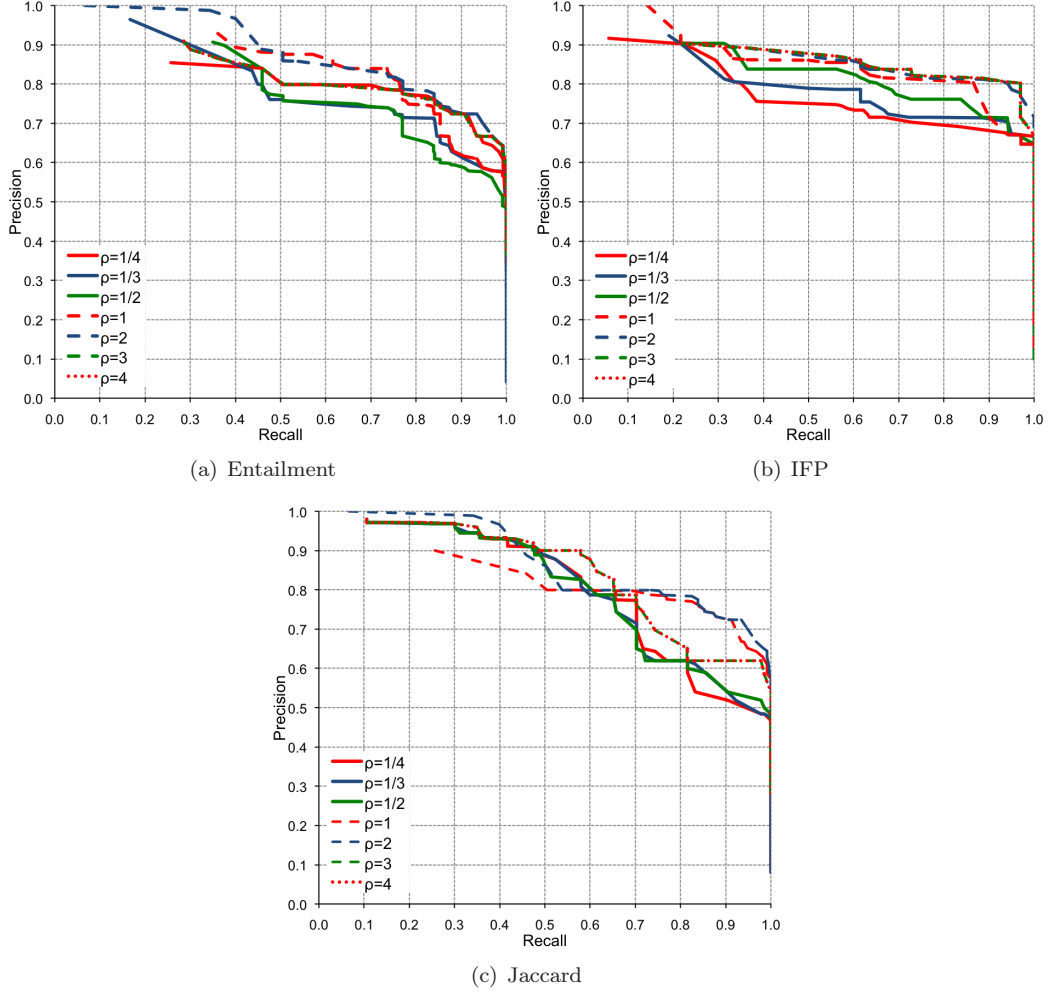


Figure K.2: Precision/Recall curves when tuning ρ for SNE Self-trained SVM with different feature similarity measures

K.2 Full-Negative Enlargement

Full Negative Enlargement (FNE) increases the negative training set with all examples which have been labelled as negative following each iteration of the Self-training process. In a similar manner to SNE of the training sets, a suitable value for ρ must be derived for different combinations of classifier and feature similarity measure. Derivation of this parameter was performed using experiments over a 10% split of the dataset, and the values of ρ was chosen which optimised precision at recall 90%. The following section details the results from these experiments, and the derived values for ρ .

K.2.1 Perceptron

Experiments with Perceptron and various feature similarity measures indicate high-precision levels for low levels of recall. As recall increases precision is reduced, resulting in, on average, low levels for precision at recall 90%. Figure K.4(a) indicates that $\rho = 2$ optimizes precision at recall 90% when Perceptron is combined with Entailment, for Perceptron with IFP precision at recall 90% is optimised using $\rho = 2$ also and similarly for Perceptron combined with

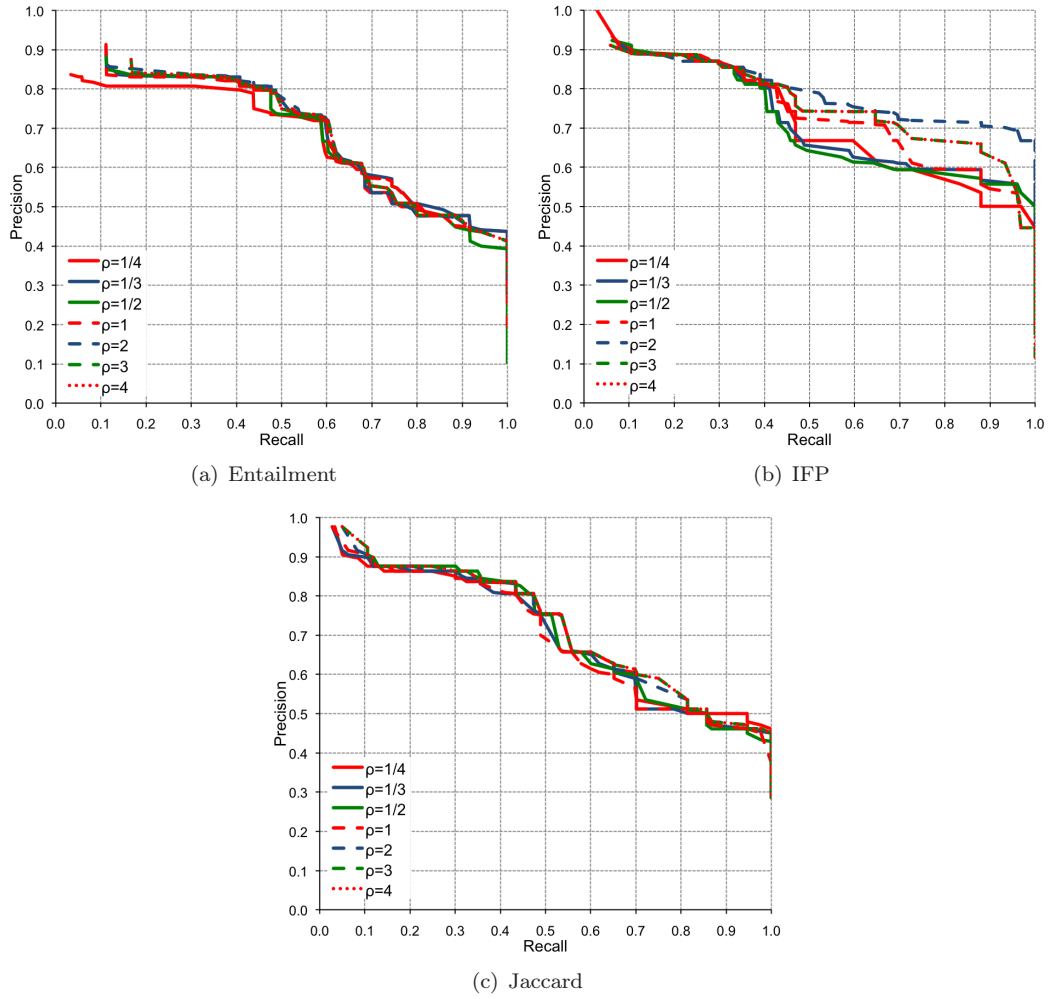


Figure K.3: Precision/Recall curves when tuning ρ for SNE Self-trained Naive Bayes with different feature similarity measures

Jaccard $\rho = 2$ also yields the optimum parameter setting. A notable difference between SNE Perceptron and FNE Perceptron are the recall levels achieved by each Self-training method. In the case of SNE precision levels remain high whilst recall increases, whereas enlargement of the negative set using all negative samples leads to a significant reduction in precision as recall increases.

K.2.2 SVM

When applying Self-training to Support Vector Machines, and enlarging the negative training set with all examples classed as negative, the combination with Entailment optimises precision at recall 90% where $\rho = 3$. Figure K.5(b) indicates that the same measure is optimised where $\rho = 2$ for SVM with IFP, and $\rho = \frac{1}{4}$ when combined with Jaccard.

K.2.3 Naive Bayes

Figure K.6 presents the precision/recall curves produced from tuning the parameter ρ for negative set generation with the Naive Bayes classifier. The combination of Naive Bayes

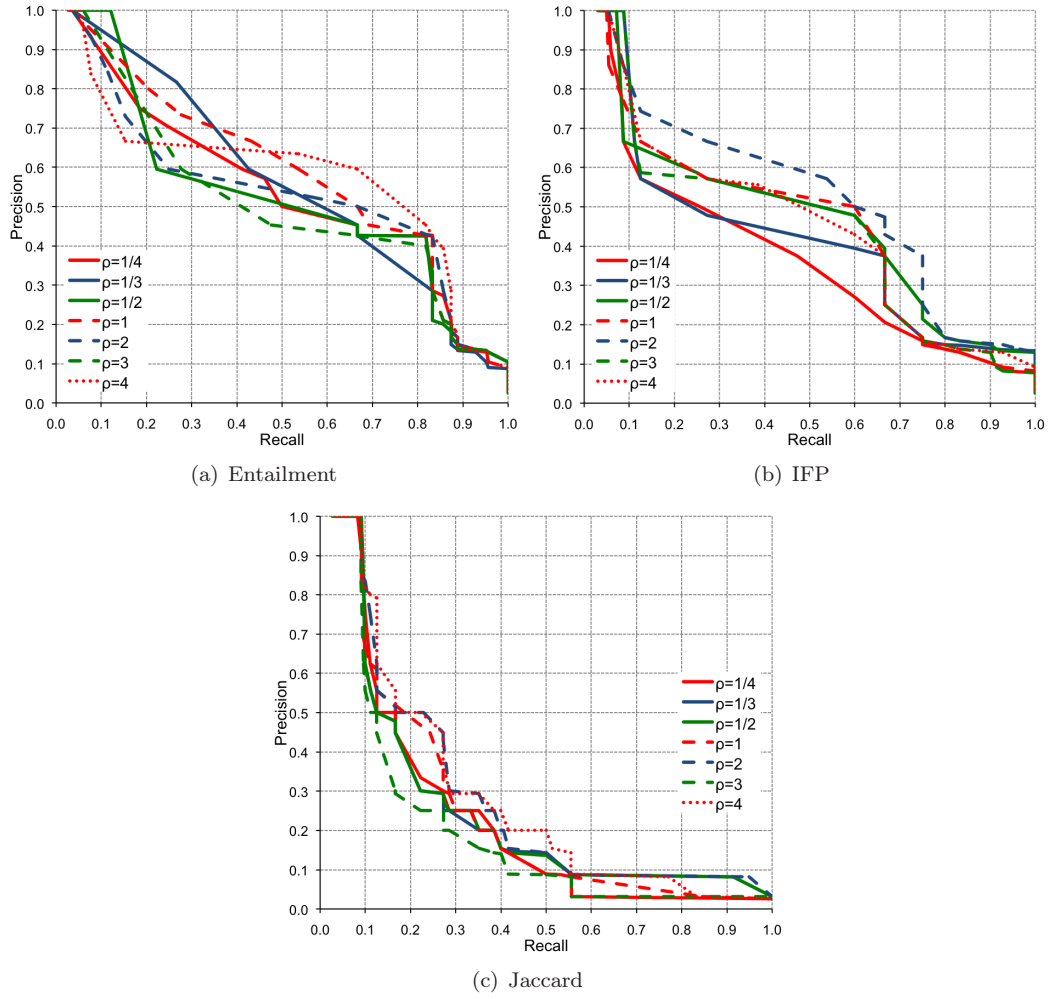


Figure K.4: Precision/Recall curves when tuning ρ for FNE Self-trained Perceptron with different feature similarity measures

with Entailment yields an optimum value for precision at recall 90% where $\rho = 2$ when the classifier is combined with Entailment as a feature similarity measure. For the combination of Naive Bayes with IFP $\rho = 3$ optimises precision at recall 90%, and for Naive Bayes with Jaccard the optimum setting is $\rho = \frac{1}{3}$.

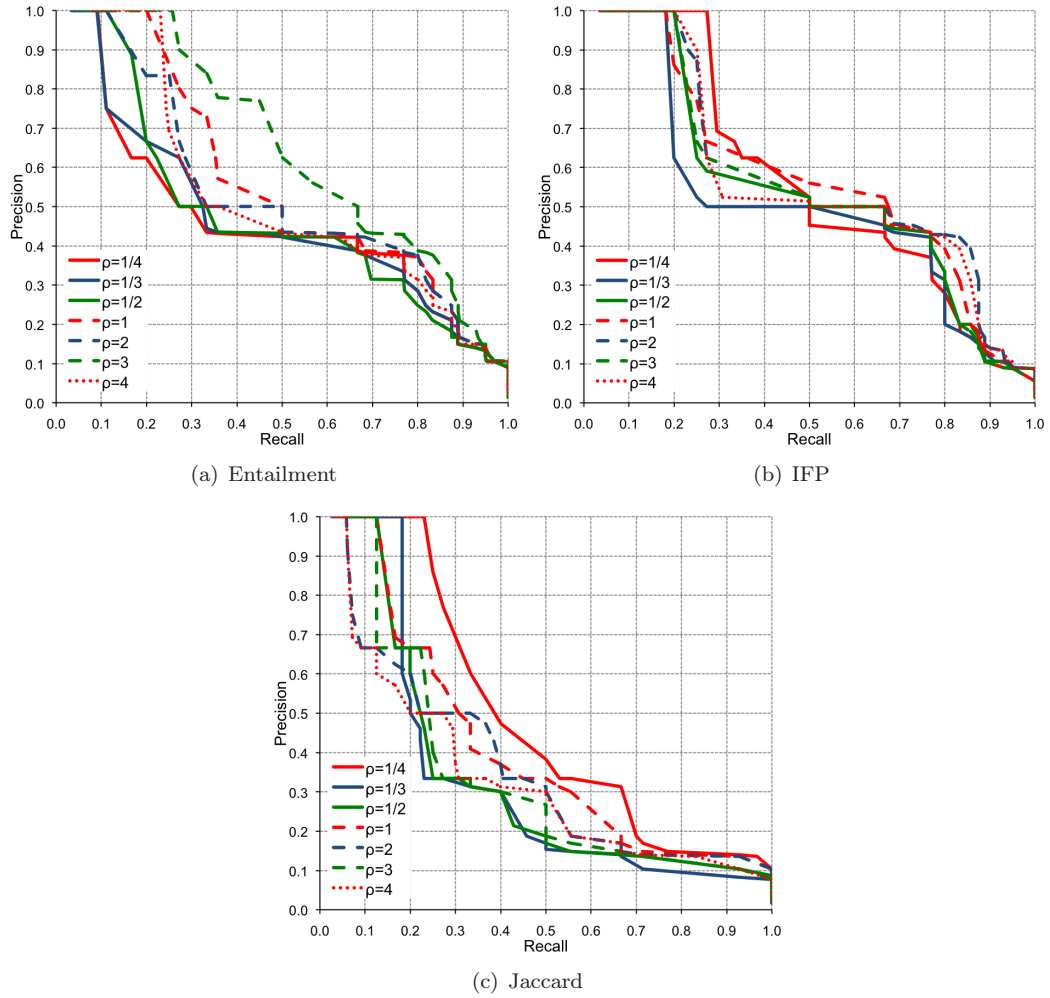


Figure K.5: Precision/Recall curves when tuning ρ for FNE Self-trained SVM with different feature similarity measures

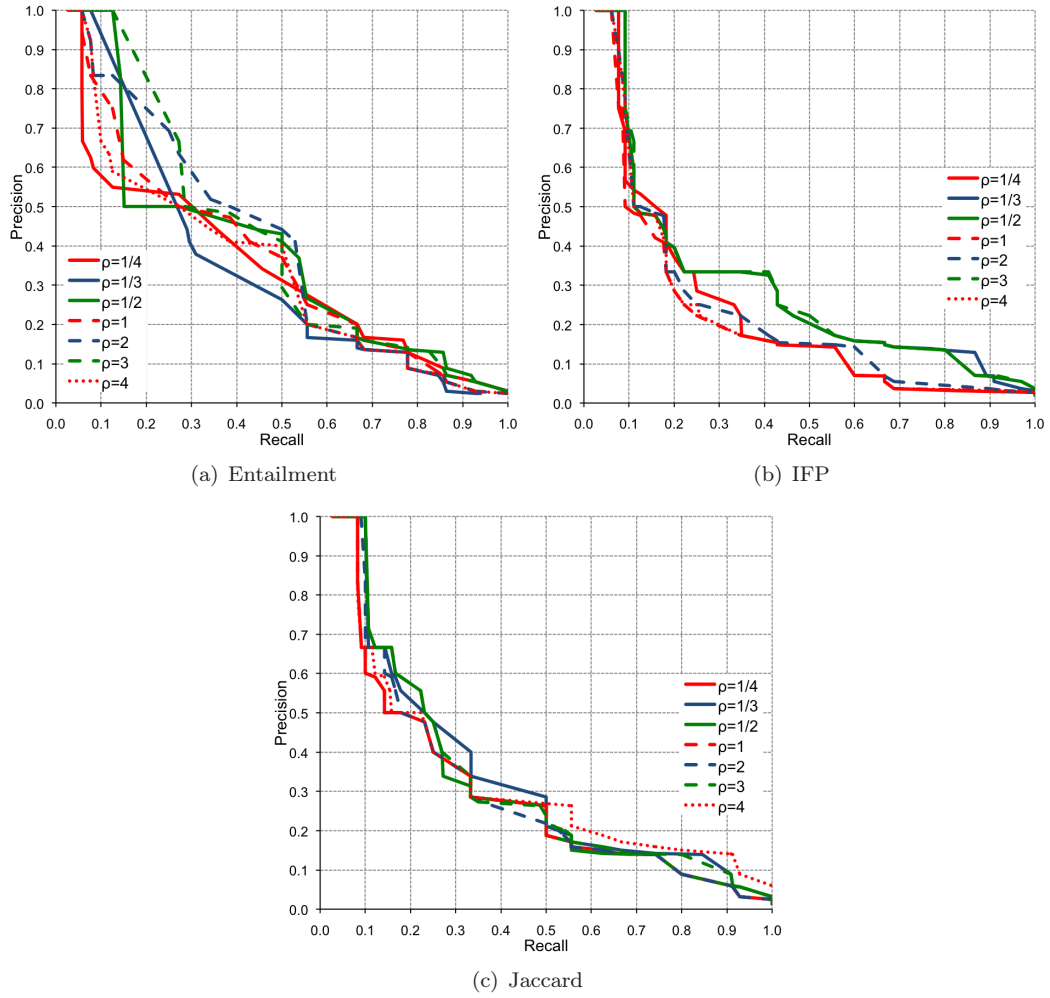


Figure K.6: Precision/Recall curves when tuning ρ for FNE Self-trained Naive Bayes with different feature similarity measures

Appendix L

Tuning Yarowsky Algorithm

The implementation of the Yarowsky algorithm allows the explored disambiguation techniques to be compared against an existing bootstrapping approach using a shallow feature representation. Chapter 10 presents an overview of the baseline technique, which takes as input the same seed data as the disambiguation techniques presented in chapters 7, 8 and 9, and compiles a bag-of-words representation from the provided data. To contrast the performance of the shallow approach, the same classification algorithms are used as those implemented in the Self-training approach in chapter 9: Perceptron, SVM and Naive Bayes.

In a similar manner to the classifiers in chapter 9, the task remains to generate negative training data for a binary classification scenario. To do this the Rocchio classification technique is used from chapter 9, where the value of ρ must be chosen to generate negative samples. Therefore this parameter was tuned for the implementation of the Yarowsky algorithm using the 10% training split of the dataset. Each classifier was then trained using a unigram bag-of-words model, and the results plotted using precision/recall curves. Figure L.1(a) shows the precision/recall curve where $\rho = 1/4$ is shown to achieve superior performance to other values of ρ . Figure L.1(b) shows that there is little difference between the performance levels of different values of ρ when generating the negative training data for SVM, $\rho = 1/4$ is chosen as the best. Figure L.1(c) also show little variation between the performance levels for differing values of ρ when applied to Naive Bayes, $\rho = 1/4$ is chosen.

Related work within the field of text classification often uses the default setting of $\rho = 1/4$ when applying Rocchio classification to generate negative training examples. The results from tuning indicates that the same value settings when applied to a bag-of-words representation and bootstrapping the classifiers using the Yarowsky algorithm achieves optimum results over other settings of ρ . Figure ?? compares the three classifiers coupled with their best settings for ρ . Naive Bayes achieves 0.28 for recall at precision 90%, compared to 0.14 returned by SVM at the same level of precision. For precision at recall 0.9 Naive Bayes yields 0.12, SVM produces 0.363 and Perceptron produces 0.366.

Following tuning of settings for ρ for all three classifiers and the derivation of the optimum settings based on maximum precision achieved at recall 90%, each classifier and selected value of ρ was then run over the remaining 90% test split of the dataset. The results of which are shown within chapter 10, when compared against the Self-trained models which use metadata models for representing information found within a given web resource.

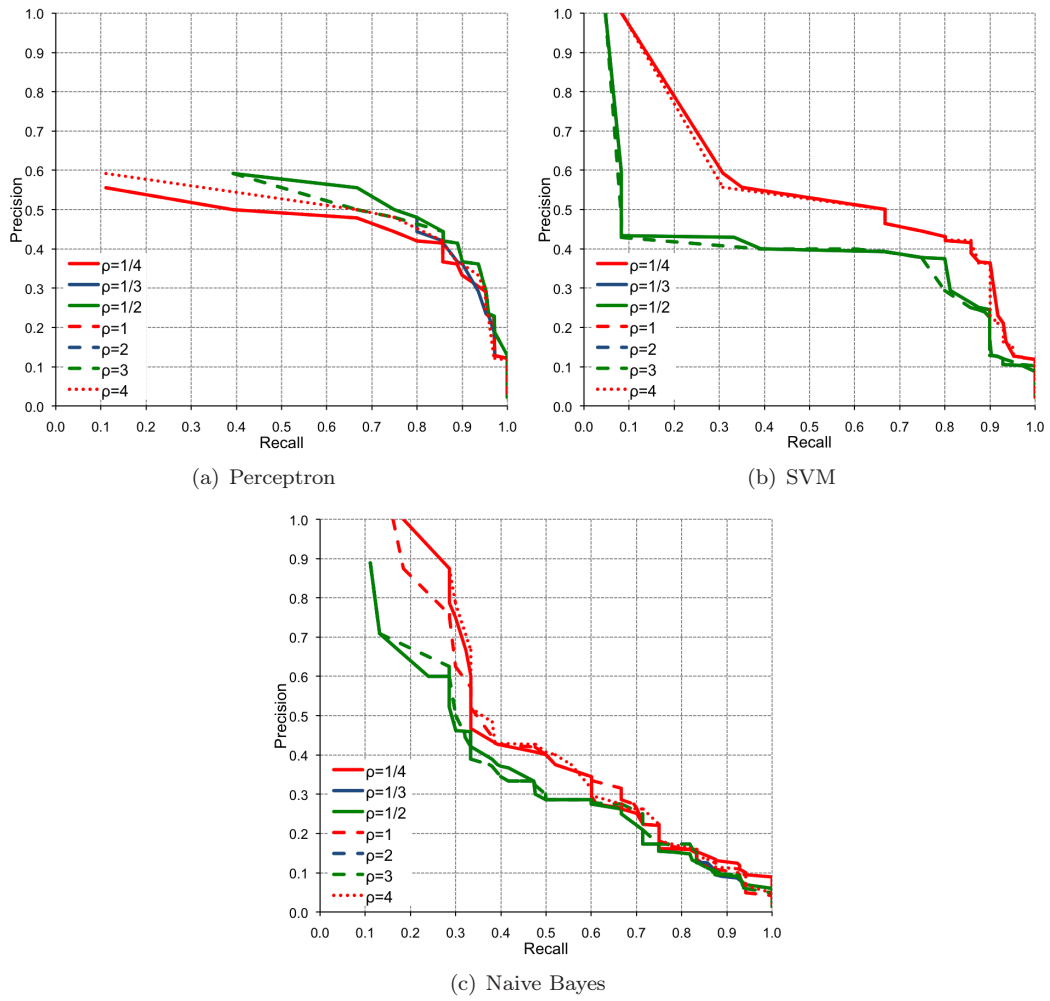


Figure L.1: Precision/Recall curves for classifiers trained using the Yarowsky algorithm with different levels of ρ

Appendix M

Self-training: Random Sampling for Negative Set Generation

The importance of constructing a good negative set is vital when using a semi-supervised learning strategy such as Self-training discussed within chapter 9. Early experiments explored the use of various strategies before settling on the use of Rocchio classification explained in §9.4.1. This appendix presents the results from using a strategy similar to work from [YHc04] by compiling a list of positive features from instances in the positive set. All instances from the universal set are then filtered out that contain any positive features from the list. This leaves a weak approximation of negative instances, given that it is possible for false negatives to be returned containing positive features which are yet to be observed. In order to reduce the effects of this possible bias the negative set is built to be the equivalent size of the positive set, both in the number of instances and the number of features within those instances. The intuition being that the use of Self-training will alter the hypothesis over time thereby correcting the effects of false negatives.

Table M.1: Accuracy levels using a random negative set generation strategy

		Precision	Recall	F_1	F_2	F_3
Perceptron	Entailment	0.645	0.636	0.540	0.555	0.578
	IFP	0.865	0.364	0.512	0.383	0.371
	Jaccard	0.935	0.318	0.474	0.341	0.327
SVM	Entailment	0.733	0.670	0.642	0.648	0.657
	IFP	0.811	0.412	0.457	0.424	0.418
	Jaccard	0.912	0.384	0.540	0.410	0.395
Naive Bayes	Entailment	0.684	0.665	0.606	0.618	0.634
	IFP	0.786	0.418	0.545	0.430	0.422
	Jaccard	0.906	0.389	0.545	0.416	0.401
	Baseline 1	0.213	0.999	0.330	0.519	0.663
	Baseline 2	0.239	0.748	0.288	0.404	0.492
	Baseline 3	0.765	0.725	0.719	0.703	0.697
	Baseline 4	0.226	0.956	0.324	0.487	0.614

As the results from Table M.1 indicate the accuracy levels, F_1 , F_2 and F_3 scores are low with respect to the accuracy levels achieved when using Rocchio classification. Despite this strategy performing poorly in terms of recall levels, it does yield high levels of precision

for the majority of permutations. However in the context of this thesis, recall is considered as being more important than precision: given that the goal is to protect against a given web user from having their identity stolen, or a piece of sensitive information being disseminated, therefore the larger values of β for F-measure indicates the unsuitability of a random sampling strategy in this context.