A Study on Attributional and Relational Similarity between Word Pairs on the Web

Danushka Tarupathi Bollegala

Doctor of Philosophy (Information Science & Technology)
Graduate School of Information Science and Technology,
The University of Tokyo

June 2009

THE UNIVERSITY OF TOKYO

“Life is like riding a bicycle. To keep your balance you must keep moving.”

Albert Einstein

Dedication

To Manisha, and to my dearest parents...

Acknowledgments

This thesis would not have been possible without the support of many people. I would like to extend my thanks to all who supported, commented and encouraged me during this long process.

First and foremost, I would like to express my sincere thanks to my adviser Prof. Mitsuru Ishizuka, Department of Creative Informatics, The University of Tokyo. I would like to take this opportunity to convey my gratitude for his guidance, support, and encouragement.

I would like to express my warm thanks to Prof. Yutaka Matsuo, Department of Technology Management for Innovation, The University of Tokyo. Discussions at our weekly research meetings (*Matsuo-gumi*) were very helpful in shaping my research. My special thanks goes to Dr. Naoaki Okazaki, The University of Tokyo, for his insightful comments at various stages of my research. I would like to thank all my colleagues at Ishizuka Lab for their kind support and encouragement: Dr. Junichiro Mori, Dr. YingZi Jin, Keigo Watanabe, and Taiki Honma. The work mentioned in this thesis was submitted to various journals, conferences, and workshops. I received invaluable comments during the process, which greatly helped to improve the quality of the research. I would like to thank all the anonymous reviewers for their comments.

Finally, my thanks goes to my family and friends. My parents have always encouraged me and have provided a wonderful education throughout my life. Intellectual discussions with my mother played and important role in widening my academic interests. Spending my time with my sister and her family has been of great help to relax my mind. My love goes out to my wife Manisha, who has supported me in every aspect of my life, and scarified her time that she would love to spend with me to let me compile this thesis.

Publications

Journal Papers

1. Danushka Bollegala, Naoaki Okazaki and Mitsuru Ishizuka. A Bottom-up Approach to Sentence Ordering for Multi-document Summarization. *Information Processing and Management (IPM), Elsevier, To Appear.*

2. Danushka Bollegala, Yutaka Matsuo and Mitsuru Ishizuka. Measuring Semantic Similarity between Words using Web Search Engines. *Journal of Web Semantics (JWS), Elsevier, submitted.*

3. Danushka Bollegala, Yutaka Matsuo and Mitsuru Ishizuka. Automatic Annotation of Ambiguous Personal Names on the Web. *Computational Intelligence, Wiley, To Appear.*

4. Danushka Bollegala, Yutaka Matsuo and Mitsuru Ishizuka. Automatic Discovery of Personal Name Aliases from the Web. *IEEE Transactions on Knowledge and Data Engineering, IEEE, submitted.*

International Conference Papers

1. Danushka Bollegala, Yutaka Matsuo and Mitsuru Ishizuka. A Relational Model of Semantic Similarity between Words using Automatically Extracted Lexical Pattern Clusters from the Web. In Proceedings of *Empirical Methods in Natural Language Processing (EMNLP'09)*, Singapore, 2009. To Appear.

2. Danushka Bollegala, Yutaka Matsuo and Mitsuru Ishizuka. Measuring the Similarity between Implicit Semantic Relations from the Web. In Proceedings of *the 18th*

International World Wide Web Conference (WWW 09), pp. 651-660, Madrid, Spain, 2009.

3. Keigo Watanabe, Danushka Bollegala, Yutaka Matsuo and Mitsuru Ishizuka. A Two-Step Approach to Extracting Attributes for People on the Web. In Proceedings of *the second Web People Search Task (WePS-2) Workshop at the 18th International World Wide Web Conference (WWW 09)*, (6 pages), Madrid, Spain, 2009.

4. Danushka Bollegala, Yutaka Matsuo and Mitsuru Ishizuka. Measuring the Similarity between Implicit Semantic Relations using Web Search Engines. In Proceedings of *the 2nd ACM International Conference on Web Search and Data Mining Conference (WSDM 09)*, pp. 104-113, Barcelona, Spain, 2009.

5. Yutaka Matsuo, Danushka Bollegala, Hironori Tomobe, YingZi Jin, Junichiro Mori, Keigo Watanabe, Taiki Honma, Masahiro Hamasaki, Kotaro Nakayama and Mizuki Oka. Social Network Mining from the Web (poster). *NSF Sponsored Symposium on Semantic Knowledge Discovery, Organization and Use,* New York, 2008.

6. Danushka Bollegala, Taiki Honma, Yutaka Matsuo and Mitsuru Ishizuka. Automatically Extracting Personal Name Aliases from the Web. In Proceedings of the 6th International Conference on Natural Language Processing (GoTAL 08), Advances in Natural Language Processing Springer LNCS 5221, pp 77-88, Gothernburg, Sweeden, 2008.

7. Danushka Bollegala, Yutaka Matsuo and Mitsuru Ishizuka. WWW sits the SAT: Measuring Relational Similarity from the Web. In Proceedings of *the 18th European Conference on Artificial Intelligence (ECAI 08)*, pp. 333-337, Patras, Greece, 2008.

8. Danushka Bollegala, Taiki Honma, Yutaka Matsuo and Mitsuru Ishizuka. Mining for Personal Name Aliases on the Web, (poster paper). In Proceedings of the 17th International World Wide Web Conference (WWW 08), Vol. II, pp. 1107-1108, Beijing, China, 2008.

9. Danushka Bollegala, Taiki Honma, Yutaka Matsuo and Mitsuru Ishizuka. Identification of Personal Name Aliases on the Web. In Online Proceedings of *WWW 2008*

Workshop on Social Web Search and Mining (SWSM 08), 8 pages, Beijing, China, 2008.

10. Danushka Bollegala, Yutaka Matsuo and Mitsuru Ishizuka. A Co-occurrence Graph-based Approach for Personal Name Alias Extraction from Anchor Texts, (poster paper). In Proceedings of *the 3rd International Joint Conferences on Natural Language Processing (IJCNLP 08)*, Vol. II. pp. 865-870, Hyderabad, India, 2008.

11. Danushka Bollegala, Yutaka Matsuo and Mitsuru Ishizuka. Measuring Semantic Similarity between Words Using Web Search Engines. In Proceedings of the *16th International World Wide Web Conference (WWW 07)*, pp. 757-766, Banff, Canada, 2007.

12. Danushka Bollegala, Yutaka Matsuo and Mitsuru Ishizuka. An Integrated Approach to Measuring Semantic Similarity between Words using Information Available on the Web. In Proceedings of *the main conference of Human Language Technologies and North American Chapter of the Association for Computational Linguistics (HLT-NAACL 07)*, pp. 340-347, Rochester, New York, 2007.

13. Danushka Bollegala, Yutaka Matsuo, Mitsuru Ishizuka. Identifying People on the Web through Automatically Extracted Key Phrases. In Proceedings of *TextLink Workshop at International Joint Conference on Artificial Intelligence (IJCAI)*, Hyderabad, India, 2007.

14. Danushka Bollegala, Yutaka Matsuo, Mitsuru Ishizuka. Disambiguating Personal Names on the Web using Automatically Extracted Key Phrases. In Proceedings of *European Conf. on Artificial Intelligence (ECAI)*, pp.553-557, Riva, Italy, 2006.

15. Danushka Bollegala, Yutaka Matsuo, Mitsuru Ishizuka. Extracting Key Phrases to Disambiguate Personal Name Queries in Web Search. In Proceedings of the *Workshop "How can Computational Linguistics improve Information Retreival?", at the joint 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL 2006)*, pp.17-24 , Sydney, Australia, 2006.

16. Danushka Bollegala, Naoaki Okazaki, Mitsuru Ishizuka. A bottom-up approach to Sentence Ordering for Multi-document Summarization. In Proceedings of the *Joint 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL 2006)* ,pp. 385-392, Sydney Australia. 2006.

17. Yutaka Matsuo, Masahiro Hamasaki, Hideaki Takeda, Junichiro Mori, Danushka Bollegala, Hiroyuki Nakamura, Takuichi Nishimura, Koiti Hashida and Mitsuru Ishizuka. Spinning Multiple Social Networks for Semantic Web. In Proceedings of the *21st National Conference on Artificial Intelligence (AAAI 2006)*, pp 1381-1387, Boston, MA, USA,2006.

18. Danushka Bollegala, Yutaka Matsuo and Mitsuru Ishizuka. Extracting Key Phrases to Disambiguate Personal Names on the Web. In Proceedings of the *7th International Conference on Intelligent Text Processing and Computational Linguistics (CICLing 2006)*, pp. 223-234, Mexico City, Mexico, 2006.

19. Danushka Bollegala, Naoaki Okazaki and Mitsuru Ishizuka. A Machine Learning Approach to Sentence Ordering for Multi-document Summarization and its Evaluation. In Proceedings of the *2nd International Joint Conference on Natural Language Processing (IJCNLP 2005)*, pp. 624-635, Jeju, South Korea, 2005.

Domestic Conference Papers

1. Danushka Bollegala, Yutaka Matsuo and Mitsuru Ishizuka. A Relational Model of Semantic Similarity. In Proceedings of *the 23rd Annual Conference of the Japanese Society for Artificial Intelligence*, 2009.

2. Danushka Bollegala, Yutaka Matsuo and Mitsuru Ishizuka. WWW sits the SAT: Measuring Relational Similarity on the Web. In Proceedings of *the 22nd Annual Conference of the Japanese Society for Artificial Intelligence*, 2008.

3. Danushka Bollegala, Yutaka Matsuo and Mitsuru Ishizuka. WebSim: A Web-based Semantic Similarity Measure. In Proceedings of *the 21st Annual Conference of the Japanese Society for Artificial Intelligence*, 2007.

4. Danushka Bollegala, Naoaki Okazaki and Mitsuru Ishizuka. Agglomerative Clustering Based Approach to Sentence Ordering for Multi-document Summarization. In Proceedings of *Natural Language Understanding and Models of Communication (NLC) The Institute of Electronics, Information and Communication Engineers (IEICE)*, pp 13-18, Biwako, Japan, 2006.

5. Danushka Bollegala, Yutaka Matsuo and Mitsuru Ishizuka. Disambiguating Personal Names on Web. In Proceedings of *Annual Meeting of the Japanese Society of Artificial Intelligence (JSAI)*, Tokyo, Japan, 2006.

6. Danushka Bollegala, Naoaki Okazaki and Mitsuru Ishizuka. Agglomerative Clustering Based Approach to Sentence Ordering for Multi-document Summarization. In Proceedings of *Annual Meeting of the Japanese Society of Artificial Intelligence (JSAI)*, Kokura, Japan, 2005.

7. Danushka Bollegala, Naoaki Okazaki and Mitsuru Ishizuka. A Machine Learning Approach to Sentence Ordering for Multi-document Summarization. In Proceedings of *Annual Meeting of the Natural Language Processing Society of Japan (NLP)*, pp. 636-639, Takamatsu, Japan, 2005.

8. Danushka Bollegala, Naoaki Okazaki and Mitsuru Ishizuka. A Machine Learning Approach to Sentence Ordering for Multi-document Summarization. In Proceedings of the *Annual Meeting of the Information Processing Society of Japan (IPSJ)*, Tokyo, Japan, 2005.

Abstract

Similarity is a fundamental concept that extends across numerous fields such as artificial intelligence, natural language processing, cognitive science and psychology. Similarity provides the basis for learning, generalization and recognition. Similarity can be broadly divided into two types: semantic (or attributional) similarity, and relational similarity. Attributional similarity is the correspondence between the attributes of two objects. If two objects have identical or close attributes, then those two objects are considered attributionally similar. For example, the two concepts, *car* and *automobile*, both have an identical set of attributes: four wheels, doors, and both are used for transportation. Consequently, the two words *car* and *automobile* show a high degree of semantic (attributional) similarity and are considered synonymous. On the other hand, relational similarity is the correspondence between the implicit semantic relations that exist between two pairs of words. For example, consider the two word-pairs *(ostrich, bird)* and *(lion, cat)*. Ostrich is a large bird and lion is a large cat. The implicitly stated semantic relation *is a large* holds between the two words in each word-pair. Therefore, those two word-pairs are considered relationally similar. Typically, word analogies show a high degree of relational similarity.

This thesis addresses the problem of measuring both semantic (attributional) and relational similarity between words or pairs of words from the web. I define the two types of similarity in detail and analyze the concept of similarity from numerous view-points, its philosophical, linguistic, and mathematical interpretations. In particular, I compare different models such as the contrast model, transformational model and relational model, of similarity. I presents a supervised approach to measure the semantic similarity between two words using a web search engine. To measure the attributional similarity between two given words, first, I search for those words individually and also in a conjunctive combination in a

web search engine. I then extract lexical patterns that describe numerous semantic relations between the two words under consideration. Moreover, I compute popular co-occurrence measures such as the Jaccard coefficient, Overlap coefficient, Dice coefficient, and pointwise mutual information, using the page counts retreived from a search engine. All those measures are integrated with lexical patterns through a machine learning framework. The training data for the algorithm are selected from synsets in WordNet. The proposed method reports a high correlation with human ratings in a benchmark dataset for semantic similarity. Moreover, The proposed semantic similarity is used in a community clustering task and a word sense disambiguation task. Both those tasks show the ability of the proposed semantic similarity measure to accurately compute the similarity between named-entities. This is particularly useful because semantic similarity measures that require manually created lexical resources such as dictionaries are unable to compute the similarity between named-entities, which are not well covered by dictionaries.

Chapter 3 studies the problem of relational similarity. Given two word-pairs *(A,B)* and *(C,D)*, I propose a relational similarity measure, $\mathrm{relsim}((A,B),(C,D))$ to compute the similarity between the implicit semantic relations that hold between the two words A and B, and C and D. To represent the implicitly stated semantic relations between two words, I extract lexical patterns from the snippets retrieved from a web search engine for the two words. However, not all lexical patterns describe a different semantic relation. Some relations can be represented by more than one lexical pattern. For example, both patterns ***X** is a **Y***, and ***Y** such as **X*** describe a hypernymic relation between **X** and **Y**. Then the extracted patterns are clustered using distributional similarity to identify the different patterns that describe a particular semantic relation. Finally, machine learning approaches are used to compute the relational similarity between two given word-pairs using the lexical patterns extracted for each word-pair. I experiment with support vector machines, and information theoretic metric learning approach to learn a relational similarity measure.

The second half of this thesis describes the applications of semantic and relational similarity. As a working problem, I concentrate on personal name disambiguation on the web. A name of a person can be ambiguous on the web because of two main reasons. First, different people can share the same name (namesake disambiguation problem). Second, a single individual can have multiple aliases on the web (alias detection problem). Chapter 4 analyzes the namesake disambiguation problem, whereas, Chapter 5 focuses on the alias detection problem. I propose fully automatic methods to solve both these problems with a high accuracy. Extracting attributes for a particular person such as date of birth, nationality, affiliation, job title, etc. is particularly helpful to disambiguate that person from his or her namesakes on the web. I present some preliminary work that I have conducted in this area in Chapter 6. In Chapter 7, I present a relational model of semantic similarity that links relational and semantic similarity measures that were introduced in the thesis. In contract to the feature model of semantic similarity, which models objects using their attributes, the relational model attempts to compute the semantic similarity between two given words directly using the numerous semantic relations that hold between the two words. In Chapter 8, I conclude this thesis with a description of potential future work in web-based similarity measurement.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 The Concept of Similarity

Similarity is a fundamental concept that has been studied across various fields such as cognitive science, philosophy, psychology, artificial intelligence, and natural language processing. The quest for similarity dates back to Plato. The famous Plato's problem asks how do people acquire as much knowledge as they do on the basis of as little information as they get. Landauer and Dumais [78] propose Latent Semantic Analysis, a general theory of acquired similarity and knowledge representation based on co-occurrences of words, as a solution to the Plato's problem.

Similarity plays a fundamental role in categorization. Individuals categorize newly encountered objects to existing categories by comparing them using a notion of similarity. Then a newly encountered object is assigned to the category to which it is most similar. Once the newly encountered object is properly categorized, we can infer additional properties about it using the properties of its category. In fact, this is a process adopted by biologists to categorize newly found species to existing animal or plant classes. For example, if we already know that crocodiles are very sensitive to the cold and we found out that crocodiles and alligators are similar, then we can infer that alligators are also sensitive to cold. Tanenbaum [143] showed that as the similarity between two concepts A and B increases, so does the probability of correctly inferring that B has the property X upon knowing that A has X. This view of similarity as a key concept of categorization was introduced in 1970s. However, in 1980s a new view emerged that held that similarity was too weak and vague to ground human categorization. Under this new view, similarity judgments are governed by theories that determine the relevant properties of evaluation. If a theory describing a category could successfully explain the newly encountered object, then it would be assigned to that category. However, this theory-based view of similarity has lead to dissatisfaction among researchers, and richer and more well-developed similarity-based approaches are being proposed.

Similarity has been studied in theories of learning, a process in which features are generalized over a set of given training examples to recognize underline patterns. For example, in Tversky's feature model of similarity [153], an object is represented by a set of features. Similarity between two objects is then computed using the common and distinct features between the two objects. Consequently, if we know a set of objects that belong to the same category (i.e. positive and negative training instances in a binary classification setting), then most machine learning algorithms attempt to identify the features that are common to instances in the category.

Measuring the similarity between words is a fundamental step in numerous tasks in natural language processing such as word-sense disambiguation [127], language modeling [129], synonym extraction [84], and automatic thesauri extraction [34, 97]. In word-sense disambiguation the goal is to determine which sense of a polysemous word (i.e. a word that has multiple senses) is used in a given text. Similarity measures have been used to compare the words that appear in the immediate context of a ploysemous word against the words that are used in definitions given in a dictionary (i.e. glosses) for each of the senses. Then the sense that has the highest similarity with the given context is selected as the correct sense of the polysemous word.

In language modeling the objective is to create a probabilistic model of language using word-sequence occurrences. If a probabilistic model can accurately predict words in a language, then that model can be considered as accurate. In other words, a language model with a low perplexity value is desirable. Perplexity of a probability distribution p is defined as $2^{H(p)}$, where $H(p)$ denotes the entropy of p. A major problem faced during the computation of word-sequence probabilities is the data sparseness. Certain sequences of words are rare even in large text corpora and accurately estimating their probabilities is difficult.

Synonyms have been used to overcome the sparseness problem [129]. In this approach, an occurrence of a synonym of a word w is also counted as an occurrence of w.

Lin [84] proposed the use of distributional similarity to identify similar words. Sentences are first parsed using a dependency parser and word pairs with a dependency relation are extracted. If two words X and Y frequently have a dependency relation with a word Z, then X and Y are considered to be similar. Similarity measures have found useful applications in automatic thesaurus extraction. Matsuo et al. [97] proposed the use of pointwise mutual information (PMI) computed using web page counts to measure the association between two words in the web. Thesauri of synonymous or related words are used in information retrieval to expand the user queries or to suggest related queries to the user, thereby improving the recall in search.

1.2 Attributional vs. Relational Similarity

Before we further extend our study of similarity it is important to define two types of similarities: attributional similarity and relational similarity. Given two objects A and B, the attributional similarity between A and B is the correspondence between the *attributes* of A and B. For example, consider the two words *car* and *automobile*. Both cars and automobiles have similar attributes: four wheels, doors, and used for transportation. Consequently, we observe a high degree of attributional similarity between cars and automobiles. Synonyms are typical examples of high degree of attributional similarity. Previous literature on similarity have also used the term *semantic similarity* to refer to attributional similarity. In this thesis, I use semantic similarity as a synonym for attributional similarity.

On the other hand, relational similarity is the correspondence between the *relations* that hold between the two words in word pairs. Given two word pairs, (A, B) and (C, D), if the semantic relations that hold between A and B in the first word pair are similar to the semantic relations that hold between C and D in the second word pair, we define the two word pairs to be relationally similar. For example, consider the two word pairs *(ostrich, bird)* and *(lion, cat)*. Ostrich is the largest bird on the planet and lion is the largest cat. The relation *is the largest* holds between the two words in each word pair. Consequently, those two word pairs are considered to be relationally similar. Analogies are typical examples of high relational similarity. The *(ostrich, bird)* vs. *(lion, cat)* example is particularly interesting because it shows that even through there is almost no attributional similarity between *lions* and *ostriches*, it is still possible to have high relational similarity between the two word pairs.

All applications of similarity introduced in section 1.1 are examples of attributional similarity. In comparison to attributional similarity, which has been studied extensively for its numerous applications [127, 69, 87, 22, 26, 17], the problem of measuring relational similarity has received less attention. In order to measure relational similarity between two pairs of words, one must first identify the relations that hold between the two words in each pair and then compare the relations between the pairs. Turney et al. [152] proposed the use of Scholastic Aptitude Test (SAT) word-analogy questions as a benchmark to evaluate relational similarity measures. An SAT analogy question comprises a source-pairing of concepts/terms and a choice of (usually five) possible target pairings, only one of which accurately reflects the source relationship. A typical example is shown below.

Question: Ostrich is to Bird as:

a. Cub is to Bear

b. *Lion is to Cat*

c. Ewe is to Sheep

d. Turkey is to Chicken

e. Jeep is to Truck

Here, the relation *is a large* holds between the two words in the question (e.g. Ostrich and Bird), which is also shared between the two words in the correct answer (e.g. Lion *is a large* Cat). Solving word analogy question has been difficult even for humans. The average SAT score reported for the college level students on the SAT word analogy questions is 57%.

Noun-modifier pairs such as *flu virus*, *storm cloud*, *expensive book*, etc are frequent in English language. In fact, WordNet contains more than $26,000$ noun-modifier pairs. According to the relations between the noun and the modifier, Natase and Szpakowicz [108] classified noun-modifiers into five classes: causal (groups relations enabling or opposing an occurrence, e.g. *flu virus*), participant (groups relations between an occurrence and its participants or circumstances, e.g. *student protest*), spatial (groups relations that place an occurrence at an absolute or relative point in space, e.g. *outgoing mail*), temporal (groups relations that place an occurrence at an absolute or relative point in time, e.g. *weekly game*), and quality (groups the remaining relations between a verb or noun and its arguments, e.g. *stylish writing*). Turney [150] used a relational similarity measure to compute the similarity between noun-modifier pairs and classify them according to the semantic relations that hold between a noun and its modifier. Using a relational similarity measure, he proposes a k-nearest neighbor classifier to find the relation between the noun and the modifier.

In word sense disambiguation (WSD) [7], identifying the various relations that hold between an ambiguous word and its context is vital. For example, the word "plant" can refer to an industrial plant or a living organism. If the word "food" appears in the immediate context of "plant", then a typical WSD approach is to compare the attributional similarity between "food" and "industrial plant" to that of "food" and "living organism" and to select the sense with higher attributional similarity. Considering the fact that industrial plants often produce food and living orgasms often serve as food, the decision may not be very clear. However, if we can identify the relation between "food" and "plant" as "food *for* the plant" then it strongly suggests that the plant is a living organism. On the other hand, a relation such as "food *at* the plant" suggests the plant to be an industrial plant.

An interesting application of relational similarity in information retrieval is to search using implicitly stated analogies [94, 156]. For example, the query "Muslim Church" is expected to return "mosque", and the query "Hindu bible" is expected to return "the Vedas". These queries can be formalized as word pairs: (Christian, Church) vs. (Muslim,X), and (Christian, Bible) vs. (Hindu,Y). We can then find the words X and Y that maximize the relational similarity in each case. Searching using implicitly stated semantic relations between word pairs is an interesting novel paradigm of information retrieval. Almost all existing commercial search engines are "keyword-based". These search engines usually return a set of relevant documents for a user query. Although similarity does not always mean relevance (or vice versa), we can think of existing search engines as measuring attributional similarity between a user query and a document. In contrast, in analogical search, the underlying mechanism is essentially relational similarity.

1.3 Similarity in Structure

Measuring the similarity between mathematical structures such as graphs, is important in identifying equivalences. Two networks can be considered *structurally equivalent* if there is a resemblance in the way the nodes connect to each other in the two networks. For example, in a star network only a single node is connected to rest of the nodes in the network. If we ignore the number of nodes, the weights and directions of the edges, then we can consider two star networks to be structurally equivalent. The central node (i.e. the node that is connected to all the other nodes in the network) plays an important role in a star network because without it the network will be completely disconnected. On the other hand, in a clique, each node is connected to all the other nodes (i.e. vertices in a triangle). There is no clear *central* player in a clique. Between the two extremes of star networks and cliques, various other network structures are possible. The ability to measure the structural equivalence of mathematical structures is particularly helpful to understand the topology of a given structure. For example, if we are told that a particular organization has a star structure, then we would expect to find a single important person in the organization. All shortest paths between rest of the nodes go through this central person. Consequently, the betweenness value for this central person is one. When more and more edges form between other nodes in the network (i.e. more relations materialize between other people in the organization), the betweenness value of the central node decreases. Recent growth of online large-scale social network systems (SNSs) such as Facebook [1], has increased the attention on network similarity measures. For example, if two people A and B has the same set of friends, then it is likely that A and B are also friends. If A is not already a friend of B, then the SNS can suggest B to A.

Lorrain and White [89] define two actors (i.e. nodes) to be structurally equivalent, if they have identical ties (i.e. edges) to and from all the other actors in a digraph or a non-directed graph. Given the adjacency matrix of a network, structural equivalence of two nodes can be easily detected by comparing corresponding row and column vectors of the nodes. Vector comparison can be performed using conventional similarity measures such as cosine measure or Manhattan distance (in the case of binary valued edges). This

[1] http://www.facebook.com/

definition of structural equivalence only consider the first order connections (i.e. nodes that are directly connected). However, it is possible to extend this definition easily to second or higher order connections by considering the powers of the adjacency matrix. For example, a non-zero value in the square of the adjacency matrix indicates a second order connection between the corresponding two nodes. It it noteworthy that if the network has islands (i.e. disconnected components), then nodes in different islands are unreachable. Consequently, certain elements in the n-th power of the adjacency matrix can still be zero for a network with n nodes.

1.4 Similarity vs. Distance

Distance is considered as the inverse of similarity in many applications. However, strict definition of distance as a metric must satisfy the following conditions. A function d to become a distance metric, it must satisfy all of the following constraints for any three different objects A, B, and C.

self-similarity: $d(A, A) = d(B, B)$

minimality: $d(A, B) \geq d(A, A)$

symmetry: $d(A, B) = d(B, A)$

triangularity: $d(A, B) + d(B, C) \geq d(A, C)$

Equivalent transformations of those constrains have also been proposed. For example, the minimality constraint and self-similarity constrained together suggest that there must be some constant minimum value d_{min} to d. We can subtract this value from d and redefine d such that both minimality and self-similarity constraints can be replaced with a *non-negativity* constraint: $d(A, B) \geq 0$, where equality holds when $A = B$. When two objects are identical the distance between them would be zero and similarity will take its maximum value. When two objects differ from each other the distance between them increases and similarity reduces. Typically distance metrics are defined to take values in range $[0, +\infty)$, whereas similarity functions are limited to a range $[0, 1]$. Given a distance metric $d(A, B)$, we can define its corresponding similarity function using a monotonously decreasing function that squashes the $[0, +\infty)$ range to $[0, 1]$. For example, $\exp(-d(A, B))$ is such a function. However, it is noteworthy that similarity functions are not necessarily required to satisfy the above mentioned metric constraints. Therefore, given a similarity function we cannot always find a corresponding distance function that satisfies the above mentioned metric conditions.

The question whether similarity and distance are inversely related, has been debated in great deal in cognitive science. Psychological experiments have been conducted where human subjects assign similarity scores and distance scores to some pairs of objects. If there is a negative correlation between similarity judgments and distance judgments, then we can conclude that there is an inverse relationship between similarity and distance. In these experiments, similarity scores are plotted against distance scores and linear regression analysis is used to find the relationship between the two sets of scores. Hosman and Kuennapas [65] compare the similarity and difference of lowercase letters and found that the agreement (product moment correlation) between the judgments to be -0.98 and the slope of the regression line to be -0.91. Slope of the regression line (or Pearson correlation coefficient) ranges between -1 to 1. Therefore, the strong negative correlation coupled with the high inter-judge agreement observed by Hosman and Kuennapas indicate that similarity is inversely related to distance. Tversky [153] conduct an experiment using 21 pairs of country names. Similarity and distances scores are assigned to each pair of country names using a 20-point scale. The results of the experiment show that the sum of the two judgments for each pair is quite close to 20 in all cases. Moreover, the product moment correlation between the scores is -0.98.

However, there is also a considerable amount of psychological experimental evidence that suggest otherwise. Hollingworth [64] asked people to rank samples of handwriting with respect to their similarity or difference from a standard. Each participant must rank similarities on two occasions and differences on two occasions. He found that the degree of correlation between the two difference judgments to be greater than the degree of correlation between similarity judgments. Hollingworth interpreted this experimental result as a consequence of difference judgments tend to be based on fine details, whereas similarity judgments are based on more general criteria. This can also be thought to be the case with antonyms. Antonyms often share almost the same set of attributes. However, only one attribute value is different between a pair of antonyms. For example, consider the antonyms *hot* and *cold*. One can use both hot and cold as adjectives to modify almost the same set of nouns (i.e. hot day vs. cold day, hot water vs. cold water, etc.). However, the value of the attribute *temperature* is contrastingly different between the two antonyms.

Tversky [153] propose a model of similarity in which he considers both common and distinct features of two objects to compute the similarity between them. This model is popularly known as the *feature mode* or the *contrast model* of similarity, and is defined as follows,

$$S(A, B) = \theta f(A \cap B) - \alpha f(A - B) - \beta f(B - A). \tag{1.1}$$

Here, $S(A, B)$ is the similarity between two objects A and B. It is given as the weighted sum of three sets of features: $f(A \cap B)$ (the set of features in common to both A and B), $f(A - B)$ (the set of features that appear only in A), and $f(B - A)$ (the set of features that appear only in B. Parameters θ, α, and β respectively determine the weight assigned to the three sets of features.

In his experiment using names of countries, Tversky [153] observed that the more well-known or prominent a pair of countries is, more likely that it gets selected as both similar and dissimilar by humans. He set up an experiment in which he constructed 20 pairs of four countries on the basis of a pilot test. Each set included two pairs of countries: a prominent pair (i.e. countries that were well-known to the human subjects such as, USA and USSR), and a non-prominent pair (i.e. countries that are not well-known to the human subjects to the degree of prominent pairs such as, Tunis and Morocco). All subjects were presented with the same 20 sets. One group of 30 subjects selected between the two pairs in each set the pair of countries that were more similar, whereas, a different group of 30 subjects selected between the two pairs in each set the pair of countries that were more different. The results of this experiment show that on average, the prominent pairs are selected more frequently than the non-prominent pairs in both the similarity and difference tasks. Tversky reports that, for example, 67% of the subjects in the similarity group selected West Germany and East Germany as more similar to each other than Ceylon and Nepal,

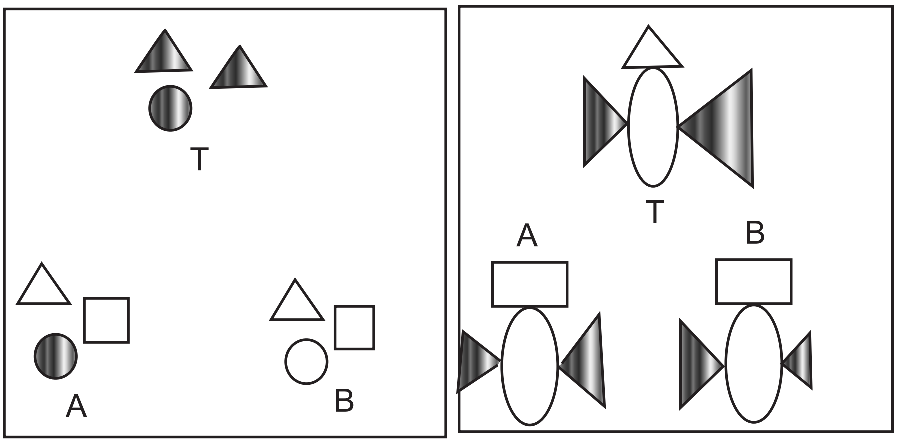


Figure 1.1: Two sample stimuli used by Medin et al. [99] in their experiments.

while 70% of the subjects in the difference group selected West Germany and East Germany as more different from each other than Ceylon and Nepal. This interesting result can be interpreted within the feature model. When measuring the similarity between two objects using Formula 1.1, humans assign a high weight to common features than distinct features (i.e. $\theta > \alpha$ and $\theta > \beta$), whereas, when measuring the difference between two objects they emphasize the unique features of each object (i.e. $\theta < \alpha$ and $\theta < \beta$).

Medin et al. [99] propose another line of argument to explain why similarity measurements and difference measurements might not be inversely related. They provide experimental evidence to support their hypothesis that relative weighting of attributes and relations depends on whether we are measuring similarity or difference. They conducted two interesting experiments using visual stimuli like the ones shown in Figure 1.1. In Figure 1.1, the picture on the left (geometric) shows two samples A and B being compared against a standard, T. The standard T is attributionally similar to A because they both have a checkered circle. B does not contain this attributional similarity to T. However, T and B are relationally similar because they both share the relation, “same-shading”. The picture on the right (butterfly) in Figure 1.1 shows three butterflies. T and A are relationally similar, because in both cases the left wing is smaller than the right wing. T is attributionally more similar to B because the left wings of the two butterflies are of the same size.

In their experiments, they asked human subjects to assign similarity and difference judgments to the visual stimuli. Results of their experiments show that when human subjects make similarity judgments they focus more on relations, whereas, when they perform difference judgments they tend to emphasize attributes. To analyze their results they plot the proportion of times the alternative with the extra relational match was picked on the similarity trials against the proportion of times it was picked on difference trials. The relational choice was made 58% of the time for the geometric stimuli and 59% time for the butterfly stimuli. If there was an inverse relation between similarity and difference judgments then these values must be 50% and the majority of data points must lie on the line. However, in their experiments they observed that the majority of data points lie above the line ($y = -x$). Tversky's feature model does not explicitly incorporate relations between objects. However, considering the duality of attributes and relations (described later in Section 1.8), we can represent a relation R between two objects A and B as a shared feature between A and B, thereby incorporating relations in an implicit manner within the feature model. However, Medin et al. [99] show that even after incorporating relations into the feature model, it still cannot explain the result of their experiment. I will next explain the analysis that is originally presented by Medin et al.

Consider the left picture in Figure 1.1. Let A_c and A_u respectively be the features of being checkered and not being checkered, R_s and R_d respectively be the relation of same shading and different shading. Standard T is checkered and the circle and the two triangles have the same checkered shade. Therefore, the set of features for T can be written as, $T = \{A_c, R_s\}$. In A, the circle is checkered as in T, but the shadings among the three shapes are different (there are both checkered and non-checkered shapes in A). Consequently, the set of features for A can be written as, $A = \{A_c, R_d\}$. Likewise, the set of attributes for B can be written as $B = \{A_u, R_s\}$. Next, using Formula 1.1, we can write the similarity of T to A, and T to B as follows,

$$S(T, A) = \theta f(A_c) - \alpha f(R_s) - \beta f(R_d), \tag{1.2}$$

$$S(T, B) = \theta f(R_s) - \alpha f(A_c) - \beta f(A_u). \tag{1.3}$$

Then, the difference in similarity of choice B to the target T and the choice A to the target T can be computed by subtraction

$$S(T,B) - S(T,A) = \theta f(R_s - A_c) - \alpha f(A_c - R_s) - \beta f(A_u - R_d). \tag{1.4}$$

For B to be selected both as more similar and more dissimilar to T than A, both sides of Formula 1.4 must be positive for similarity judgments and negative for dissimilarity judgments. Only the parameters θ, α, and β can vary from one stimulus to another. Note that both the checkered/non-checkered attribute and same/different shading relation are complementary (i.e. in Figure 1.1 the presence of A_c automatically negates A_u and vice-versa). Therefore, we can assume that attributes and relations are counterbalanced in this example. If we define $R_s = R_d = \phi$ and $A_c = A_d = \sigma$, then Formula 1.4 reduces to

$$S(T,B) - S(T,A) = (\theta + \alpha + \beta) f(\phi - \sigma). \tag{1.5}$$

From Formula 1.5 we see that by adjusting the weights we cannot change the sign of $S(T,B) - S(T,A)$. Therefore, we cannot use the feature model to explain the experimental results observed by Medin at al [99].

In summary, results from psychological experiments have shown that in some situations similarity judgments and difference judgments do not have a clear inverse relation. Two explanations are proposed in previous work investigating this phenomenon. The first proposal, which was made by Tversky is that how well-known the two objects being compared to the human subjects (more popular a pair of objects, the more likely it gets classified as both similar and dissimilar). The second proposal, which was made by Medin et al. is that whether relations or attributes are being focused on (human subjects make similarity judgments when they focus more on relations, whereas, when they perform difference judgments they tend to emphasize attributes). The two proposals do not contradict each other nor do they overlap in the sense that they explain different exceptions to the conventional view-point that similarity and difference (distance or dissimilarity) are inversely related. In Chapter 2, I propose an alternative model of semantic similarity: the *relational* model of similarity.

It is worth mentioning that contrast model is not the only feature-based model of similarity. Other combinations of its components $f(A \cap B)$, $f(A - B)$, and $f(B - A)$ are also possible. Tversky [153] in his seminal paper proposes the *ratio model* as another matching function. Ratio model is defined as follows,

$$S(A, B) = \frac{f(A \cap B)}{f(A \cap B) + \alpha f(A - B) + \beta f(B - A)}. \tag{1.6}$$

Therein: parameters $\alpha, \beta \geq 0$. Unlike the contrast model defined in Formula 1.1, the ratio model has the nice property that the similarity scores are normalized to the range $[0, 1]$. Moreover, it has one parameter (θ) less than the contrast model. Other combinations based on popular set theoretic association measures such as Jaccard coefficient, Overlap coefficient, and Dice coefficient are also possible. I will describe those association measures in details in the next chapter. The debate between similarity vs. distance is not yet settled. There can be other reasons for similarity judgments and distance judgments to behave different (i.e. not necessarily inversely), besides the two reasons mentioned above. Further psychological experiments will shed more light on this matter. However, it must be noted that in most applications of similarity or distance measures in natural language processing field this distinction has been ignored for more practical reasons. For example, in the vector space model, documents are represented by a vector of their words (typically tfidf weighted vectors computed under the bag-of-words model) and the similarity (or distance) between two documents is computed using cosine similarity or Euclidean distance between the corresponding document vectors. For example, if we consider two L_2 normalized vectors $\mathbf{a}$, and $\mathbf{b}$ (i.e. $||\mathbf{a}||_2 = ||\mathbf{b}||_2 = 1$), the following inverse relation holds between the cosine similarity $\cos(\mathbf{a}, \mathbf{b})$, and Euclidean distance $\mathrm{Euc}(\mathbf{a}, \mathbf{b})$,

$$\mathrm{Euc}(\mathbf{a}, \mathbf{b}) = 2(1 - \cos(\mathbf{a}, \mathbf{b})).$$

1.5 Symmetric vs. Asymmetric Similarity

As discussed in Section 1.4, a distance metric is expected to satisfy the symmetry constraint. However, whether this constraint is satisfied with similarity measures is a question that has received wide attention in cognitive science and psychology. If a similarity measure S is symmetric, it must satisfy $S(A, B) = S(B, A)$ for any two objects A and B. Tversky [153] conducted an experiment in which 77 human subjects rated 21 pairs of country names for their similarity using a 20 point scale. The pairs were constructed in a way that one of the countries in a pair is prominent (well-known to the human subjects) than the other. For example, the test set contained pairs of country names such as, (USA,Mexico), (Red China,North Vietnam), (Belgium,Luxemburg), etc. The ordering of two countries in a pair was determined by asking 69 subjects to select in each pair the country they regarded as more prominent. Tversky reports that two thirds of the subjects agreed with the original ordering. A second additional evaluation of the dataset was conducted by asking a different group of 69 human subjects to choose between the two phrases *"country A is similar to country B"* or *"country B is similar to country A"*. He found that in all 21 pairs, most of the subjects chose the phrase in which the less prominent country served as the subject and the more prominent country served as the referent. Moreover, the average similarity for all pairs (A, B) where A is the prominent country and B is the less prominent country, was significantly different ($p < 0.01$, $t(20) = 2.92$ in paired t-test) from that for pairs (B, A). Interestingly, when the subjects were asked to judge for difference instead of similarity, Tversky found that again the average difference ratings were also significantly different for the two orderings (A, B) and (B, A). This experiment shows that similarity as well as difference is not symmetric. It is noteworthy that both contrast model and ratio model are in general asymmetric. For example, in the contrast model (Formula 1.1), $S(A, B) = S(B, A)$, if and only if $(\alpha - \beta)f(A - B) = (\alpha - \beta)f(B - A)$. This condition can be satisfied either if $\alpha = \beta$ or $f(A - B) = f(B - A)$. The first case indicates that the task is non-directional because we consider features that are unique to A and B equally. In this case, we can reduce the contrast model to the following more simpler model,

$$S(A, B) = \theta f(A \cap B) - \alpha(f(A - B) + f(B - A)). \tag{1.7}$$

The second term in Formula 1.7 is the set of features that are not in common between A and B. Therefore, a symmetric similarity measure can be defined as the combination of common and distinct features between two objects. Moreover, the two sets of features are independent and their sum yields the union of individual objects' feature set. Lin [87] proposed an information theoretic formalization of Formula 1.7 to compute the attributional similarity between words.

The second case where a similarity measure can be symmetric is when $f(A - B) = f(B - A)$. If the feature additivity holds, then in this case we have $f(A) = f(B)$. In other words, the two sets of features representing A and B must be equal when measured using f. In this case, the contrast model can be simplified to,

$$S(A, B) = \theta f(A \cap B) - (\alpha + \beta) f(A - B). \tag{1.8}$$

Note that in both cases Formulas 1.7 and 1.8 express the same functional form. Similarity can be reduced to a linearly weighted combination of common features and different features of two objects A, B under the feature model – both features unique to A and B being weighted equally. In particular in the second case, whenever $\alpha > \beta$ we have $S(A, B) > S(B, A)$ iff $f(B) > f(A)$. If we assume the parameters α and β as a measure of salience of a set of features, then the direction of asymmetry can be interpreted as less salient a stimulus is more similar to the stimulus than vice versa.

Despite the evidence from psychological experiments that similarity can be asymmetric, most semantic similarity measures proposed to measure the similarity between words are symmetric. The design choice of symmetric measures is more of a practical one. For example, applications that use similarity measures such as document clustering with cosine measure between tfidf weighted term vectors, require similarity to be symmetric. However, there has been some work on asymmetric similarity measures that use ontologies such as the *Ontology Structure based Similarity* (OSS) measure proposed by Schickel-Zuber and Faltings [134]. In the OSS measure an ontology is considered as a directed acyclic graph (DAG). This is true for most real-world ontologies including WordNet. It then assigns an a-priori score to each node in the DAG. The similarity between two nodes (or concepts in an ontology) is then computed as the transfer of score between the two nodes. The OSS measure makes three assumptions to simplify the computation of the a-priori score: (a) the score depends on features of the concept, (b) each feature contributes independently to the score, and (c) unknown and disliked features make no contribution to the score. Additionally, they assume that a-priori scores are uniformly distributed in the range $[0, 1]$ over all nodes in the DAG. The estimated a-priori score can be computed under these assumptions to be $1/(n + 2)$, where n is the number of descendants of a node. For leaf nodes (i.e. $n = 0$), their a-priori score is 0.5. When we travel up the hierarchy, both the a-priori score of a node as well as the difference of a-priori scores between nodes decrease. To compute the similarity between two concepts in the ontology, they find the path that connects the two concepts via the lowest common ancestor (LCA). In the case of a tree, LCA of two nodes is uniquely defined. However, in a more general DAG, there can be multiple LCAs. Schickel-Zuber and Faltings propose a heuristic criterion to select a path through LCAs in a DAG that considers both the depth of the LCA (i.e. distance of the longest path between the root of the DAG and the LCA), and its reinforcement (i.e. the number of different paths that connect the LCA to the root of the DAG).

When traveling upwards from a node x to the LCA y, we remove features. Because the score of a node only depend on its set of features (assumption (a)), and features are independent (assumption (b)), the score that transfers from a node to another node in an upward movement is determined by the ratio of the a-priori scores,

$$\begin{aligned} T(x, y) &= \alpha(x, y), \\ \alpha(x, y) &= APS(y)/APS(x), \end{aligned} \tag{1.9}$$

where we define APS as the a-priori score of a node. However, when we travel downwards from LCA y to its descendant z, we add new features to concepts. It follows from the assumptions in OSS measure and that we must add the difference in a-priori score that can be attributable to the new features to the score of the current node. Moreover, assuming the score of the initial node to be 0.5 (under the assumption that score of any node is uniformly distributed as discussed earlier), for the transfer of scores in the downward inference is given by,

$$\begin{aligned} T(y, z) &= \frac{1}{1 + 2\beta(x, y)}, \\ \beta(x, y) &= APS(z) - APZ(y). \end{aligned} \tag{1.10}$$

Furthermore, Schickel-Zuber and Faltings convert the OSS measure to a distance metric that satisfies the triangular inequality as follows,

$$D(x, z) = -\frac{\log(T(x, z)}{\max D}. \tag{1.11}$$

Here, $\max D$ is the longest distance between any two concepts in the ontology. $D(x, z)$ is normalized to the range $[0, 1]$ and can be shown to satisfy the triangular inequality. Experimentally Schickel-Zuber and Faltings show that the OSS measure outperforms existing semantic similarity measures on two ontologies: WordNet and GeneOntology.

1.6 Transformational Model of Similarity

Hahn et al. [57] define similarity between two representations as the complexity required to transform one representation into the other. Their model of similarity is based on *Representational Distortion* theory, which aims to provide a theoretical framework of similarity judgments. If a complex transformation is required to convert one representations into another, then they hypothesize that the similarity between the objects that correspond to those representations must be low. The representation of an object, which transformations are allowed on a representation, and how to measure the complexity of a transformation, are all important choices in this transformational model of similarity. For example, the edit distance (or Levenshtein distance) between two words is a measure of complexity of lexical transformations between two words. In edit distance, the transformation operations are deletion, insertion, and substitution of a letter. Each operation is associated with a cost. The sequence of transformations with minimum cost that transforms one word to the other is computed using dynamic programming. Edit distance have been used to find approximate matches in a dictionary. It can be operated on phonemes instead of letters to find words that "sounds similar". For example, the words "spat", "at" and "pot" are found as similar in sound to "pat" using this method. Edit distance has been used to correct spelling mistakes by comparing misspelled words against a dictionary.

Kolomogorov [82] complexity of a string is defined as the length of the computer program that can generate the string. For a representation x, its Kolmogorov complexity is

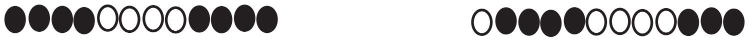


Figure 1.2: A sample stimuli used by Hanh et al. [57]

written as $K(x)$. The length of a computer program can be measured by the number of machine instructions required to execute the program. If x can be generated by a simple program (i.e. few lines of code), then $K(x)$ is small. For example, a string containing only the digit 1 can be generated by a very simple program irrespective of the length of x. On the other hand, the text in a novel cannot be generated by a simple computer program. It is noteworthy that the value of the Kolmogorov complexity is independent of the actual programming language (i.e. C++, Java, Python, etc.). In fact, the invariance theorem of Kolmogorov complexity theory states that the Kolmogorov complexity of a representation is invariant, up to a constant additive factor, with respect to the choice of programming language.

Hahn et al. [57] conduct a series of interesting experiments to check whether there is an inverse relation between the number of transformations required to convert one object to another, and the perceived similarity by humans between the two objects under consideration. In one of their experiments, they asked 35 human subjects to evaluate stimuli, like the one shown in Figure 1.6, for similarity. Two sequences of filled and unfilled circles are shown in Figure 1.6. The basic transformations allowed are mirror imaging, reversal, phase shift, insertion, and deletion. The sequence on the right in Figure 1.6 can be generated from the sequence on the left by performing a phase shift of one circle to the left. Therefore, the two sequences can be considered as being highly similar. In their experiments, Hanh et al. showed 56 pairs of stimuli of different number of transformations. In some pairs, only a single transformation is needed to convert one sequence to the other, whereas some sequences require a combination of basic transformations. Human subjects rated each pair of stimuli for their similarity using a scale of 1 (very dissimilar) to 7 (very similar). Finally, the correlation between human ratings and the number of transformations required to convert one sequence to the other in each pair of stimuli was plotted. Interestingly, they observed a significantly high negative correlation between those measures (i.e. Spearman's rank correlation coefficient of -0.69 ($P < 0.005$)). Moreover, the inter-judge concordance was high as 25 of 35 human subjects showed significant correlations.

If we define the color of a circle at a specific position as a feature of a sequence of circles, like the two sequences shown in Figure 1.6, then we can use the contrast model [154] to measure the similarity between the two sequences in Figure 1.6. The circles at positions $2 - 4$, $6 - 8$, and $10 - 12$ have the same color. Therefore, the overlap of features in the two sequences is $9/12$ or 0.75. Despite the fact that only a phase shift of a single circle maps one sequence to the other, the similarity computed based upon contrast model drops by 75%. Moreover, the drop in similarity increases rapidly if we further shift the two sequences by phase. This observation was confirmed by the results of their experiments that showed a low correlation between human ratings and similarity scores computed based on the contrast model (i.e. Spearman rank correlation coefficient of -0.28 ($P < 0.05$)).

However, it must be noted that similarity scores computed using contrast model (or any feature model for that reason) depends on the definition of features and the definition of the overlap measure. A different definition of features that can absorb a phase shift would produce a high similarity score for the two sequences in Figure 1.6. For example, one can define the color of the next circle as the feature of the current position in a sequence. Under this definition, we get a perfect similarity score for the two sequences in Figure 1.6. However, to define such features we must first realize that there exist a simple transformation between the two sequences. Hanh et al. argue that the "features" upon which featural models of similarity are based on are salient because of the perceived transformational relationships which are fundamental to the representational distortion theory. In their account, transformational relations are primal and the existence of features depends on those relations. This *relations first* view-point of similarity is emphasized further in structure mapping theory [45].

However, transformational similarity is not without unresolved issues. For example, it is not clear how to explain the asymmetries in similarity observed in previous studies ([154, 99]) using representational distortion theory. To support asymmetry a transformation and its inverse must have different effects. Moreover, Hanh et al. do not consider the fact that certain transformations are more complex than others. For example, a phase shift vs. mirroring or phase shift of one circle vs. phase shift of two or more circles. Associating a weight to each transformation could provide a partial solution to this problem.

However, a function that weights a transformation is not readily obvious. For example, for the phase shift transformation, we must have a periodical function such that the value of the weighting function remains invariant under a shift by the total length of the sequence. The task of finding simple transformations between given representations can be as difficult or easy as finding salient features in the feature model. Given N distinct objects to be compared against each other, in the feature model we only need to find features for each object once. However, in the transformational model we must find transformations for each pair of objects, which is $N(N-1)/2$. This quadratic computational complexity required by the transformational model can easily turn out to be prohibitively large if we consider an application such as ranking similar documents on the web. Moreover, pre-computing is difficult in an online setting with the transformational model, because we will have to find transformations between a new object and each of the existing objects. Some of the issues mentioned in this paragraph are not unique to transformational model of similarity and can be associated with any model of similarity that takes a relational representation, such as the structure mapping theory or the relational model of attributional similarity described in Chapter 7.

1.7 Challenges in Web-based Similarity Measurement

Web is the largest collection of text available to any natural language processing algorithm. The latest estimates of the web reports well over 10 billion web pages. Web is a dynamic corpus where new words are constantly being coined and existing words are used in novel senses. It is also a collection of knowledge written by a large number of humans. Web is an invaluable source of information for similarity measuring algorithms. The algorithms presented in this thesis utilizes the web to measure both attributional similarity between a pair of words and relational similarity between two pairs of words. Semantic similarity between two words is measured subjectively by a set of human annotators in benchmark datasets such as the Miller-Charles' dataset, Rubenstein-Goodenough's dataset, and WordSimilarity-353 dataset. However, all these datasets average the similarity scores assigned by a small number of human annotators. For example, in the case of Miller-Charles' dataset, the similarity score assigned to a pair of words is the average score by 38 human subjects. In contrast, web is a collection of documents written by millions if not billions of people. Therefore, the web can be considered as the best sample expressing the human notion of similarity, hence an attractive source of information for any semantic similarity algorithm. Web has been be used successfully for various tasks in previous work in natural language processing. Lapata and Keller [74] experimentally showed that simple, unsupervised models can be improved when n-gram counts are obtained from the web rather than from a large text corpus. Resnik and Smith [128] extracted bilingual sentences from the web to create parallel corpora for machine translation. Turney [147] defined a information retrieval (IR) inspired pointwise mutual information (PMI) measure using the number of hits returned by a web search engine. By measuring the association between nouns and adjectives that describe positive or negative (PN) sentiment (e.g. *good* and *bad*) using pointwise mutual information, Turney [144] proposed a method to classify nouns according to the sentiment associated with them on the web.

Despite the favorable arguments presented above for using the web for measuring similarity using the web as a corpus, any algorithm that attempts to process text on the web must overcome several unique challenges. First, the huge scale of the web means that we cannot run our algorithm on the entire text on the web. Despite the storage technique developed for web-scale data processing such as the Google File System [52] and distributed computational models such as the MapReduce [41] model, it is not feasible to run a state-of-the-art similarity measure using the entire text on the web because of computational cost. In contrast, the algorithms described in this thesis use web search engines as an interface to the vast information available on the web. Specifically, we use page counts and snippets retrieved from a web search engine as the input to the proposed similarity measures. Page count of a query indicates how many of the indexed web pages contain that query. This is different from the number of hits of query because unlike for hits, for page counts, multiple occurrences of a query in a page are not considered. However, most commercially available large scale web search engines provide only an estimated value for page counts. Therefore, page counts are not exact counts as one would obtain from a fixed text corpus. Moreover, considering the scale and the noise in web text, one must account for the random occurrences of words. Therefore, small values of page counts are not statistically reliable. Snippets are provided by most commercial search engines alongside with search results. A snippet is a brief window of text, often of length 10-15 words, extracted from a web document from the neighborhood of the query in that document. Snippets are useful in web search because most of the times a user can read the snippet and decide whether a search result is relevant for his or her query without actually opening a url. If a search result is clearly irrelevant, then the user does not need to click on the url pointing to the web document, thereby saving time. Snippets have been recognized as an important component in a search engine and have received much attention lately[2]. Snippets are used as an approximation of the local context of words by the algorithms described in this thesis. However, most commercially available search engines impose an upper limit (e.g. 1000) for the number of search results (thereby snippets) that one can retrieve for a single query. This is both for efficiency reasons as well as due to the fact that relevance of search results drops with the ranking. Therefore, an algorithm that uses snippets as the input is limited to only a fraction of search results. To circumvent the above mentioned limitations associated with page counts and snippets, the algorithms proposed in this thesis employ a range of techniques such as, issuing multiple queries and aggregating search results, use snippets to generate subsequence lexical patterns and use page counts to compute their frequency on the web, use machine learning approaches to integrate both page counts and snippets in a single model, and use anchor text co-occurrence statistics.

The quality and the level of noise in web text is a challenge for most natural language processing systems. In particular, a large number of new words (neologisms) that are not registered in manually compiled dictionaries such as WordNet exist on the web. For example, in Japanese, novel or borrowed words are written in katakana script. Consequently, processing of words written in katakana is an important component in any Japanese morphological analyzer. Considering newspaper articles, which are written by professional writers and proof read, the texts found on the web such as Blog entries or on bulletin board services sometimes do not contain proper punctuation or not properly structured. Algorithms tuned using newspaper corpora have shown suboptimal performance when evaluated on web texts. The noise in web texts makes most basic text processing tasks such as part-of-speech (POS) tagging, noun phrase (NP) chunking, syntactic or dependency parsing, or named-entity recognition (NER) difficult to perform with a high accuracy.

[2] http://www.wssp.info/2009.html

The high redundancy in web means that we can expect to find the same piece of information duplicated in multiple web sites. This can be simply due to the mirroring of web sites or different authors writing about the same event. However, it is not always the case that all web sites agree upon the information they provide. Sometimes web sites might provide contradictory information. Some web sites intentionally provide false information or contain web spam. The reliability of information is a difficult challenge for any web-based NLP system. Two working hypothesis are used indirectly by the algorithms described in this thesis as a partial solution to the problem of determining the reliability of input. First, if a statement is repeated in a large number of web sites then we can assume it as true. The proposed algorithms only consider lexical patterns that occur more than a predefined number of times on the web (i.e. lexical patterns that have page counts more than a predefined lower limit). Second, if a web page is linked by many other web pages (i.e. has a large number of inbound links) then the reliability of that web site can be considered to be high. The PageRank algorithm [111] utilizes such citation information to compute the static rank of web pages. The proposed method only processes the top ranking search results, thereby attempts to avoid processing unreliable information on the web. This is by no means a perfect solution and further studies must be conducted.

The use of commercial search engines for natural language is processing tasks has several drawbacks [75]. Commercial search engines are tuned for one or two word queries issued by web search users and do not always have elaborated searching techniques required by NLP applications such as regular expression matching or searching by a particular tag. Commercial search engines does not perform POS tagging or deep parsing of text that is often required by NLP applications. Moreover, the number of queries that can be issued is limited. The search APIs provided by commercial search engines entangle adverts etc. and limit the number of results that can be retrieved at a time. Moreover, the ranking algorithms used by commercial search engines is often a complex combination of various factors such as authority of a page, novelty of the content, structure of the page, paid content such as advert keywords, and refresh/update rate of the page. The exact ranking algorithm is not disclosed publicly. There is no guarantee that an NLP system can find the information that it requires within the top ranked search results by a commercial search engine. There has been several initiatives to design search engines for natural language processing applications [75]. However, they still lack the scales achieved by the commercial search engines. The algorithms proposed in the thesis use shallow linguistics approaches such as stemming, stop word removal and lexical patterns. They do not require POS tagging, NER or syntactic/dependency parsing that are currently unavailable in commercial search engines.

The web is expected to grow continuously. Web-based NLP algorithms must be designed in such a way that the ever increasing size of the web has a positive effect on the performance of the algorithm. Techniques that does not slow down an algorithm when the size of the web increases are desirable. In this regard, the use of web search engines as the interface to the vast information available on the web is attractive. Irrespective of the size of the web, the input to the proposed similarity measures will remain at a constant size because I only use the page counts and the top ranked snippets in the proposed algorithms. Moreover, the increasing size of the web means that we can expect the redundancy of information also to increase in future. This is beneficial to any algorithm that use statistical information such as page counts of web queries because we can expect to obtain more reliable page counts when the web grows.

1.8 Attribute-Relation Duality

I defined attributional similarity as the correspondence between attributes of two words, whereas relational similarity is the correspondence between the relations that exist between two pairs of words. These definitions implicitly assume that we have a clear distinction between attributes and relations. Attributes are commonly viewed as properties of entities, whereas relations are connections between two or more entities. From a predicate logical view-point, attributes are predicates that take only one argument, whereas relations can take two or more arguments. For example, if an object X is heavy then we can say that HEAVY(X) is TRUE. Being heavy is an attribute of X. We can also express the fact that object X is heavier than an object Y by stating that HEAVIER(X, Y) is TRUE. Here, the predicate HEAVIER expresses the relation *is heavier than* between X and Y.

Because the definition of *relations* as given above involves the consideration of more than one entity, typically *attributes* are considered to be more primal than relations. From a generative view-point, attributes exist first and relations materialize when we recognize the connections between the attributes of different objects. Under this view-point relational are secondary. We will call this the *attribute-primal* view-point. However, in theories of analogical learning such as the structure mapping theory, relations are considered more important than attributes. According to this view-point relations are primal. Relations exist first and entities are formed by instantiating the relations. Under this view-point attributes are secondary. We will call this the *relation-primal* view-point. For example, when we say that X is heavy, we are implicitly *comparing* X with other objects known to us (e.g. Y).

Compared to the relations-primal view-point, the attributes-primal view-point has received much attention. For example, in computational linguistics numerous attributional similarity measures have been proposed and successfully used in a variety of tasks. However, only recently there has been some work on relational similarity measures. Moreover, relational similarity measures are mostly evaluated on solving word analogy questions and yet to be evaluated in real-world applications. The comparatively lesser applications of relational similarity measures can be attributable to the difficulty of measuring relational similarity between real-word entities.

In machine learning algorithms, training/testing cases are frequently represent as feature vectors. Features are defined for individual objects and attributes of the object are selected as features. For example, in a discriminative learning algorithm such as Support Vector Machines (SVMs) or Conditional Random Fields (CRFs) would *learn* the features that discriminates (separates) one class (i.e. positive) from another (i.e. negative). The success of discriminative learning algorithms in numerous tasks have enforced the attribute-primal view-point. However, recently there have been some work on statistical relational learning algorithms such as the Probabilistic Relational Model [51] that take a relation-primal view-point.

Turney [145] experimentally showed that relational similarity measures can be used to measure attributional similarity in the case of proportional analogies. Proportional analogies are a subset of analogies and can be expressed in the form A:B::C:D. It asserts that A is to B as C is to D. For example, stone:mason::wood:carpenter asserts that stone is to mason as wood is to carpenter. If the two word pairs (A,B) and (C,D) involved in the proportional analogy possess a high degree of relational similarity and A and B are synonymous then we can expect C and D also to be synonymous. In fact, he provides experimental evidence to the fact not only synonyms but also antonyms can be learned by this approach. From the above argument it follows that in some cases (i.e. proportional analogies) relational similarity subsumes its attributional counterpart. However, Turney has also shown experimentally that the use of attributional similarity measures to solve word analogy problems yield poor results. Specifically, using an attributional similarity measure sim_a he defined the relational similarity sim_r in a proportional analogy $A{:}B{::}C{:}D$ as follows,

$$\mathrm{sim}_r(A : B :: C : D) = \frac{1}{2}(\mathrm{sim}_a(A, B) + \mathrm{sim}_a(C, D)). \tag{1.12}$$

Turney [150] used numerous previously proposed attributional similarity measures as sim_a and computed the performance in solving 374 SAT word analogy questions. The highest F-score of 35.0 was obtained using the attributional similarity measure proposed by Turney [146] using pointwise mutual information. However, this is still statistically significantly low compared to the F-score reported when a relational similarity measure such as the Vector Space Method-based relational similarity measure (VSM) [151] is used (ca. 47.0). From the above mentioned empirical results we can conclude that at least in the case of proportional analogies, relational similarity is more primal compared to attributional similarity. A future direction of research in similarity measure would be to confirm whether this is the case for more complex analogies. The debate as to which view-point is primal is still open.

1.9 Overview of the thesis

The remainder of this thesis is organized as follows. In Chapter 2, I introduce the problem of measuring attributional similarity from the web. A supervised approach that uses web search engines is proposed to measure attributional similarity between two words. Next, in Chapter 3 I present a supervised approach to measure relational similarity from the web. The proposed relational similarity measure is evaluated on SAT word analogy dataset as well as on a named entity dataset. The second half of this thesis is dedicated to applications of semantic and relational similarity measures. As a working problem, I consider the person name disambiguation on the web. In Chapter 4, I analyze the namesake disambiguation problem (i.e. disambiguating different people with the same name), whereas, in Chapter 5, I study the name alias problem (i.e. single individual represented by different names). I propose fully automatic methods to solve both these problems with high accuracy. In Chapter 7, I propose a relational model to compute the attributional similarity between two words. Finally, in Chapter 8 I discuss potential future research directions in this field and conclude this thesis.

Chapter 2

Semantic Similarity

Accurately measuring the semantic similarity between words has been an important problem in Semantic Web. information retrieval, and natural language processing. Semantic Web applications such as, community extraction, ontology generation, and entity disambiguation, require the ability to accurately measure semantic relatedness between concepts or entities. In information retrieval, one of the main problems is to retrieve a set of documents that is semantically related to a given user query. Efficient measurement of semantically similarity between words is critical for various natural language processing tasks such as word sense disambiguation (WSD), textual entailment, and automatic text summarization.

Manually created general-purpose lexical ontologies such as WordNet, group semantically related words (i.e. synsets) for concepts. However, Semantic similarity between entities changes over time and across domains. For example, *apple* is frequently associated with *computers* on the Web. However, this sense of apple is not listed in most general-purpose thesauri or dictionaries. A user who searches for *apple* on the Web, might be interested in this sense of apple and not apple as a fruit. New words are constantly being created as well as new senses are assigned to existing words. Manually maintaining ontologies to capture these new words and senses is costly if not impossible.

I propose an automatic method to measure semantic similarity between words or entities using Web search engines. It is not feasible to analyze each document separately and directly because of the vastly numerous documents and the high growth rate of the Web,

Web search engines provide an efficient interface to this vast information. Page counts and snippets are two useful information sources provided by most Web search engines. Page count of a query is the number of pages that contain the query words. In general, page count may not necessarily be equal to the word frequency because the queried word might appear many times on one page. Page count for the query *P AND Q* can be considered as a global measure of co-occurrence of words *P* and *Q*. For example, the page count of the query *"apple"* AND *"computer"* in *Google*[1] is $288,000,000$, whereas the same for *"banana"* AND *"computer"* is only $3,590,000$. More than 80 times numerous page counts for *"apple"* AND *"computer"* indicate that *apple* is more semantically similar to *computer* than is *banana*.

Despite its simplicity, using page counts alone as a measure of co-occurrence of two words presents several drawbacks. First, page count analyses ignores the position of a word in a page. Therefore, even though two words appear in a page, they might not be actually related. Secondly, page count of a polysemous word (a word with multiple senses) might contain a combination of all its senses. For an example, page counts for *apple* contains page counts for *apple* as a fruit and *apple* as a company. Moreover, given the scale and noise in the Web, some words might occur arbitrarily, i.e. by random chance, on some pages. For those reasons, page counts alone are unreliable when measuring semantic similarity.

Snippets, a brief window of text extracted by a search engine around the query term in a document, provide useful information regarding the local context of the query term. Semantic similarity measures defined over snippets, have been used in query expansion [131], personal name disambiguation [16], and community mining [26]. Processing snippets is also efficient because it obviates the trouble of downloading web pages, which might be time consuming depending on the size of the pages. However, a widely acknowledged drawback of using snippets is that, because of the huge scale of the web and the large number of documents in the result set, only those snippets for the top-ranking results for a query can be processed efficiently. Ranking of search results, hence snippets, is determined by a complex combination of various factors unique to the underlying search engine. Therefore, no guarantee exists that all the information we need to measure semantic similarity between a given pair of words is contained in the top-ranking snippets.

[1] http://www.google.com

I propose a method that considers both page counts and *lexico-syntactic patterns* extracted from snippets, thereby overcoming the problems described above. For example, let us consider the following snippet from Google for the query ***Jaguar*** *AND* ***cat***.

> *"The* ***Jaguar*** *is the largest* **cat** *in Western Hemisphere and can subdue larger prey than can the puma"*

Figure 2.1: A snippet retrieved for the query ***Jaguar*** *AND* ***cat***.

Here, the phrase *is the largest* indicates a hypernymic relationship between Jaguar and cat. Phrases such as *also known as, is a, part of, is an example of* all indicate various semantic relations. Such indicative phrases have been applied to numerous tasks with good results, such as hypernym extraction [60] and fact extraction [112]. From the previous example, we form the pattern ***X*** *is the largest* ***Y***, where we replace the two words *Jaguar* and *cat* by two variables **X** and **Y**.

2.1 Previous Work on Semantic Similarity

Semantic similarity measures are important in many Web-related tasks. In query expansion [21, 105, 159] a user query is modified using synonymous words to improve the relevancy of the search. One method to find appropriate words to include in a query is to compare the previous user queries using semantic similarity measures. If there exist a previous query that is semantically related to the current query, then it can be either suggested to the user, or internally used by the search engine to modify the original query.

Semantic similarity measures have been used in Semantic Web related applications such as automatic annotation of Web pages [30], community mining [102, 96], and keyword extraction for inter-entity relation representation [106]. Cimiano et al., [30] proposed the PANKOW (Pattern-based Annotation through Knowledge on the Web) system to automatically annotate a web page with metadata. Given a web page to annotate, the PANKOW system first extracts candidate phrases such as proper nouns. It then classifies the extracted candidate phrases into a set of given concepts (e.g. Country, Hotel), using the number of hits returned by a Web search engine for lexical patterns such as ***X*** *is a* ***Y***, *the* ***X Y***, etc. Matsuo et al., [96] proposed the use of Web hits for extracting communities on the Web.

They measured the association between two personal names using the overlap (Simpson) coefficient, which is calculated based on the number of Web hits for each individual name and their conjunction (i.e., *AND* query of the two names).

Semantic similarity measures are necessary for various applications in natural language processing such as word-sense disambiguation [127], language modeling [129], synonym extraction [84], and automatic thesauri extraction [34]. Manually compiled taxonomies such as WordNet[2] and large text corpora have been used in previous work on semantic similarity [84, 126, 69, 87]. Regarding the Web as a live corpus has become an active research topic recently. Simple, unsupervised models demonstrably perform better when n-gram counts are obtained from the Web rather than from a large corpus [74, 79]. Resnik and Smith [128] extracted bilingual sentences from the Web to create a parallel corpora for machine translation. Turney [146] defined a point-wise mutual information (PMI-IR) measure using the number of hits returned by a Web search engine to recognize synonyms. Matsuo et. al, [97] used a similar approach to measure the similarity between words and apply their method in a graph-based word clustering algorithm.

Given a taxonomy of concepts, a straightforward method to calculate similarity between two words (concepts) is to find the length of the shortest path connecting the two words in the taxonomy [122]. If a word is polysemous then multiple paths might exist between the two words. In such cases, only the shortest path between any two senses of the words is considered for calculating similarity. A problem that is frequently acknowledged with this approach is that it relies on the notion that all links in the taxonomy represent a uniform distance.

Resnik [126] proposed a similarity measure using information content. He defined the similarity between two concepts C_1 and C_2 in the taxonomy as the maximum of the information content of all concepts C that subsume both C_1 and C_2. Then the similarity between two words is defined as the maximum of the similarity between any concepts that the words belong to. He used WordNet as the taxonomy; information content is calculated using the Brown corpus.

Li et al., [160] combined structural semantic information from a lexical taxonomy and

[2] http://wordnet.princeton.edu/

information content from a corpus in a nonlinear model. They proposed a similarity measure that uses shortest path length, depth and local density in a taxonomy. Their experiments reported a Pearson correlation coefficient of 0.8914 on the Miller and Charles [103] benchmark dataset. They did not evaluate their method in terms of similarities among named entities. Lin [87] defined the similarity between two concepts as the information that is in common to both concepts and the information contained in each individual concept.

Cilibrasi and Vitanyi [29] proposed a distance metric between words using only page-counts retrieved from a web search engine. The proposed metric is named *Normalized Google Distance* (NGD) and is given by,

$$NGD(x,y) = \frac{\max\{\log H(x), \log H(y)\} - \log H(x,y)}{\log N - \min\{\log H(x), \log H(y)\}}. \tag{2.1}$$

Here, x and y are the two words between which distance $NGD(x,y)$ is to be computed, $H(x)$ denotes the page-counts for the word x, and $H(x,y)$ is the page-counts for the query *x AND y*. NGD is based on normalized information distance [81], which is defined using Kolmogorov complexity. Because NGD does not take into account the context in which the words co-occur, it suffers from the drawbacks described in the previous section that are characteristic to similarity measures that consider only page-counts.

Sahami et al., [131] measured semantic similarity between two queries using snippets returned for those queries by a search engine. For each query, they collect snippets from a search engine and represent each snippet as a TF-IDF-weighted term vector. Each vector is L_2 normalized and the centroid of the set of vectors is computed. Semantic similarity between two queries is then defined as the inner product between the corresponding centroid vectors. They did not compare their similarity measure with taxonomy-based similarity measures.

Chen et al., [26] proposed a double-checking model using text snippets returned by a Web search engine to compute semantic similarity between words. For two words P and Q, they collect snippets for each word from a Web search engine. Then they count the occurrences of word P in the snippets for word Q and the occurrences of word Q in the snippets for word P. These values are combined nonlinearly to compute the similarity

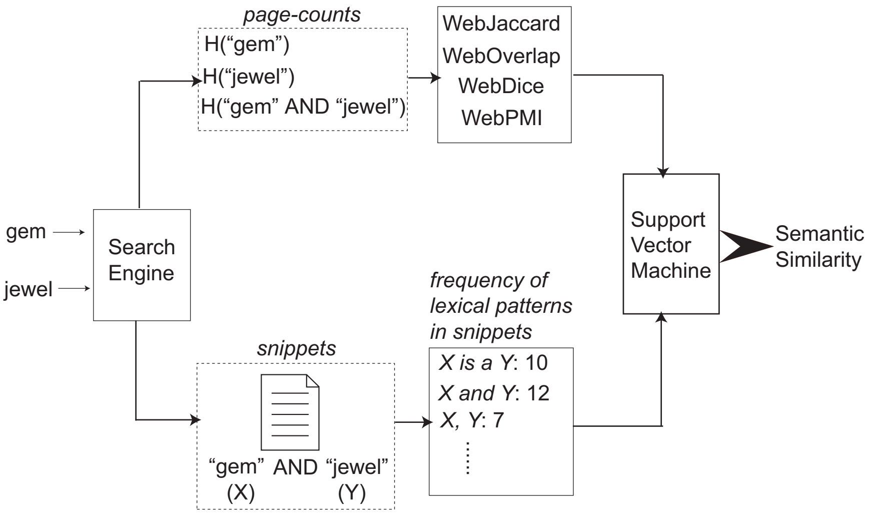


Figure 2.2: Outline of the proposed method.

between P and Q. The *Co-occurrence Double-Checking* (CODC) measure is defined as,

$$\mathrm{CODC}(P, Q) = \begin{cases} 0 & \text{if } f(P@Q) = 0 \\ \exp\left(\log\left[\frac{f(P@Q}{H(P)} \times \frac{f(Q@P)}{H(Q)}\right]^{\alpha}\right) & \text{otherwise.} \end{cases} \tag{2.2}$$

Here, $f(P@Q)$ denotes the number of occurrences of P in the top-ranking snippets for the query Q in Google, $H(P)$ is the page count for query P, and α is a constant in this model which is experimentally set to the value 0.15. This method depends heavily on the search engine's ranking algorithm. Although two words P and Q might be very similar, we cannot assume that one can find Q in the snippets for P, or vice versa, because a search engine considers many other factors besides semantic similarity, such as publication date (novelty) and link structure (authority) when ranking the result set for a query. This observation is confirmed by the experimental results in their paper which reports zero similarity scores for many pairs of words in the Miller and Charles [103] benchmark dataset.

2.2 Method

2.2.1 Outline

I propose a method which integrates both page counts and snippets to measure semantic similarity between a given pair of words. Figure 2.2 illustrates an example of using the proposed method to compute the semantic similarity between two words, *gem* and *jewel*. First, I query a Web search engine using the two words between which I must compute semantic similarity, and retrieve page-counts for the two words and for their conjunctive (i.e. *AND*) query. In section 2.2.2, I define four similarity scores using page counts. Second, I find the frequency of numerous lexico-syntactic patterns in snippets returned for the conjunctive query of the two words. The lexical patterns I utilize are extracted using the method described in section 2.2.3. I rank the patterns extracted by the proposed algorithm according to their ability to express semantic similarity. I use two-class support vector machines (SVMs) to find the optimal combination of page counts-based similarity scores and top-ranking patterns. The SVM is trained to classify synonymous word-pairs and non-synonymous word-pairs. I select synonymous word-pairs (positive training instances) from WordNet *synsets* (i.e. a set of synonymous words). Non-synonymous word-pairs (negative training instances) are automatically created using a random shuffling technique. I convert the output of SVM into a posterior probability. I define the semantic similarity between two words as the posterior probability that they belong to the synonymous-words (positive) class.

2.2.2 Page-count-based Similarity Scores

Page counts for the query *P AND Q* can be considered as an approximation of co-occurrence of two words (or multi-word phrases) P and Q on the Web. However, page counts for the query *P AND Q* alone do not accurately express semantic similarity. For example, Google returns $11,300,000$ as the page count for *"car" AND "automobile"*, whereas the same is $49,000,000$ for *"car" AND "apple"*. Although, *automobile* is more semantically similar to *car* than *apple* is, page counts for the query *"car" AND "apple"* are more than four times greater than those for the query *"car" and "automobile"*. One must consider the page counts not just for the query *P AND Q*, but also for the individual words *P* and *Q* to assess semantic similarity between *P* and *Q*.

I modify four popular co-occurrence measures; Jaccard, Overlap (Simpson), Dice, and Pointwise mutual information (PMI), to compute semantic similarity using page counts. For the remainder of this chapter I use the notation $H(P)$ to denote the page counts for the query P in a search engine. The *WebJaccard* coefficient between words (or multi-word phrases) P and Q, $\mathrm{WebJaccard}(P,Q)$, is defined as,

$$\mathrm{WebJaccard}(P,Q) = \begin{cases} 0 & \text{if } H(P \cap Q) \leq c \\ \frac{H(P \cap Q)}{H(P)+H(Q)-H(P \cap Q)} & \text{otherwise} \end{cases}. \tag{2.3}$$

Therein, $P \cap Q$ denotes the conjunction query *P AND Q*. Given the scale and noise in Web data, it is possible that two words may appear on some pages purely accidentally. In order to reduce the adverse effects attributable to random co-occurrences, I set the WebJaccard coefficient to zero if the page count for the query $P \cap Q$ is less than a threshold c[3]. Similarly, I define *WebOverlap*, $\mathrm{WebOverlap}(P,Q)$, as,

$$\mathrm{WebOverlap}(P,Q) = \begin{cases} 0 & \text{if } H(P \cap Q) \leq c \\ \frac{H(P \cap Q)}{\min(H(P),H(Q))} & \text{otherwise} \end{cases}. \tag{2.4}$$

WebOverlap is a natural modification to the Overlap (Simpson) coefficient. I define the *WebDice* coefficient as a variant of the Dice coefficient. $\mathrm{WebDice}(P,Q)$ is defined as,

$$\mathrm{WebDice}(P,Q) = \begin{cases} 0 & \text{if } H(P \cap Q) \leq c \\ \frac{2H(P \cap Q)}{H(P)+H(Q)} & \text{otherwise} \end{cases}. \tag{2.5}$$

Pointwise mutual information (PMI) [28] is a measure that is motivated by information theory; it is intended to reflect the dependence between two probabilistic events. I define *WebPMI* as a variant form of pointwise mutual information using page counts as,

$$\mathrm{WebPMI}(P,Q) = \begin{cases} 0 & \text{if } H(P \cap Q) \leq c \\ \log_2\left(\frac{\frac{H(P \cap Q)}{N}}{\frac{H(P)}{N}\frac{H(Q)}{N}}\right) & \text{otherwise} \end{cases}. \tag{2.6}$$

[3] I set $c = 5$ in my experiments

Here, N is the number of documents indexed by the search engine. Probabilities in Equation 2.6 are estimated according to the maximum likelihood principle. To calculate PMI accurately using Equation 2.6, we must know N, the number of documents indexed by the search engine. Although estimating the number of documents indexed by a search engine [8] is an interesting task itself, it is beyond the scope of this work. In the present work, I set $N = 10^{10}$ according to the number of indexed pages reported by Google.

2.2.3 Extracting Lexico-Syntactic Patterns from Snippets

Text snippets are returned by search engines alongside with the search results. They provide valuable information regarding the local context of a word. I extract lexico-syntactic patterns that indicate various aspects of semantic similarity. For example, consider the following text snippet returned by Google for the query *"cricket" AND "sport"*.

*"**Cricket** is a **sport** played between two teams, each with eleven players."*

Figure 2.3: Pattern Extraction Example

Here, the phrase *is a* indicates a semantic relationship between **cricket** and **sport**. Many such phrases indicate semantic relationships. For example, *also known as, is a, part of, is an example of* all indicate semantic relations of different types. In the example given above, words indicating the semantic relation between *cricket* and *sport* appear between the query words. Replacing the query words by variables *X* and *Y* we can form the pattern *X is a Y* from the example given above. However, in some cases the words that indicate the semantic relationship do not fall between the query words. For example, consider the following example.

*"**Toyota** and **Nissan** are two major Japanese car manufacturers."*

Figure 2.4: Pattern Extraction Example

Here, the relationship between *Toyota* and *Nissan* is that they are both *car manufacturers*. Identifying the exact set of words that convey the semantic relationship between two entities is a difficult problem which requires deeper semantic analysis. However, such an analysis is not feasible considering the numerous ill-formed snippets we need to process on the Web. I propose a shallow pattern extraction method to capture the semantic relationship between words in text snippets.

Algorithm 1 Extract patterns from snippets.

```
Input: set $S$ of word-pairs
Output: set $P$ of patterns
for wordpair $(A, B) \in S$ do
    $D \leftarrow$ GetSnippets(“$A$ $B$”)
end for
$N \leftarrow$ null
for snippet $d \in D$ do
    $N \leftarrow N +$ GetNgrams$(d, A, B)$
end for
$P \leftarrow$ CountFreq$(N)$
return $P$
```

The proposed pattern extraction algorithm is illustrated in Algorithm 1. Given a set S of synonymous word-pairs, GetSnippets function returns a list of text snippets for the query *“A”* AND *“B”* for each word-pair (A, B) in S. For each snippet found, I replace the two words in the query by two variables. Let us assume these variables to be *X* and *Y*. For each snippet d in the set of snippets D returned by GetSnippets, function GetNgrams extracts word n-grams for $n = 2, 3, 4$ and 5. I select n-grams which contain exactly one *X* and one *Y*. For example, the snippet in Figure 2.4 yields the patterns *X and Y, X and Y are, X and Y are two*. Finally, function CountFreq counts the frequency of each pattern I extracted. The procedure described above yields a set of patterns with their frequencies in text snippets obtained from a search engine. It considers the words that fall between X and Y as well as words that precede X and succeeds Y.

To leverage the pattern extraction process, I select synonymous nouns from WordNet [104] synsets. A WordNet synset contains a group of synonymous words. Different senses of a word have different synsets. The WordNet 2.0 database which I use in my experiments contains $114, 648$ nouns and $79, 689$ synsets. I randomly select 5000 nouns for which synsets with more than three entries are available. I do not select abbreviations or multi-word nouns. For polysemous nouns I select the synonyms for the dominant sense.

Table 2.1: Contingency table

	v	other than v	All
Frequency in snippets for synonymous word pairs	p_v	$P - p_v$	P
Frequency in snippets for non-synonymous word pairs	n_v	$N - n_v$	N

The pattern extraction algorithm described in Algorithm 1 yields $4,562,471$ unique patterns. Of those patterns, 80% occur less than 10 times. It is impossible to train a classifier with such numerous sparse patterns. We must measure the confidence of each pattern as an indicator of synonymy. For that purpose, I employ the following procedure.

First, I run the pattern extraction algorithm described in Algorithm 1 with a non-synonymous set of word-pairs and count the frequency of the extracted patterns. I then use a test of statistical significance to evaluate the probable applicability of a pattern as an indicator of synonymy. The fundamental idea of this analysis is that, if a pattern appears a statistically significant number of times in snippets for synonymous words than in snippets for non-synonymous words, then it is a reliable indicator of synonymy.

To create a set of non-synonymous word-pairs, I select two nouns from WordNet arbitrarily. If the selected two nouns do not appear in any WordNet synset then I select them as a non-synonymous word-pair. I repeat this procedure until I obtain 5000 pairs of non-synonymous words.

For each extracted pattern v, I create a contingency table, as shown in Table 2.1 using its frequency p_v in snippets for synonymous word pairs and n_v in snippets for non-synonymous word pairs. In Table 2.1, P denotes the total frequency of all patterns in snippets for synonymous word pairs ($P = \sum_v p_v$) and N is the same in snippets for non-synonymous word pairs ($N = \sum_v n_v$). Using the information in Table 2.1, I calculate the χ^2 [93] value for each pattern as,

$$\chi^2 = \frac{(P+N)(p_v(N-n_v) - n_v(P-p_v))^2}{PN(p_v+n_v)(P+N-p_v-n_v)}. \tag{2.7}$$

I select the top ranking 200 patterns experimentally as described in section 2.3.2, according to their χ^2 values. Some selected patterns are shown in Table 2.2.

Before we proceed to the integration of patterns and page-counts-based similarity scores, it is necessary to introduce some constraints to the development of semantic similarity measures. Evidence from psychological experiments suggest that semantic similarity can be context-dependent and even asymmetric [154, 99]. Human subjects have reportedly assigned different similarity ratings to word-pairs when the two words were presented in the reverse order. However, experimental results investigating the effects of asymmetry report that the average difference in ratings for a word pair is less than 5 percent [99]. In this work, I assume semantic similarity to be symmetric. This is in line with previous work on semantic similarity described in section 2.1. Under this assumption, I can interchange the query word markers *X* and *Y* in the extracted patterns.

2.2.4 Integrating Patterns and Page Counts

In section 2.2.2 I defined four similarity scores using page counts. Section 2.2.3 described a lexico-syntactic pattern extraction algorithm and ranked the patterns according to their ability to express synonymy. In this section I describe the leverage of a robust semantic similarity measure through integration of all the similarity scores and patterns described in previous sections.

Algorithm 2 Create a feature vector **F** for a word pair (A, B).

```
Input: word pair $(A, B)$
Output: feature vector F
$D \leftarrow$ GetSnippets("$A$ $B$")
$N \leftarrow null$
for snippet$d \in D$ do
    $N \leftarrow N +$ GetNgrams$(d, A, B)$
end for
$SelPats \leftarrow$ SelectPatterns$(N, GoodPats)$
$PF \leftarrow$ Normalize$(SelPats)$
F$\leftarrow [PF, WebJaccard, WebOverlap, WebDice, WebPMI]$
return F
```

For each pair of words (A, B), I create a feature vector **F** as shown in Algorithm 2. First, I query Google for *"A" AND "B"* and collect snippets. Then I replace the query words A and B with two variables X and Y, respectively in each snippet. Function GetNgrams extracts n-grams for $n = 2, 3, 4$ and 5 from the snippets. I select n-grams having exactly one X and one Y as I did in the pattern extraction algorithm (Algorithm 1). Let us assume the set of patterns selected based on their χ^2 values in section 2.2.3 to be GoodPats. Then, the function SelectPatterns selects the n-grams from N which appear in GoodPats. In Normalize(SelPats), I normalize the count of each pattern by dividing it from the total number of counts of the observed patterns. This function returns a vector of patterns where each element is the normalized frequency of the corresponding pattern in the snippets for the query "A" "B". I append similarity scores calculated using page counts in section 2.2.2 to create the final feature vector $\mathbf{x_i}$ for the word-pair (A_i, B_i). This procedure yields a $L+4$ dimensional (4 page-counts based similarity scores and L number of lexical patterns) feature vector. I form such feature vectors for all synonymous word-pairs (positive training examples) as well as for non-synonymous word-pairs (negative training examples). I then train a two-class support vector machine with the labeled feature vectors.

Once we have trained an SVM using synonymous and non-synonymous word pairs, we can use it to compute the semantic similarity between two given words. Following the same method I used to generate feature vectors for training, I create a feature vector $\mathbf{x}^*$ for a pair of words (A^*, B^*), between which we must measure the semantic similarity. I define the semantic similarity $\mathrm{SemSim}(A^*, B^*)$ between A^* and B^* as the posterior probability $P(y^* = 1|\mathbf{x}^*)$ that the feature vector $\mathbf{x}^*$ corresponding to the word-pair (A^*, B^*) belongs to the synonymous-words class. I denote the label assigned to a feature vector $\mathbf{x_i}$ by $y_i \in \{-1, 1\}$. Here, $y_i = 1$ denotes the synonymous-words (positive) class and $y_i = -1$ denotes the non-synonymous words (negative) class. $\mathrm{SemSim}(A^*, B^*)$ is given by,

$$\mathrm{SemSim}(A^*, B^*) = P(y^* = 1|\mathbf{x}^*). \tag{2.8}$$

Because SVMs are large margin classifiers, the output of an SVM is the distance from the classification hyperplane. The distance $f(\mathbf{x}^*)$ to an instance x^* from the classification hyperplane is given by,

$$f(\mathbf{x}^*) = h(\mathbf{x}^*) + b.$$

Here, b is the bias term and the hyperplane, $h(\mathbf{x})$, is given by,

$$h(\mathbf{x}^*) = \sum_i y_i \alpha_i K(\mathbf{x_i}, \mathbf{x}^*).$$

Here, α_i is the Lagrange multiplier corresponding to the support vector $\mathbf{x_i}$[4], and $K(\mathbf{x_i}, \mathbf{x}^*)$ is the value of the kernel function for a training instance $\mathbf{x_i}$ and the instance to classify, $\mathbf{x}^*$. However, $\mathbf{f}(\mathbf{x}^*)$ is not a calibrated posterior probability. Following Platt [121], I use sigmoid functions to convert this uncalibrated distance into a calibrated posterior probability. The probability, $P(y = 1|f(\mathbf{x}))$, is computed using a sigmoid function defined over $f(\mathbf{x})$ as follows,

$$P(y = 1|f(\mathbf{x})) = \frac{1}{1 + \exp(\lambda f(\mathbf{x}) + \mu)}.$$

Here, λ and μ are parameters which are determined by maximizing the likelihood of the training data. Log-likelihood of the training data is given by,

$$\begin{aligned} L(\lambda, \mu) &= \sum_{i=1}^{N} \log P(y_i|\mathbf{x_i}; \lambda, \mu) \qquad (2.9) \\ &= \sum_{i=1}^{N} \{t_i \log(p_i) + (1 - t_i)\log(1 - p_i)\}. \end{aligned}$$

Here, to simplify the formula I have used the notations $t_i = (y_i + 1)/2$ and $p_i = P(y_i = 1|\mathbf{x_i})$. The maximization in Formula 2.9 with respect to parameters λ and μ can be done using various optimization algorithms [109]. Platt [121] used a model-trust minimization algorithm [53] for the optimization.

2.3 Experiments

Evaluating a semantic similarity measure is a difficult task because the notion of semantic similarity varies across domains and from person to person. To evaluate the proposed

[4] From K.K.T. conditions it follows that the Lagrange multipliers corresponding to non-support vectors become zero.

method I conduct two types of experiments. First, I compare the similarity scores produced by the proposed measure against a benchmark dataset of semantic similarity. The benchmark dataset is detailed in section 2.3.1. The degree of correlation between a benchmark dataset of semantic similarity and the similarity scores produced by an automatic similarity measure, can be considered as a measurement of how well the automatic similarity measure captures the notion of semantic similarity held by humans. I analyze the behavior of the proposed measure with the number of patterns used as features, the number of snippets used to extract the patterns, and the size of the training dataset.

Secondly, I apply the proposed measure in two real-world applications: community mining and entity disambiguation. This enables us to evaluate the performance of the proposed method in measuring semantic similarity between named entities for which no manually created lexical resources such as dictionaries exist.

2.3.1 The Benchmark Dataset

I evaluate the proposed method against Miller-Charles [103] dataset, a dataset of 30 word-pairs[5] rated by a group of 38 human subjects. The word pairs are rated on a scale from 0 (no similarity) to 4 (perfect synonymy). Miller-Charles' data set is a subset of Rubenstein-Goodenough's [130] original data set of 65 word pairs. Although Miller-Charles experiment was carried out 25 years later than Rubenstein-Goodenough's, two sets of ratings are highly correlated (pearson correlation coefficient=0.97). Therefore, Miller-Charles ratings is considered as a reliable benchmark for evaluating semantic similarity measures.

2.3.2 Pattern Selection

I trained a linear kernel SVM with top N pattern features (ranked according to their χ^2 values) and calculated the correlation coefficient against the Miller-Charles benchmark dataset. Results of the experiment are shown in Figure 2.5. In Figure 2.5 a steep improvement of correlation with the number of top-ranking patterns is apparent; it reaches a maximum at 200 features. With more than 200 patterns correlation drops below this

[5] Because of the omission of two word pairs in earlier versions of WordNet, most researchers had used only 28 pairs for evaluations.

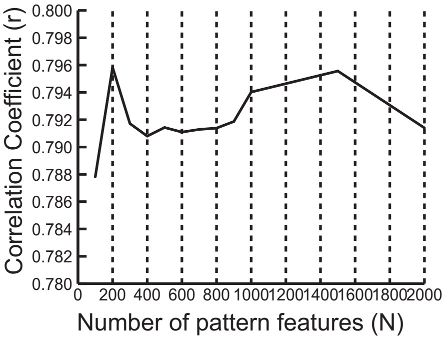


Figure 2.5: Correlation vs. No of pattern features

Table 2.2: Features with the highest SVM linear kernel weights

feature	χ^2	SVM weight
WebDice	N/A	8.19
X/Y	33459	7.53
X, Y :	4089	6.00
X or Y	3574	5.83
X Y for	1089	4.49
X . the Y	1784	2.99
with X (Y	1819	2.85
X=Y	2215	2.74
X and Y are	1343	2.67
X of Y	2472	2.56

maximum. Considering that the patterns are ranked according to their ability to express semantic similarity and the majority of patterns are sparse, I selected only the top ranking 200 patterns for the remaining experiments.

Features with the highest linear kernel weights are shown in Table 2.2 alongside their χ^2 values. The weight of a feature in the linear kernel can be considered as a rough estimate of the influence it imparts on the final SVM output. WebDice has the highest kernel weight followed by a series of pattern-based features. WebOverlap (rank=18, weight=2.45), WebJaccard (rank=66, weight=0.618) and WebPMI (rank=138, weight=0.0001) are not shown in Table 2.2 because of space limitations. It is noteworthy that the pattern features in Table 2.2 agree with intuition. Lexical patterns (e.g., *X or Y, X and Y are, X of Y*) as well as syntax patterns (e.g., bracketing, comma usage) are extracted by my method.

Table 2.3: Performance with different Kernels

Kernel Type	Correlation
Linear	0.8345
Polynomial degree=2	0.5872
Polynomial degree=3	0.5387
RBF	0.6632
Sigmoid	0.5277

I experimented with different kernel types as shown in Table 2.3. Best performance is achieved with the linear kernel. When higher degree kernels such as quadratic (Polynomial degree=2) and cubic (Polynomial degree=3) are used, correlation with the human ratings decreases rapidly. Second best is the Radial Basis Functions (RBFs), which reports a correlation coefficient of 0.6632. For the rest of the experiments in this chapter I use the linear kernel.

2.3.3 Semantic Similarity

I score the word pairs in Miller-Charles' dataset using the page-count-based similarity scores defined in section 2.2.2, Web-based semantic similarity measures proposed in previous work (Sahami [131], Chen [26]), Normalized Google Distance (NGD) [29], and the proposed method (Proposed). Results are shown in Table 2.4. Because NGD is a distance measure and all other measures compared in Table 2.4 are similarity scores, for consistency with other measures, I consider the value after deducting NGD from one in the NGD column in Table 2.4. All figures, except those for the Miller-Charles ratings, are normalized into values in $[0, 1]$ range for the ease of comparison. Pearson's correlation coefficient is invariant against a linear transformation. Proposed method earns the highest correlation of 0.834 in my experiments. It shows the highest similarity score for the word-pair *magician* and *wizard*. Lowest similarity is reported for *cord* and *smile* I did not use any of the words in the benchmark dataset or their synsets for training. My reimplementation of Co-occurrence Double Checking (CODC) measure [26] indicates the second-best correlation of 0.6936. CODC measure reports zero similarity scores for many word-pairs in the benchmark. One reason for this sparsity in CODC measure is that even though two words in a

Table 2.4: Semantic Similarity of Human Ratings and Baselines on Miller-Charles' dataset

Word Pair	MC	Web Jaccard	Web Dice	Web Overlap	Web PMI	NGD	Sahami [131]	CODC [26]	Proposed
automobile-car	3.920	0.650	0.664	0.831	0.427	0.466	0.225	0.008	0.980
journey-voyage	3.840	0.408	0.424	0.164	0.468	0.556	0.121	0.005	0.996
gem-jewel	3.840	0.287	0.300	0.075	0.688	0.566	0.052	0.012	0.686
boy-lad	3.760	0.177	0.186	0.593	0.632	0.456	0.109	0.000	0.974
coast-shore	3.700	0.783	0.794	0.510	0.561	0.603	0.089	0.006	0.945
asylum-madhouse	3.610	0.013	0.014	0.082	0.813	0.782	0.052	0.000	0.773
magician-wizard	3.500	0.287	0.301	0.370	0.863	0.572	0.057	0.008	1.000
midday-noon	3.420	0.096	0.101	0.116	0.586	0.687	0.069	0.010	0.819
furnace-stove	3.110	0.395	0.410	0.099	1.000	0.638	0.074	0.011	0.889
food-fruit	3.080	0.751	0.763	1.000	0.449	0.616	0.045	0.004	0.998
bird-cock	3.050	0.143	0.151	0.144	0.428	0.562	0.018	0.006	0.593
bird-crane	2.970	0.227	0.238	0.209	0.516	0.563	0.055	0.000	0.879
implement-tool	2.950	1.000	1.000	0.507	0.297	0.750	0.098	0.005	0.684
brother-monk	2.820	0.253	0.265	0.326	0.623	0.495	0.064	0.007	0.377
crane-implement	1.680	0.061	0.065	0.100	0.194	0.559	0.039	0.000	0.133
brother-lad	1.660	0.179	0.189	0.356	0.645	0.505	0.058	0.005	0.344
car-journey	1.160	0.438	0.454	0.365	0.205	0.410	0.047	0.004	0.286
monk-oracle	1.100	0.004	0.005	0.002	0.000	0.579	0.015	0.000	0.328
food-rooster	0.890	0.001	0.001	0.412	0.207	0.568	0.022	0.000	0.060
coast-hill	0.870	0.963	0.965	0.263	0.350	0.669	0.070	0.000	0.874
forest-graveyard	0.840	0.057	0.061	0.230	0.495	0.612	0.006	0.000	0.547
monk-slave	0.550	0.172	0.181	0.047	0.611	0.698	0.026	0.000	0.375
coast-forest	0.420	0.861	0.869	0.295	0.417	0.545	0.060	0.000	0.405
lad-wizard	0.420	0.062	0.065	0.050	0.426	0.657	0.038	0.000	0.220
cord-smile	0.130	0.092	0.097	0.015	0.208	0.460	0.025	0.000	0
glass-magician	0.110	0.107	0.113	0.396	0.598	0.488	0.037	0.000	0.180
rooster-voyage	0.080	0.000	0.000	0.000	0.228	0.487	0.049	0.000	0.017
noon-string	0.080	0.116	0.123	0.040	0.102	0.488	0.024	0.000	0.018
Correlation	**1.000**	**0.260**	**0.267**	**0.382**	**0.549**	**0.205**	**0.580**	**0.694**	**0.834**

pair (P, Q) are semantically similar, we might not always find Q among the top snippets for P (and vice versa). As might be apparent from the definition of the CODC measure in Equation 2.2, it returns zero under these conditions. Ranking of snippets, (hence the value of $f(P@Q)$), depends directly upon the search engine's specifications. A search engine considers various factors such as novelty, authority, link structure, user preferences when ranking search results. Consequently, CODC measure is influenced by these factors.

Similarity measure proposed by Sahami et al. [131] is placed third, reflecting a correlation of 0.5797. This method use only those snippets when calculating semantic similarity. Among the four page-counts-based measures, WebPMI garners the highest correlation ($r = 0.5489$). NGD considers only page-counts and it reports a low correlation (0.205) with Miller-Charles' ratings. Overall, the results in Table 2.4 suggest that similarity measures based on snippets are more accurate than the ones based on page counts in capturing semantic similarity between words.

2.3.4 Taxonomy-Based Methods

Table 2.5: Comparison with taxonomy-based methods

Method	Correlation
Human replication	0.9015
Resnik (1995)	0.7450
Lin (1998)	0.8224
Li et al. (2003)	0.8914
Edge-counting	0.664
Information content	0.745
Jiang & Conrath (1998)	0.8484
Proposed	0.8129

Table 2.5 presents a comparison of the proposed method to the WordNet-based methods. The proposed method outperforms simple WordNet-based approaches such as Edge-counting and Information Content measures. It is comparable with Lin (1998) [87] Jiang & Conrath (1998) [69] and Li (2003) [160] methods. Unlike the WordNet based methods,

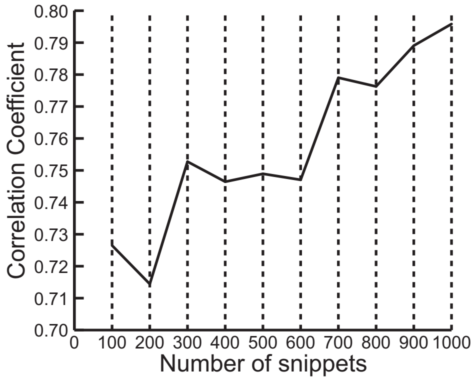


Figure 2.6: Correlation vs. No of snippets

proposed method does not require a hierarchical taxonomy of concepts or sense-tagged definitions of words. Therefore, in principle the proposed method could be used to calculate semantic similarity between named entities, etc, which are not listed in WordNet or other manually compiled thesauri. However, considering the high correlation between human subjects (0.9), there is still room for improvement.

2.3.5 Accuracy vs. Number of Snippets

I computed the correlation with the Miller-Charles ratings for different numbers of snippets to investigate the effect of the number of snippets used to extract patterns upon the semantic similarity measure. I started with 100 snippets and increased this number by 100 snippets at a time. Because of the constraints imposed by Google on the maximum number of snippets that can be collected for a query, we are limited to a maximum of 1000 snippets. The experimental results are presented in Figure 2.6. From Figure 2.6 it is apparent that overall the correlation coefficient improves with the number of snippets used for extracting patterns. The probability of finding better patterns increases with the number of processed snippets. That fact enables us to represent each pair of words with a rich feature vector, resulting in better performance.

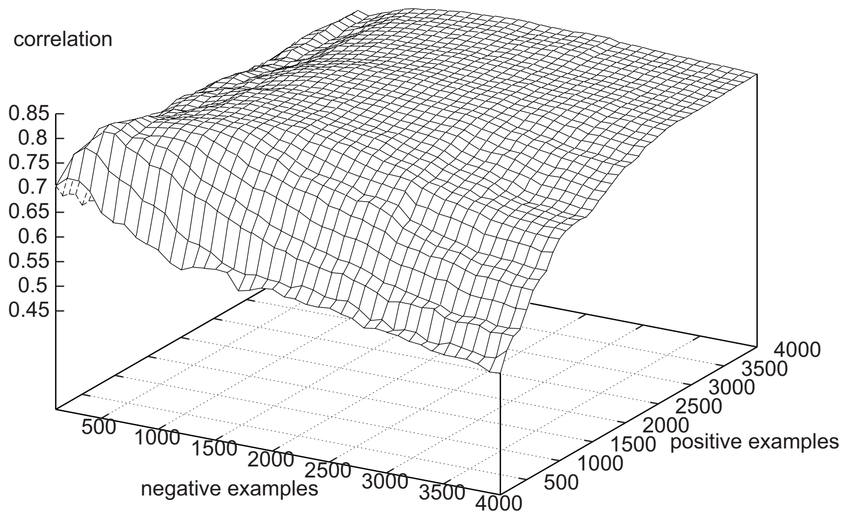


Figure 2.7: Correlation vs. No of positive and negative training instances

2.3.6 Training Data

I used synonymous word pairs extracted from WordNet synsets as positive training examples and automatically generated non-synonymous word pairs as negative training examples to train a two-class support vector machine in section 2.2.4. To determine the optimum combination of positive and negative training examples, I trained a linear kernel SVM with different combinations of positive and negative training examples and evaluated accuracy against the human ratings in the Miller-Charles benchmark dataset. Experimental results are summarized in Figure 2.7. Maximum correlation coefficient of 0.8345 is achieved with 1900 positive training examples and 2400 negative training examples. Generally, in English language the number of non-synonymous word pairs is much greater than the number of synonymous word pairs. I infer that the bias shown toward negative training examples in Figure 2.7 implies that fact. Moreover, Figure 2.7 reveals that correlation does not improve beyond 2500 positive and negative training examples. Therefore, I can conclude that 2500 examples are sufficient to leverage the proposed semantic similarity measure.

Table 2.6: Effect of page-counts and snippets on the proposed method.

Dataset	WordSimilarity-353 [46]	Miller-Charles [103]
page-counts only	0.3571	0.5239
snippets-only	0.4583	0.8078
both	0.8581	0.5183

2.3.7 Page-counts vs. Snippets

The proposed semantic similarity measure integrates two types of features: word association measures computed using page-counts, and frequencies of numerous lexical patterns in snippets. A obvious question that arises in such a hybrid approach is the contribution of page-counts and snippets towards the overall performance of the proposed algorithm. I experimentally evaluate the effect of page-counts and snippets as follows. I compute feature vectors using only the page-counts and train a linear kernel SVM (parameter C is set to 10) and use this model to compute the semantic similarity between word pairs in two benchmark datasets: Miller Charles dataset [103], and WordSimilarity-353 [46] dataset. Only four features will be computed when page-counts are used. Namely, the four features are WebJaccard, WebDice, WebOverlap, and WebPMI. Likewise, I compute feature vectors representing word pairs only using lexical pattern frequencies in snippets. Because we selected 200 lexical patterns as described in Section 2.3.2, we can compute 200 features only using snippets. A linear kernel SVM (parameter C is set to 10) is then trained using the labelled feature vectors.

As the name implies, WordSimilarity-353 dataset contains human ratings for 353 word pairs. This dataset is used in previous work on semantic similarity measures as an evaluation dataset. Compared to Miller-Charles dataset which has only 30 (ca. only 28 used by most previous work) pairs of words, WordSimilarity-353 dataset has a comparatively a large (ca. 353) number of word pairs. Therefore, statistically more reliable evaluations can be performed using the WordSimilarity-353 dataset. However, it is noteworthy that the definition of semantic similarity in the WordSimilarity-353 dataset is more relaxed compared to the Miller-Charles dataset. Consequently, we can find many word pairs that are distinctively related in the WordSimilarity-353 dataset.

Experimental results are shown in Table 2.6. From Table 2.6 we can see that compared to page-counts, snippets have a higher impact on the performance of the proposed method. Moreover, the combination of page-counts-based features and snippets-based features improves the performance of the proposed method when compared using page-counts or snippets alone. We already observed in Table 2.4 that similarity measures computed using snippets have higher correlation coefficients than similarity measures that are limited to page-counts. This is because snippets-based measures can use the local context of the two words between which we must measure semantic similarity. Access to the local context enables snippet-based similarity measures to disambiguate the different senses of the words and match informative lexical patterns, that cannot be done using page-counts. Experimental results in Table 2.6 further emphasizes this point. From Table 2.6 and Table 2.4 collectively, we can conclude that neither individual page-counts-based similarity measures, nor a combinations of those measures can outperform a similariy measure that is defined purely on snippets-based features.

2.3.8 Community Mining

Measuring semantic similarity between named entities is vital in many applications such as query expansion [131], entity disambiguation (e.g. namesake disambiguation) and community mining [96]. Because most named entities are not covered by WordNet, similarity measures that are based on WordNet cannot be used directly in these tasks. Unlike common English words, named entities are being created constantly. Manually maintaining an up-to-date taxonomy of named entities is costly, if not impossible. The proposed semantic similarity measure is appealing for these applications because it does not require pre-compiled taxonomies.

In order to evaluate the performance of the proposed measure in capturing the semantic similarity between named-entities, I set up a community mining task. I select 50 personal names from 5 communities: *tennis players*, *golfers*, *actors*, *politicians* and *scientists* , (10 names from each community) from the open directory project (DMOZ)[6]. For each pair of names in my dataset, I measure their similarity using the proposed method and baselines.

[6] http://dmoz.org

I use group-average agglomerative hierarchical clustering (GAAC) to cluster the names in my dataset into five clusters.

Initially, each name is assigned to a separate cluster. In subsequent iterations, group average agglomerative clustering process, merges the two clusters with highest correlation. Correlation, $\mathrm{Corr}(\Gamma)$ between two clusters A and B is defined as the following,

$$\mathrm{Corr}(\Gamma) = \frac{1}{2}\frac{1}{|\Gamma|(|\Gamma|-1)} \sum_{(u,v)\in\Gamma} \mathrm{sim}(u,v) \tag{2.10}$$

Here, Γ is the merger of the two clusters A and B. $|\Gamma|$ denotes the number of elements (persons) in Γ and $\mathrm{sim}(u,v)$ is the semantic similarity between two persons u and v in Γ. I terminate GAAC process when exactly five clusters are formed. I adopt this clustering method with different semantic similarity measures $\mathrm{sim}(u,v)$ to compare their accuracy in clustering people who belong to the same community.

I employed the ***B-CUBED*** metric [6] to evaluate the clustering results. The B-CUBED evaluation metric was originally proposed for evaluating cross-document co-reference chains. I compute precision, recall and F-score for each name in the data set and average the results over the dataset. For each person p in the data set, let us denote the cluster that p belongs to by $C(p)$. Moreover, I use $A(p)$ to denote the affiliation of person p, e.g., A("Tiger Woods") ="Tennis Player". Then I calculate precision and recall for person p as,

$$\mathrm{Precision}(p) = \frac{\text{No. of people in C(p) with affiliation A(p)}}{\text{No. of people in C(p)}}, \tag{2.11}$$

$$\mathrm{Recall}(p) = \frac{\text{No. of people in C(p) with affiliation A(p)}}{\text{Total No. of people with affiliation A(p)}}. \tag{2.12}$$

Since, I selected 10 people from each of the five categories, the denominator in Formula 2.12 is 10 for all the names p.

Then, the F-score of person p is defined as,

$$\mathrm{F}(p) = \frac{2 \times \mathrm{Precision}(p) \times \mathrm{Recall}(p)}{\mathrm{Precision}(p) + \mathrm{Recall}(p)}. \tag{2.13}$$

Overall precision, recall and F-score are computed by taking the averaged sum over all

Table 2.7: Results for Community Mining

Method	Precision	Recall	F Measure
WebJaccard	0.5926	0.712	0.6147
WebOverlap	0.5976	0.68	0.5965
WebDice	0.5895	0.716	0.6179
WebPMI	0.2649	0.428	0.2916
Sahami [131]	0.6384	0.668	0.6426
Chen [26]	0.4763	0.624	0.4984
Proposed	0.7958	0.804	0.7897

the names in the dataset.

$$\text{Precision} = \frac{1}{N}\sum_{p \in DataSet} \text{Precision}(p) \tag{2.14}$$

$$\text{Recall} = \frac{1}{N}\sum_{p \in DataSet} \text{Recall}(p) \tag{2.15}$$

$$F\text{-Score} = \frac{1}{N}\sum_{p \in DataSet} \text{F}(p) \tag{2.16}$$

Here, $DataSet$ is the set of 50 names selected from the open directory project. Therefore, $N = 50$ in my evaluations.

Experimental results are shown in Table 2.7. The proposed method shows the highest entity clustering accuracy in Table 2.7 with a statistically significant ($p \leq 0.01$ Tukey HSD) F score of 0.7897. Sahami et al. [131]'s snippet-based similarity measure, WebJaccard, WebDice and WebOverlap measures yield similar clustering accuracies.

2.3.9 Entity Disambiguation

Disambiguating named entities is important in various applications such as information retrieval [35], social network extraction [96, 10, 16], Word Sense Disambiguation (WSD) [98], citation matching [58] and cross-document co-reference resolution [120, 47].

For example, *Jaguar* is a cat, a car brand and also an operating system for computers. A user who searches for *Jaguar* on the Web, may be interested in either one of these different senses of Jaguar. However, only the first sense (Jaguar as a cat) is listed in WordNet. Considering the number of new senses constantly being associated to the existing words on the Web, it is costly, if not impossible to maintain sense tagged dictionaries to cover all senses.

Contextual Hypothesis for Sense [135] states that the context in which a word appears can be used to determine its sense. For example, a Web page discussing Jaguar as a car, is likely to talk about other types of cars, parts of cars etc. Whereas, a Web page on Jaguar the cat, is likely to contain information about other types of cats and animals. In this section, I utilize the clustering algorithm described in section 2.3.8 to cluster the top 1000 snippets returned by Google for two ambiguous entities *Jaguar* and *Java*. I represent each snippet as a bag-of-words and calculate the similarity $\mathrm{SIM}(S_a, S_b)$ between two snippets S_a,$S_b$as follows,

$$\mathrm{SIM}(S_a, S_b) = \frac{1}{|S_a||S_b|} \sum_{a \in S_a, b \in S_b} sim(a, b) \tag{2.17}$$

In Formula 2.17 $|S|$ denotes the number of words in snippet S. I used different semantic similarity measures for sim in Formula 2.17 and employed the group average agglomerative clustering explained in section 2.3.8. I manually analyzed the snippets for queries *Java* (3 senses: programming language, Island, coffee) and *Jaguar* (3 senses: cat, car, operating system) and computed precision, recall and F-score for the clusters created by the algorithm. My experimental results are summarized in Table 2.8. Proposed method reports the best results among all the baselines compared in Table 2.8.

In this chapter I proposed a measure that uses both page counts and snippets to robustly calculate semantic similarity between two given words or named entities. The method consists of four page-count-based similarity scores and automatically extracted lexico-syntactic patterns. I integrated page-counts-based similarity scores with lexico syntactic patterns using support vector machines. Training data were automatically generated using WordNet synsets. Proposed method outperformed all the baselines including previously proposed Web-based semantic similarity measures on a benchmark dataset. A high correlation (correlation coefficient of 0.834) with human ratings was found for semantic similarity on this benchmark dataset. Only 1900 positive examples and 2400 negative examples

Table 2.8: Entity Disambiguation Results

	Jaguar			Java		
Method	Precision	Recall	F	Precision	Recall	F
WebJaccard	0.5613	0.541	0.5288	0.5738	0.5564	0.5243
WebOverlap	0.6463	0.6314	0.6201	0.6228	0.5895	0.56
WebDice	0.5613	0.541	0.5288	0.5738	0.5564	0.5243
WebPMI	0.5607	0.478	0.5026	0.7747	0.595	0.6468
Sahami [131]	0.6061	0.6337	0.6019	0.751	0.4793	0.5761
CODC [26]	0.5312	0.6159	0.5452	0.7744	0.5895	0.6358
Proposed	0.6892	0.7144	0.672	0.8198	0.6446	0.691

are necessary to leverage the proposed method, which is efficient and scalable because it only processes the snippets (no downloading of Web pages is necessary) for the top ranking results by Google. A contrasting feature of my method compared to the WordNet based semantic similarity measures is that my method requires no taxonomies, such as WordNet, for calculation of similarity. Therefore, the proposed method can be applied in many tasks where such taxonomies do not exist or are not up-to-date. In Chapter 3 I investigate the problem of measuring relational similarity between word pairs. We revisit the problem of measuring semantic similarity between words from a relational view-point in Chapter 7.

Chapter 3

Relational Similarity

In Chapter 1, I categorized similarity measures into two types: *attributional* similarity measures and *relational* similarity measures. For attributional similarity measures, the objective is to compute the similarity between two given words by comparing the attributes of each word. For example, the two words *car* and *automobile* share many attributes (e.g. *has wheels*, *is used for transportation*). Consequently, they are considered as synonyms. In Chapter 2, I showed how to compute attributional similarity using web search engines. On the other hand, relational similarity is the correspondence between semantic relations that exist between two *word pairs*. Word pairs that show a high degree of relational similarity are considered as *analogies*. For example, the two word pairs *(ostrich, bird)* and *(lion, cat)*. Ostrich *is a large* bird and lion *is a large* cat are illustrative of high relational similarity. The semantic relation, *is a large*, pertains between the two words in each word-pair. In this Chapter I study the problem of measuring relational similarity between word pairs.

The information available on the Web can be considered as a vast, hidden network of classes of objects (e.g. named entities) that is interconnected by various semantic relations applying to those objects. Measuring the similarity between semantic relations is an important intermediate step in various tasks in information retrieval and natural language processing such as relation extraction [32, 33, 161], in which the goal is to retrieve instances of a given relation. For example, given the relation, ACQUIRER-ACQUIREE, a relation extraction system must extract the instance (*Google*, *YouTube*) from the sentence *Google completed the acquisition of YouTube*. Bootstrapping methods [113, 23, 43], which require a few seeds (ca. 10 pairs of instances per relation) have extracted numerous candidate instance pairs from a text corpus. Given a set of candidate instance pairs, a relational similarity measure can be used to compute the similarity between the relations in the seeds and in the candidates. Candidate instance pairs with high relational similarity with the seed pairs can then be selected as the correct instances of a relation.

Relational similarity measures have been used to find word analogies [37, 107, 148, 150, 157]. Word analogy questions have been used from the Scholastic Aptitude Test (SAT; Educational Testing Service) to benchmark relational similarity measures. An SAT word analogy question consists of a stem word-pair that acts as the question and five choice word pairs, out of which only one is analogous to the stem. A relational similarity measure is used to compare the stem word-pair with each choice word-pair and to select the choice word-pair with the highest relational similarity as the answer.

An interesting application of relational similarity in information retrieval is to search using implicitly stated analogies [94, 156]. For example, the query "Muslim Church" is expected to return "mosque", and the query "Hindu bible" is expected to return "the Vedas". These queries can be formalized as word pairs: (Christian, Church) vs. (Muslim,X), and (Christian, Bible) vs. (Hindu,Y). We can then find the words X and Y that maximize the relational similarity in each case.

Despite the wide applications of relational similarity measures, accurately measuring the similarity between implicitly stated relations remains a challenging task for several reasons. First, relational similarity is a dynamic phenomenon: it varies with time. For example, two companies can be competitors initially; subsequently one company might acquire the other. Second, there can be more than one relation between a given word-pair. For example, between the two words *ostrich* and *bird*, aside from the relation *is a large*, there is also the relation *is a flightless*. A relational similarity measure must first extract all relations between the two words in each word-pair before it can compute the similarity between the word pairs. Third, there can be more than one way to express a particular semantic relation in a text. For example, the three patterns – ***X** was acquired by **Y***, ***Y** completed the acquisition of **X***, and ***Y** buys **X*** – all indicate an acquisition relation between **X** and **Y**. In addition to the problems described above, measuring relational similarity between pairs in which one or both words are named entities (e.g., company names, personal names, locations, etc.) is even more difficult because such words are not well covered by manually created dictionaries such as WordNet[1] [104].

As described herein, I propose a relational similarity measure that uses a Web search engine to measure the similarity between implicitly stated semantic relations in two word pairs. Formally, given two word pairs, *(a,b)* and *(c,d)*, I design a function, *relsim*$((a, b), (c, d))$, that returns a similarity score in the range $[0, 1]$. Two different approaches are considered in this thesis. First, a supervised learning approach using Support Vector Machines is introduced in Section 3.2. Second, a supervised Mahalanobis distance metric approach is proposed in Section 3.3. Before I detail each of these methods I briefly review the previous work on measuring relational similarity.

3.1 Previous Work on Relational Similarity

The Structure Mapping Theory (SMT) [45] is based on the premise that an analogy is a mapping of knowledge from one domain (base) into another (target), which conveys that a system of relations known to hold in the base also holds in the target. The target objects need not resemble their corresponding base objects. This structural view of analogy is based on the intuition that analogies are about relations, rather than simple features. Although this approach works best when the base and the target are rich in higher-order causal structures, it can fail when structures are missing or flat [158].

Turney et al. [152] combined 13 independent modules by considering the weighted sum of the outputs of each module to solve SAT analogy questions. The best performing individual module was based on the Vector Space Model (VSM). In the VSM approach [151], a vector is first created for a word-pair *(X,Y)* by counting the frequencies of various lexical patterns containing *X* and *Y*. In their experiments, they used 128 manually created patterns such as "*X* of *Y*", "*Y* of *X*", "*X* to *Y*", and "*Y* to *X*". These patterns are then used as queries to a search engine. The numbers of hits for respective queries are used as elements in a vector to represent the word-pair. Finally, the relational similarity is computed as the cosine of the angle between the two vectors that represent the two word pairs. Turney et al. [152] introduced a dataset containing 374 SAT analogy questions to evaluate relational similarity

[1] http://wordnet.princeton.edu/

measures. An SAT analogy question consists of a stem word-pair that acts as the question, and five choice word pairs. The choice word-pair that has the highest relational similarity with the stem word-pair is selected by the system as the correct answer. The average SAT score reported by high school students for word-analogy questions is 57%. The VSM approach achieves a score of 47% on this dataset.

Turney [148, 150] proposed Latent Relational Analysis (LRA) by extending the VSM approach in three ways: a) lexical patterns are automatically extracted from a corpus, b) the Singular Value Decomposition (SVD) is used to smooth the frequency data, and c) synonyms are used to explore variants of the word pairs. Similarly, in the VSM approach, LRA represents a word-pair as a vector of lexical pattern frequencies. First, using a thesaurus, he finds related words for the two words in a word-pair and create additional word pairs that are related to the original word pairs in the dataset. Second, n-grams of words are extracted from the contexts in which the two words in a word-pair cooccur. The most frequent n-grams are selected as lexical patterns to represent a word-pair. Then a matrix of word pairs vs. lexical patterns is created for all the word pairs in the original dataset and the additional word pairs. Elements of this matrix correspond to the frequency of a word-pair in a lexical pattern. Singular value decomposition is performed on this matrix to reduce the number of columns (i.e. patterns). Finally, the relational similarity between two word pairs is computed as the average cosine similarity over the original word pairs and the additional word pairs derived from them. In fact, LRA achieves a score of 56.4% on SAT analogy questions.

Both VSM and LRA require numerous search engine queries to create a vector to represent a word-pair. For example, with 128 patterns, the VSM approach requires at least 256 queries to create two pattern-frequency vectors for two word pairs before it can compute the relational similarity. In fact, LRA considers synonymous variants of the given word pairs. For that reason, it requires even more search engine queries. Methods that require numerous queries impose a heavy load on search engines. Despite efficient implementations, singular value decomposition of large matrices is time consuming. In fact, LRA takes over 9 days to process the 374 SAT analogy questions [150]. This is problematic when computing relational similarity on the scale of the Web. Moreover, in the case of named entities, thesauri of related words are not usually available or are not complete, which becomes a problem when creating the additional word pairs required by LRA.

Veale [157] proposed a relational similarity measure based on the taxonomic similarity in WordNet. The quality of a candidate analogy *A:B::C:D* (i.e. *A* to *B* as *C* to *D*) is evaluated through comparison of the paths in the WordNet, joining *A* to *B* and *C* to *D*. Relational similarity is defined as the similarity between the *A:B* paths and *C:D* paths. However, WordNet does not fully cover named entities such as personal names, organizations and locations, which becomes problematic when using this method to measure relational similarity between named entities.

Using a relational similarity measure, Turney [149] proposed an unsupervised learning algorithm to extract patterns that express implicit semantic relations from a corpus. His method produces a ranked set of lexical patterns that unambiguously describes the relation between the two words in a given word-pair. Patterns are ranked according to their expected relational similarity (i.e. pertinence); they are computed using an algorithm similar to LRA. To answer an SAT analogy question, first, ranked lists of patterns are generated for each of the six word pairs (one stem word-pair and five choice word pairs). Then each choice is evaluated by taking the intersection of its patterns with the stem's patterns. The shared patterns are scored by the average of their rank in the stem's list and the choice's lists. The algorithm picks the choice with the lowest scoring shared pattern as the correct answer. This method reports an SAT score of 54.6%.

Relational similarity measures have been applied in natural language processing tasks such as generating word analogies [37], and classifying noun-modifier compounds based on the relation between the head and the modifier [150, 107, 36, 107]. Davidov and Rappoport [37] proposed an unsupervised algorithm to discover general semantic relations that pertain between lexical items. They represent a semantic relation with a cluster of patterns. They use the pattern clusters to generate SAT-like word analogy questions for English and Russian languages. The generated questions are then solved by human subjects. They do not evaluate their method for relational similarity between named entities.

Relational similarity measures have been used to classify the relationships between the head and the modifier in noun-compounds [150, 107, 36]. For example, in the compound *viral flu*, the *flu* (head) is *caused by* a *virus* (modifier). The *Diverse* dataset of Barker and Szpakowicz [9], which consists of 600 head-modifier pairs (noun-noun, adjective-noun and

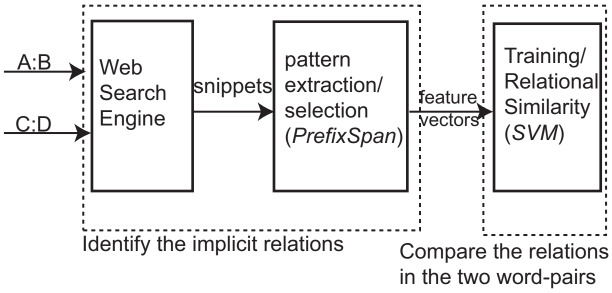

A:B
C:D
Web Search Engine
snippets
pattern extraction/ selection (PrefixSpan)
feature vectors
Training/ Relational Similarity (SVM)
Identify the implicit relations
Compare the relations in the two word-pairs

Figure 3.1: Supervised learning of relational similarity using SVMs.

adverb-noun) is used as a benchmark dataset to evaluate relation classification of noun-compounds. Each noun-modifier pair in this dataset is annotated with one of the following five relations: *causal*, *temporal*, *spatial*, *participant*, and *quality*. Nakov and Hearst [107] proposed a linguistically motivated method that utilizes verbs, prepositions, and coordinate conjunctions that can help make explicit the hidden relations between the target nouns. They report a classification accuracy of 40.5% on the Diverse dataset using a single nearest neighbor classifier.

3.2 Method I: Supervised Learning using Support Vector Machines

The proposed supervised relational similarity measure is outlined in Figure reffig:outline. It can be described in two main steps: *identifying the implicit relations* between the two words in each word-pair and *comparing the relations* that exist in each word-pair. In order to measure the relational similarity between two word-pairs *A:B* and *C:D*, we must first identify the relations implied by each word-pair. For example, the relation *X is-a-large Y* holds between the the two words in pairs *ostrich:bird* and *lion:cat*. I propose the use of *PrefixSpan* [119], a sequential pattern mining algorithm, to extract implicit relations from snippets returned by a web search engine for two words. I train a Support Vector Machine (SVM) [155] using SAT multiple-choice analogy questions as training data to compare the extracted relations and identify analogous word-pairs. Next I describe each of these steps

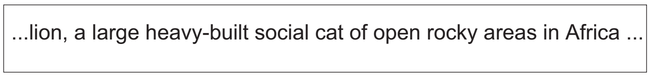

...lion, a large heavy-built social cat of open rocky areas in Africa ...

Figure 3.2: A snippet returned by Google for the query "lion*******cat".

in detail.

3.2.1 Pattern Extraction and Selection

I represent the implicit relations indicated by the two words in a word-pair *X:Y* using automatically extracted lexical patterns. Although automatic pattern extraction methods [124, 139] have been proposed based on dependency parsing of sentences, extracting lexical patterns from snippets using such methods is difficult because most snippets are not grammatically correct complete sentences. However, lexical syntactic patterns have been successfully used to extract semantic information such as qualia structures [32] from web text snippets. Consequently, I employ a shallow pattern extraction method based on sequential pattern mining.

To identify the implicit relations between two words *X* and *Y*, I first query a web search engine using the phrasal query *"X*******Y"*. Here, the wildcard operator "*" would match any word or nothing. This query retrieves snippets that contain both *X* and *Y* within a window of 7 words. For example, Google returns the snippet shown in Figure 3.2 for the word-pair *lion:cat*. I use *PrefixSpan* (i.e., prefix-projected sequential pattern mining) [119] algorithm to extract frequent subsequences from snippets that contain both *X* and *Y*. PrefixSpan extracts all word subsequences which occur more than a specified frequency in snippets. I select subsequences that contain both query words (eg. *lion* and *cat*) and replace the query words respectively with variables *X* and *Y* to construct lexical patterns. For example, some of patterns I extract from the snippet in Figure 3.2 are *"X a large Y"*, *"X a large Y of"* and *"X, a large social Y"*. PrefixSpan algorithm is particularly suitable for the current task because it can efficiently extract a large number of lexical patterns.

I used the SAT analogy questions dataset which was first proposed by Turney and Littman [151] as a benchmark to evaluate relational similarity measures, to extract lexical patterns. The dataset contains 2176 unique word-pairs across 374 analogy questions.

Table 3.1: Contingency table for a pattern v

	v	patterns other than v	Total
Freq. in snippets for question and correct answer	p_v	$P - p_v$	P
Freq. in snippets for question and incorrect answer	n_v	$N - n_v$	N

For each word-pair, I searched Google and download the top 1000 snippets. From the patterns extracted by the above mentioned procedure, I select ones that occur more than three times and have less than seven words. The variables X and Y in patterns are swapped to create a reversed version of the pattern. The final set contains 9980 unique patterns. However, out of those patterns only 10% appear in both for a question and one of its choices. It is impossible to learn with such a large number of sparse patterns. Therefore, I perform a pattern selection procedure to identify those patterns that convey useful clues about implicit semantic relations.

First, for each extracted pattern v, I count the number of times where v appeared in any of the snippets for both a question and its correct answer (p_v) and in any of the snippets for both a question and any one of its incorrect answers (n_v). I then create a contingency table for each pattern v, as shown in Table 3.1. In Table 3.1, P denotes the total frequency of all patterns that occur in snippets for a question and its correct answer ($P = \sum_v p_v$) and N is the same for incorrect answers ($N = \sum_v n_v$). If a pattern occurs many times in a question and its correct answer, then such patterns are reliable indicators of latent relations between words. To evaluate the reliability of an extracted pattern as an indicator of a relation, I calculate the χ^2 [93] value for each pattern using Table 3.1 as,

$$\chi^2 = \frac{(P+N)(p_v(N-n_v) - n_v(P-p_v))^2}{PN(p_v+n_v)(P+N-p_v-n_v)}.$$

Patterns with χ^2 value greater than a specified threshold are used as features for training. Some of the selected patterns are shown later in Table 3.3.

3.2.2 Training

For given two pairs of words *A:B* and *C:D*, I create a feature vector using the patterns selected in section 3.2.1. First, I record the frequency of occurrence of each selected pattern in snippets for each word-pair. I call this the *pattern frequency*. It is a local frequency count, analogous to *term frequency* in information retrieval [132]. Secondly, I combine the two pattern frequencies of a pattern (i.e., frequency of occurrence in snippets for *A:B* and that in snippets for *C:D*) using various feature functions to compute the feature-values for training. The different feature functions experimented in this chapter are explained in section 3.2.3.

I model the problem of computing relational similarity as a one of identifying analogous and non-analogous word-pairs, which can be solved by training a binary classifier. Using SAT analogy questions as training data, I train a two-class support vector machine (SVM) as follows. From each question in the dataset, I create a positive training instance by considering *A:B* to be the word-pair for the question and *C:D* to be the word-pair for the correct answer. Likewise, a negative training instance is created from a question word-pair and one of the incorrect answers.

The trained SVM model can then be used to compute the relational similarity between two given word-pairs *A:B* and *C:D* as follows. First, I represent the two word-pairs by a feature vector F of pattern frequency-based features. Second, I define the relational similarity $\mathrm{RelSim}(A : B, C : D)$ between the two word-pairs *A:B* and *C:D* as the posterior probability $\mathrm{Prob}(F|analogous)$ that feature vector F belongs to the analogous-pairs (positive) class,

$$\mathrm{RelSim}(A : B, C : D) = \mathrm{Prob}(F|analogous).$$

Being a large margin classifier, the output of an SVM is the distance from the decision hyper-plane. For the purpose of solving SAT questions, I can directly use the distance from the decision hyper-plane and rank the candidate answers. However, distance from the decision hyper-plane is not a calibrated posterior probability that lies between $[0, 1]$ range. I use sigmoid functions to convert this uncalibrated distance into a calibrated posterior probability (see [121] for a detailed discussion on this topic).

3.2.3 Experiments

For the experiments in this chapter I used the 374 SAT college-level multiple-choice analogy questions dataset which was first proposed by Turney et al. [152]. I compute the total score for answering SAT questions as follows,

$$\mathrm{score} = \frac{\text{no. of correctly answered questions}}{\text{total no. of questions}}. \tag{3.1}$$

Formula 3.1 does not penalize a system for marking incorrect answers.

Feature Functions

Evidence from psychological experiments suggest that similarity can be context-dependent and even asymmetric [154, 100]. Human subjects have reportedly assigned different similarity ratings to word-pairs when the two words were presented in the reverse order. However, experimental results investigating the effects of asymmetry reports that the average difference in ratings for a word-pair is less than 5 percent [100]. Consequently, in this chapter I assume relational similarity to be symmetric and limit our discussion to symmetric feature functions. This assumption is in line with previous work on relational similarity described in section 3.1.

Let us assume the frequency of a pattern v in two word-pairs *A:B* and *C:D* to be f_{AB} and f_{CD}, respectively. I compute the value assigned to the feature corresponding to pattern v in the feature vector that represents the two word-pairs *A:B* and *C:D* using the following four feature functions.

$|f_{AB} - f_{CD}|$**:** The absolute value of the difference of pattern frequencies is considered as the feature-value.

$(f_{AB} - f_{CD})^2$**:** The square of the difference of pattern frequencies is considered as the feature-value.

$f_{AB} \times f_{CD}$**:** The product of the pattern frequencies is considered as the feature-value.

JS divergence: Ideally, if two word-pairs are analogous we would expect to see similar distributions of patterns in each word-pair. Consequently, the *closeness* between the

Table 3.2: Performance with various feature weighting methods

Feature function	Score
$\|f_{AB} - f_{CD}\|$	0.30
$(f_{AB} - f_{CD})^2$	0.30
$f_{AB} \times f_{CD}$	0.40
$JS(v)$	0.32

pattern distributions can be regarded as an indicator of relational similarity. I define a feature function based on Jensen-Shannon divergence [93] as a measure of the closeness between pattern distributions. Jensen-Shannon (JS) divergence $D_{JS}(P||Q)$, between two probability distributions P and Q is given by,

$$D_{JS}(P||Q) = \frac{1}{2}D_{KL}(P||M) + \frac{1}{2}D_{KL}(Q||M). \tag{3.2}$$

Here, $M = (P+Q)/2$ and D_{KL} is the Kullback-Leibler divergence, which is given by,

$$D_{KL}(P||Q) = \sum_{v} P(v) \log \frac{P(v)}{Q(v)}. \tag{3.3}$$

Here, $P(v)$ denotes the normalized pattern frequency of a pattern v in the distribution P. Pattern frequencies are normalized s.t. $\sum_v P(v) = 1$ by dividing the frequency of each pattern by the sum of frequencies of all patterns. I define the contribution of each pattern towards the total JS-divergence in Formula 3.2 as its feature value, $JS(v)$. Substituting Formula 3.3 in 3.2 and collecting the terms under summation, I derive $JS(v)$ as,

$$JS(v) = \frac{1}{2}(p \log \frac{2q}{p+q} + q \log \frac{2p}{p+q}).$$

Here, p and q respectively denote the normalized pattern frequencies of f_{AB} and f_{CD}.

To evaluate the effect of various feature functions on performance, I trained a linear kernel SVM with each of the feature functions. I randomly selected 50 questions from the

Table 3.3: Patterns with the highest SVM linear kernel weights

pattern	χ^2	SVM weight
and Y and X	0.8927	0.0105
Y X small	0.0795	0.0090
X in Y	0.0232	0.0087
use Y to X	0.5059	0.0082
from the Y X	0.3697	0.0079
to that Y X	0.1310	0.0077
or X Y	0.0751	0.0074
X and other Y	1.0675	0.0072
a Y or X	0.0884	0.0068
that Y on X	0.0690	0.0067

Table 3.4: Performance with different Kernels

Kernel Type	Score
Linear	0.40
Polynomial degree=2	0.34
Polynomial degree=3	0.34
RBF	0.36
Sigmoid	0.36

SAT analogy questions for evaluation. The remainder of the questions (374-50) are used for training. Experimental results are summarized in Table 3.2. Out of the four feature functions in Table 3.2, product of pattern frequencies performs best. For the remainder of the experiments in this chapter I used this feature function. Patterns with the highest linear kernel weights are shown in Table 3.3 alongside their χ^2 values. The weight of a feature in the linear kernel can be considered as a rough estimate of the influence it imparts on the final SVM output. Patterns shown in Table 3.3 express various semantic relations that can be observed in SAT analogy questions.

I experimented with different kernel types as shown in Table 3.4. Best performance is achieved with the linear kernel. A drop of performance occurs with more complex kernels, which is attributable to over-fitting. Figure 3.3 plots the variation of SAT score with the number of snippets used for extracting patterns. From Figure 3.3 it is apparent that overall

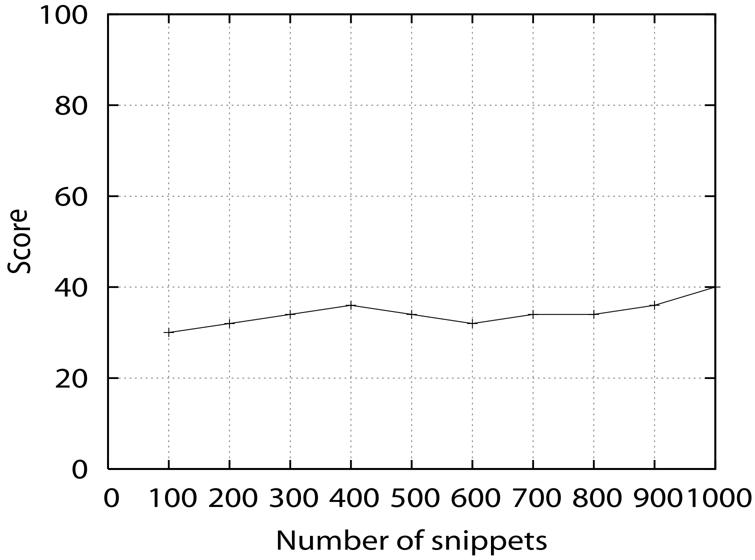


Figure 3.3: Performance with the number of snippets

the score improves with the number of snippets used for extracting patterns. Typically, with more snippets to process, the number of patterns that can be extracted for a word-pair increases. That fact enables us to represent a word-pair with a rich feature vector, resulting in better performance.

Table 3.5 summarizes various relational similarity measures proposed in previous work. All algorithms in Table 3.5 are evaluated on the same SAT analogy questions. Score is computed by Formula 3.1. Because SAT questions contain 5 choices, a random guessing algorithm would obtain a score of 0.2 (lower bound). The score reported by average senior high-school student is about 0.570 [151] (upper bound). I performed 5-fold cross validation on SAT questions to evaluate the performance of the proposed method. The first 13 (rows 1-13) algorithms were proposed by Turney et al. [152], in which they combined these modules using a weight optimization method. For given two word-pairs, the phrase vector (row 1) algorithm creates a vector of manually created pattern-frequencies for each word-pair and compute the cosine of the angle between the vectors. Algorithms in rows 2-11 use WordNet to compute various relational similarity measures based on different semantic relations defined in WordNet. **Similarity:dict** (row 12) and **Similarity:wordsmith** (row 13) respectively use `Dictionary.com` and `Wordsmyth.net` to find the definition of

Table 3.5: Comparison with previous relational similarity measures.

	Algorithm	score		Algorithm	score
1	Phrase Vectors	0.382	11	Holonym:member	0.200
2	Thesaurus Paths	0.250	12	Similarity:dict	0.180
3	Synonym	0.207	13	Similarity:wordsmyth	0.294
4	Antonym	0.240	14	Combined [152]	0.450
5	Hypernym	0.227	15	Proposed (SVM)	0.401
6	Hyponym	0.249	16	WordNet [157]	0.428
7	Meronym:substance	0.200	17	VSM [151]	0.471
8	Meronym:part	0.208	18	Pertinence [149]	0.535
9	Meronym:member	0.200	19	LRA [148]	0.561
10	Holonym:substance	0.200	20	Human	0.570

words in word-pairs and compute the relational similarity as the overlap of words in the definitions. The proposed method outperforms all those 13 individual modules reporting a score of 0.401, which is comparable with Veale's [157] WordNet-based relational similarity measure.

Although LRA (row 19 in Table 3.5) reports the highest SAT score of 0.561 it takes over 8 days to process the 374 SAT analogy questions [150]. On the other hand the proposed method requires less than 6 hours using a desktop computer with a 2.4 GHz Pentium4 processor and 2GB of RAM. In Table 3.6 I compare the proposed method against LRA on runtime. The runtime figures for LRA are obtained from the original paper [150] and I have only shown the components that consume most of the processing time. The gain in speed is mainly attributable to the lesser number of web queries required by the proposed method. To compute the relational similarity between two word-pairs *A:B* and *C:D* using LRA, I first search in a dictionary for synonyms for each word. Then the original words are replaced by their synonyms to create alternative pairs. Each word-pair is represented by a vector of pattern-frequencies using a set of automatically created 4000 lexical patterns. Pattern frequencies are obtained by searching for the pattern in a web search engine. For example, to create a vector for a word-pair with three alternatives, LRA requires 12000 (4000×3) queries. On the other hand, the proposed method first downloads snippets for each word-pair and then searches for patterns only in the downloaded snippets. Because multiple snippets can be retrieved by issuing a single query, the proposed method requires

Table 3.6: Comparison with LRA on runtime

LRA	Hrs:Mins	Hardware
Find alternatives	24 : 56	1 CPU
Filter phrases and patterns	143 : 33	16 CPUs
Generate a sparse matrix	38 : 07	1 CPU
Calculate entropy	0 : 11	1 CPU
Singular value decomposition	0 : 43	1 CPU
Evaluate alternatives	2 : 11	1 CPU
Total	209 : 41	

Proposed	Hrs:Mins	Hardware
Download snippets	2 : 05	1 CPU
Pattern extraction	0 : 05	1 CPU
Pattern selection	2 : 56	1 CPU
Create feature vectors	0 : 46	1 CPU
Training	0 : 03	1 CPU
Testing	0.01	1 CPU
Total	5 : 56	

only one search query to compute a pattern-frequency vector for a word-pair. Processing snippets is also efficient as it obviates the trouble of downloading web pages, which might be time consuming depending on the size of the pages. Moreover, LRA is based on singular value decomposition (SVD), which requires time consuming complex matrix computations.

3.3 Method II: Mahalanobis Distance Metric Learning

In section 3.2, I described a supervised approach to learn relational similarity using support vector machines. Although this supervised method produced encouraging results, it has several limitations. First, although the proposed method extracts a large number of lexical patterns, not all lexical patterns describe unique semantic relations. For example, the two patterns ***X*** *acquired* ***Y*** and ***X*** *bought* ***Y*** both describe an acquisition relation between ***X*** and ***Y***. Second, semantic relations themselves can be dependent. For example, the semantic relations *IS A* and *HAS A* are closely related. I next describe a Mahalanobis distance metric

Google to acquire YouTube for $1.65 billion in stock. Combination will create new opportunities for users and content owners everywhere...

Figure 3.4: A snippet returned for the query *“Google * * * YouTube”*.

learning approach to overcome those limitations. I first present a sequential pattern clustering algorithm to identify the different lexical patterns that describe a particular semantic relation.

3.3.1 Retrieving Contexts

I must first identify the implicitly stated relations that hold between the two words in each word-pair to compute the relational similarity between two given word pairs. The context in which two words cooccur provides useful clues about the semantic relations that pertain between those words. I propose the use of text snippets retrieved using a Web search engine as an approximation of the context of two words. Snippets (also known as *dynamic teasers*) are brief summaries provided by most Web search engines along with the search results. Typically, a snippet contains a window of text selected from a document that includes the queried words. Snippets are useful for search because, most of the time, a user can read the snippet and decide whether a particular search result is relevant, without even opening the url. Using snippets as contexts is also computationally efficient because it obviates the need to download the source documents from the Web, which can be time consuming if a document is large.

A snippet for a query containing two words captures the local context in which they cooccur. For example, consider the snippet shown in Figure 3.4, returned by *Yahoo*[2] for the query *“Google * * YouTube”*. Here, the wildcard operator “*” matches one word or none in a document. The snippet in Figure 3.4 is extracted from an online newspaper article about the acquisition of YouTube by Google.

To retrieve snippets for a word-pair *(A,B)*, I use the following seven types of queries: *“A * B”*, *“B * A”*, *“A * * B”*, *“B * * A”*, *“A * * * B”*, *“B * * * A”*, and *A B*. The queries containing the wildcard operator “*” return snippets in which the two words, *A* and

[2] http://developer.yahoo.com/search/boss/

B appear within a window of specified length. I designate such queries *wildcard* queries. I search for snippets in which the query words cooccur within a maximum window of three words (tokens). This process is intended to approximate the local context of two words in a document. The quotation marks around a query will ensure that the two words appear in the specified order (e.g. *A* before *B* in snippets retrieved for the query *"A * B"*). As a fallback in the case that all wildcard queries fail to return any snippets, I use the query *A B* (without wildcards or quotations) to retrieve snippets where *A* and *B* appear in any order.

Once I collect snippets for a word-pair using the procedure described above, I remove duplicate search results. I consider two snippets to be duplicates if they contain the exact sequence of all words. Duplicate snippets exist mainly for two reasons. First, a web page can be mirrored in more than one location, and the default de-duplication mechanism of the search engine might fail to filter out the duplicates. Second, the queries I construct for a word-pair are not independent. For example, a query with two wildcards might return a snippet that can also be retrieved using a query with one wildcard. However, I observed that the ranking of search results vary with the number of wildcards used. A search engine usually returns only the top ranking results (in the case of Yahoo, only the top 1000 snippets can be downloaded). I use multiple queries per word-pair that induce different rankings, and aggregate search results to circumvent this limitation.

3.3.2 Extracting Lexical Patterns

Lexical syntactic patterns have been used in various natural language processing tasks such as extracting hypernyms [60, 139], or meronyms [12], question answering [124], and paraphrase extraction [13]. Following these previous works, I present a shallow lexical pattern extraction algorithm to represent the semantic relations between two words. The proposed method requires no language-dependent preprocessing such as part-of-speech tagging or dependency parsing, which can be both time consuming at Web scale, and likely to produce incorrect results because of the fragmented and ill-formed snippets. The pattern extraction algorithm consists of the following three steps.

Step 1: Given a context S, retrieved for a word-pair (A, B) according to the procedure described in section 3.3.1, I replace the two words A and B, respectively, with two variables X and Y. Legal abbreviations such as *Inc., Ltd., Corp.*, and titles such as *Mr., Ms., Prof., Dr., Rev.* are considered as occurrences of the query terms. For example, *Google Inc.* is considered as an occurrence of the entity *Google*. I replace all numeric values by D, a marker for digits. Punctuation marks are not removed.

Step 2: I generate all subsequences of the context S that satisfy all of the following conditions.

1. A subsequence must contain exactly one occurrence of each X and Y (i.e., exactly one X and one Y must exist in a subsequence).
2. The maximum length of a subsequence is L words.
3. A subsequence is allowed to have gaps. However, I do not allow gaps of more than g number of words. Moreover, the total length of all gaps in a subsequence should not exceed G words.
4. I expand all negation contractions in a context. For example, *didn't* is expanded to *did not*. I do not skip the word *not* when generating subsequences. For example, this condition ensures that from the snippet *X is not a Y*, I do not produce the subsequence *X is a Y*.

Step 3: I count the frequency of all generated subsequences for all word pairs in the dataset. I select subsequences with frequency greater than N as lexical patterns to represent the semantic relations between words.

My pattern extraction algorithm has four parameters (ca. *L, g, G* and *N*). I set the values of those parameters experimentally, as explained later in section 3.3.5. It is noteworthy that the proposed pattern extraction algorithm considers all the words in a snippet, and is *not* limited to extracting patterns only from the mid-fix (i.e., the portion of text in a snippet that appears between the queried words). Moreover, the consideration of gaps enables us to capture relations between distant words in a snippet. I use a modified version of the *prefixspan* algorithm [119] to generate subsequences. The conditions in Step 2 are used to prune the search space, thereby reducing the number of generated subsequences in prefixspan. For example, some patterns extracted form the snippet shown in Figure 3.4 are: ***X** to acquire **Y***, ***X** acquire **Y***, and ***X** to acquire **Y** for*.

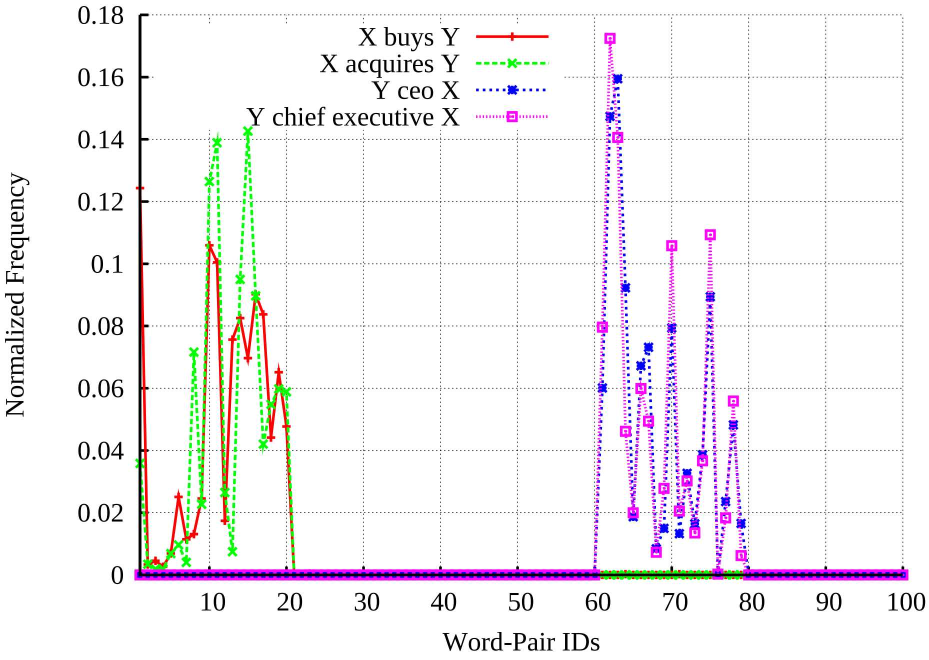


Figure 3.5: Distribution of four lexical patterns in word pairs.

3.3.3 Identifying Semantic Relations

A semantic relation can be expressed using more than one pattern. For example, consider the two distinct patterns, ***X*** *acquired* ***Y***, and ***X*** *completed the acquisition of* ***Y***. Both these patterns indicate that there exists an acquisition relation between ***X*** and ***Y***. It is important to know whether any correspondence pertains between the sets of patterns extracted for each word-pair when I compute the relational similarity between two word pairs. We can expect a high relational similarity if there are many related patterns between two word pairs.

I use the distributional hypothesis [59] to find semantically related lexical patterns. The distributional hypothesis states that words that occur in the same context have similar meanings. The distributional hypothesis has been used in various related tasks, such as identifying related words[85], discovering inference rules[88], and extracting paraphrases[13]. If two lexical patterns are similarly distributed over a set of word pairs (i.e. occurs with the same set of word pairs), then from the distributional hypothesis it follows that the two patterns must be similar. For example, consider the distributions shown in Figure 3.5 for four lexical patterns: *X buys Y*, *X acquires Y*, *Y CEO X*, and *Y chief executive X*, over a set of 100 word pairs. Each distribution is normalized such that the sum of frequencies over all word pairs equals one. Figure 3.5 shows that the distributions of patterns *Y CEO X*, and *Y chief executive X* have a high overlap (i.e., cosine similarity of 0.969). Similarly, the distributions of patterns *X buys Y*, and *X acquires Y* show a high overlap (i.e. cosine similarity of 0.853). However, almost no overlap is apparent between other combinations of distributions. Consequently, to recognize semantically related patterns, I cluster lexical patterns using the similarity of their distributions over word pairs.

I represent a pattern p by a vector $\mathbf{p}$ of word-pair frequencies. I designate $\mathbf{p}$, the *word-pair frequency* vector of pattern p. It is analogous to the *document frequency* vector of a word, as used in information retrieval. The value of the element corresponding to a word-pair (a_i, b_i) in $\mathbf{p}$, is the frequency, $f(a_i, b_i, p)$, that the pattern p occurs with the word-pair (a_i, b_i). As demonstrated later in the experiments of this study, the proposed pattern extraction algorithm typically extracts numerous lexical patterns (more than $140,000$). Clustering algorithms based on pairwise comparisons among all patterns are not feasible when the patterns are numerous. Next, I present a sequential clustering algorithm to efficiently cluster the extracted patterns.

Given a set P of patterns and a clustering similarity threshold θ, Algorithm 3 returns clusters (of patterns) that express similar semantic relations. First, in Algorithm 3, the function $SORT$ sorts the patterns into descending order of their total occurrences in all word pairs. The total occurrence of a pattern p is the sum of frequencies over all word pairs (i.e., $\sum_i f(a_i, b_i, p)$). After sorting, the most common patterns appear at the beginning in P, whereas rare patterns (i.e., patterns that occur with only few word pairs) get shifted to the end. Next, in line 2, I initialize the set of clusters, C, to the empty set. The outer for-loop (starting at line 3), repeatedly takes a pattern $\mathbf{p_i}$ from the ordered set P, and in the inner for-loop (starting at line 6), finds the cluster, c^* ($\in C$) that is most similar to $\mathbf{p_i}$. First, I represent a cluster by the centroid of all word-pair frequency vectors corresponding to the patterns in that cluster to compute the similarity between a pattern and a cluster. Next, I compute the cosine similarity between the cluster centroid ($\mathbf{c_j}$), and the word-pair frequency vector of the pattern ($\mathbf{p_i}$). If the similarity between a pattern $\mathbf{p_i}$, and its most similar cluster, $\mathbf{c}^*$, is greater than the threshold θ, I append $\mathbf{p_i}$ to $\mathbf{c}^*$ (line 14). I use the operator $\oplus$ to denote the vector addition between $\mathbf{c}^*$ and $\mathbf{p_i}$. Then I form a new cluster $\{\mathbf{p_i}\}$ and append it to the set of clusters, C, if $\mathbf{p_i}$ is not similar to any of the existing clusters beyond the threshold θ.

The only parameter in Algorithm 3, the similarity threshold, θ, ranges in $[0, 1]$. It decides the *purity* of the formed clusters. Setting θ to a high value ensures that the patterns in each cluster are highly similar. However, high θ values also yield numerous clusters (increased model complexity). In section 2.3, I investigate, experimentally, the effect of θ on the overall performance of the proposed relational similarity measure.

The computational time complexity of Algorithm 3 is $O(n|C|)$, where n is the number of patterns to be clustered and $|C|$ is the number of clusters. Usually, n is much larger than $|C|$ (i.e. $n \gg |C|$). Therefore, the overall time complexity of Algorithm 3 linearly scales with the number of patterns. The sequential nature of the algorithm avoids pairwise comparisons among all patterns. Moreover, sorting the patterns by their total word-pair frequency prior to clustering ensures that the final set of clusters contains the most common relations in the dataset.

3.3.4 Measuring Relational Similarity

Evidence from psychological experiments suggest that similarity can be context-dependent and even asymmetric [154, 100]. Human subjects have reportedly assigned different similarity ratings to word pairs when the two words were presented in reverse order. However, experimental results investigating the effects of asymmetry, report that the average difference in ratings for a word-pair is less than 5 percent [100]. Consequently, I assume relational similarity to be symmetric and limit ourselves to symmetric similarity measures. This assumption is in line with previous work on relational similarity described in section 3.1.

I model the problem of measuring relational similarity between word pairs as one of learning a Mahalanobis distance metric from a given set of relationally similar and dissimilar word pairs. Given two points $\mathbf{x_i}$, $\mathbf{x_j}$, the (squared) Mahalanobis distance between them, $d_A(\mathbf{x_i}, \mathbf{x_j})$, is parametrized using a positive definite matrix $\mathbf{A}$ as follows,

$$d_A(\mathbf{x_i}, \mathbf{x_j}) = (\mathbf{x_i} - \mathbf{x_j})^T \mathbf{A} (\mathbf{x_i} - \mathbf{x_j}). \tag{3.4}$$

Algorithm 3 Sequential pattern clustering algorithm.

```
Input: patterns $P = \{\mathbf{p_1}, \ldots, \mathbf{p_n}\}$, threshold $\theta$
Output: clusters $C$
SORT($P$)
$C \leftarrow \{\}$
for pattern $\mathbf{p_i} \in P$ do
    $max \leftarrow -\infty$
    $\mathbf{c}^* \leftarrow null$
    for cluster $\mathbf{c_j} \in C$ do
        $sim \leftarrow \text{cosine}(\mathbf{p_i}, \mathbf{c_j})$
        if $sim > max$ then
            $max \leftarrow sim$
            $\mathbf{c}^* \leftarrow \mathbf{c_j}$
        end if
    end for
    if $max > \theta$ then
        $\mathbf{c}^* \leftarrow \mathbf{c}^* \oplus \mathbf{p_i}$
    else
        $C \leftarrow C \cup \{\mathbf{p_i}\}$
    end if
end for
return $C$
```

The Mahalanobis distance is a straightforward extension of the standard Euclidean distance. In fact, if I let $\mathbf{A}$ be the identity matrix, then the Mahalanobis distance reduces to the Euclidean distance.

The motivation behind using Mahalanobis distance to measure relational similarity is two-fold. First, Mahalanobis distance can be learned from a few data points, and efficient algorithms that can scale well to high-dimensional feature spaces are known [40, 39]. Second, unlike Euclidean distance, Mahalanobis distance does not assume that features are independent. This is particularly important for relational similarity measures because semantic relations are not always independent. A posterior analysis of the Mahalanobis matrix ($\mathbf{A}$) can provide useful information related to the correlation between semantic relations.

To learn a Mahalanobis distance metric, I first represent each word-pair (a_i, b_i) as a feature vector $\mathbf{x_i}$. The j-th element of $\mathbf{x_i}$ is the total frequency of the word-pair (a_i, b_i) in the j-th cluster; it is given as $\sum_{p \in c_j} f(a_i, b_i, p)$. Here, p is a pattern in the cluster c_j, and $f(a_i, b_i, p)$ is the number of times that the word-pair (a_i, b_i) appears with the pattern p. I L_2 normalize all feature vectors.

Given a set of relationally similar pairs S and dissimilar pairs D, the problem of learning a relational similarity measure becomes one of finding a positive definite matrix $\mathbf{A}$, such that $d_A(\mathbf{x_i}, \mathbf{x_j}) \leq u$ for all $(i, j) \in S$, and $d_A(\mathbf{x_i}, \mathbf{x_j}) \geq l$ for all $(i, j) \in D$. Here u and l respectively signify upper and lower bounds of the decision threshold, and are set experimentally as described later in section 3.3.5. Intuitively, word pairs that share identical semantic relations must have a higher relational similarity. I set an additional constraint that the learned Mahalanobis matrix $\mathbf{A}$ must be "close" to the identity matrix $\mathbf{I}$ to incorporate this prior knowledge in the learning problem at hand. This keeps the Mahalanobis distance similar to the Euclidean distance; it also helps to prevent overfitting of the data. I follow the information theoretic metric learning (ITML) approach proposed by Davis et al. [40] to optimize the matrix $\mathbf{A}$.

We observe the fact that there exists a simple bijection (up to a scaling function) between the set of Mahalanobis distances and the set of equal mean multivariate Gaussian distributions to quantify the "closeness" between $\mathbf{A}$ and $\mathbf{I}$. Assuming the equal mean to be μ, for a Mahalanobis distance parameterized by $\mathbf{A}$, the corresponding Gaussian is given by $p(\mathbf{x}; A) = \frac{1}{Z} \exp(-\frac{1}{2} d_A(\mathbf{x}, \mu))$, where Z is a normalizing constant and A^{-1} is the covariance of the distribution. Then, the closeness between $\mathbf{A}$ and $\mathbf{I}$ is measurable using the Kullback-Liebler (KL) divergence between their corresponding multivariate Gaussians:

$$KL(p(\mathbf{x}; I) \parallel p(\mathbf{x}; A)) = \int p(\mathbf{x}; I) \log \frac{p(\mathbf{x}; I)}{p(\mathbf{x}; A)} d\mathbf{x}. \tag{3.5}$$

Using Formula 3.5, the learning problem can be stated as

$$\begin{aligned} & \min_A KL(p(\mathbf{x}; I) \parallel p(\mathbf{x}; A)) & \quad (3.6) \\ \text{s.t} \quad & d_A(\mathbf{x_i}, \mathbf{x_j}) \leq u \quad (i, j) \in S & \\ & d_A(\mathbf{x_i}, \mathbf{x_j}) \geq l \quad (i, j) \in D. & \end{aligned}$$

The integral form of the KL divergence presented in Formula 3.5 is difficult to numerically optimize directly. However, it has been shown that the KL divergence between two multivariate Gaussians can be expressed as the convex combination of a Mahalanobis distance between mean vectors and the LogDet divergence between the covariance matrices [38]. Therefore, assuming that the means of the Gaussians are equal, we have

$$KL(p(\mathbf{x}; I) \parallel p(\mathbf{x}; A)) = \frac{1}{2} D_{ld}(A, I). \tag{3.7}$$

Here, $D_{ld}(A, B)$ is the LogDet divergence of $n \times n$ positive-definite matrices $\mathbf{A}$, $\mathbf{B}$. It is given as

$$D_{ld}(A, B) = tr(AB^{-1}) - \log\det(AB^{-1}) - n. \tag{3.8}$$

Finally, I incorporate slack variables into the formulation 3.6 to guarantee the existence of a feasible solution for $\mathbf{A}$, and pose the following optimization problem:

$$\begin{aligned} &\min_{A \succeq 0, \xi} D_{ld}(A, I) + \gamma D_{ld}(diag(\xi), diag(\xi_{\mathbf{0}})) && \qquad (3.9)\\ \text{s.t.} \quad & tr(A(\mathbf{x_i} - \mathbf{x_j})(\mathbf{x_i} - \mathbf{x_j})^T) \leq \xi_{c(i,j)} \quad (i, j) \in S \\ & tr(A(\mathbf{x_i} - \mathbf{x_j})(\mathbf{x_i} - \mathbf{x_j})^T) \geq \xi_{c(i,j)} \quad (i, j) \in D, \end{aligned}$$

where $c(i, j)$ is the index of the (i, j)-th constraint, ξ is a vector of slack variables, initialized to $\xi_{\mathbf{0}}$ (components of $\xi_{\mathbf{0}}$ are initialized to u and v, respectively, for similar and dissimilar constraints), and γ is the parameter that controls the tradeoff between satisfying the constraints and minimizing $D_{ld}(A, I)$. Algorithm 4 solves the optimization problem 3.9 by repeatedly projecting the current solution onto a single constraint. Unlike Latent Relational Analysis [150], Algorithm 4 requires no eigen-decomposition, which is time consuming for large matrices. In Algorithm 4, a single iteration of looping through all constraints costs $O(cd^2)$, where c signifies the number of constraints, and d represents the dimensionality of feature vectors.

Once I obtain a Mahalanobis matrix $\mathbf{A}$ from Algorithm 4, I can use Formula 3.4 to compute relational distances. Distance and similarity are inversely related. Therefore, it

Algorithm 4 Information-theoretic metric learning.

Input: $\mathbf{X}$, ($d \times n$ matrix); S, set of similar pairs; D, set of dissimilar pairs; u, l: distance thresholds; $\mathbf{I}$, identity matrix; γ, slack parameter; c, constraint index function

Output: $\mathbf{A}$: Mahalanobis matrix

```
$\mathbf{A} \leftarrow \mathbf{I}$, $\lambda_{ij} \leftarrow 0 \; \forall i, j$
$\xi_{c(i,j)} \leftarrow u$ for $(i, j) \in S$; otherwise $\xi_{c(i,j)} \leftarrow l$
repeat
    Pick a constraint $(i, j) \in S$ or $(i, j) \in D$
    $p \leftarrow (\mathbf{x_i} - \mathbf{x_j})^T \mathbf{A} (\mathbf{x_i} - \mathbf{x_j})$
    $\delta \leftarrow 1$ if $(i, j) \in S$, $-1$ otherwise
    $\alpha \leftarrow \min\left(\lambda_{ij}, \frac{\delta}{2}\left(\frac{1}{p} - \frac{\gamma}{\xi_{c(i,j)}}\right)\right)$
    $\beta \leftarrow \delta\alpha/(1 - \delta\alpha\xi_{c(i,j)})$
    $\xi_{c(i,j)} \leftarrow \gamma\xi_{c(i,j)}/(\gamma + \delta\alpha\xi_{c(i,j)})$
    $\lambda_{ij} \leftarrow \lambda_{ij} - \alpha$
    $\mathbf{A} \leftarrow \mathbf{A} + \beta\mathbf{A}(\mathbf{x_i} - \mathbf{x_j})(\mathbf{x_i} - \mathbf{x_j})^T \mathbf{A}$
until convergence
return $\mathbf{A}$
```

is possible to use Formula 3.4 directly to compare word pairs. However, if one wants to convert distance values ranging in $[0, +\infty)$ to similarity scores ranging in $[0, 1]$, it can be done using sigmoid functions [121].

3.3.5 Experiments

I use two different datasets to evaluate the proposed relational similarity measure in two tasks: classifying semantic relations between named entities, and solving SAT word-analogy questions. Solving SAT word analogy questions was first proposed by Turney et al. [152] as a benchmark to evaluate relational similarity measures. An SAT analogy question consists of a stem word-pair that acts as the question, and five choice word pairs. A relational similarity measure under evaluation will compare the stem word-pair with each choice word-pair separately, and select the choice word-pair with the highest relational similarity as the correct answer. The dataset contains 374 questions.

A limitation frequently associated with the SAT dataset is that it contains no named entities or relations that Web users are typically interested in, such as relations pertaining

Table 3.7: Overview of the relational similarity dataset.

Relation Type	Total contexts	Examples (20 in all for each relation type)
ACQUIRER-ACQUIREE	91439	(Google, YouTube), (Adobe Systems, Macromedia), (Yahoo, Inktomi)
PERSON-BIRTHPLACE	72836	(Franz Kafka, Prague), (Charlie Chaplin, London), (Marie Antoinette, Vienna)
CEO-COMPANY,	82682	(Terry Semel, Yahoo), (Eric Scmidt, Google), (Steve Jobs, Apple)
COMPANY-HEADQUARTERS	100887	(Microsoft, Redmond), (Yahoo, Sunnyvale), (Google, Mountain View)
PERSON-FIELD	99660	(Albert Einstein, Physics), (Roger Federer, Tennis), (Shane Warne, Cricket)

to companies or people. Consequently, in addition to the SAT dataset, I created a dataset[3] containing only entity pairs to evaluate the proposed relational similarity measure. Hereinafter, I designate this as the **ENT** dataset. The ENT dataset contains 100 instances (i.e. named-entity pairs) of the following five relation types.

ACQUIRER-ACQUIREE This relation holds between pairs of company names *(A,B)*, where the company *B* (acquiree) is acquired by the company *A* (acquirer). I only consider acquisitions that have already completed.

PERSON-BIRTHPLACE This relation holds between pairs *(A,B)*, where *A* is the name of a person, and *B* is the location (place) where *A* was born. I consider city names and countries as locations.

CEO-COMPANY This relation holds between pairs *(A,B)*, where *A* is the chief executive officer (CEO) of a company *B*. I consider both current as well as past CEOs of companies.

COMPANY-HEADQUARTERS This relation holds between pairs *A*,*B*, where company *A*'s headquarters is located in a place *B*. I select names of cities as *B*.

[3] http://www.miv.t.u-tokyo.ac.jp/danushka/reldata.zip

PERSON-FIELD This relation holds between pairs *(A,B)*, where a person A is an expert or is known for his or her abilities in a field B. Instances of this relation contain scientists and their field of expertise, athletes and the sports they are associated with, and artists and the genre in which they perform.

I selected the relation types described above because previous studies of relation detection on the Web have frequently used those relations in evaluations [23]. I manually selected 20 instances for each of the five relation types. Instances were selected from various information sources such as Wikipedia[4], online newspapers, and company reviews[5].

For word pairs in SAT and ENT datasets using the YahooBOSS API[6], I download snippets as described in section 3.3.1. For each relation type in ENT dataset, in Table 3.7, I show some instances and the total number of contexts. I randomly split the ENT dataset into five equal-sized partitions to conduct five-fold cross validation. Four partitions are used to extract patterns, clustering and training. The remaining partition is used for testing. For ENT data, positive training instances are generated by coupling word pairs that belong to the same relation type (i.e., $5 \times (20 \times 19)/2 = 950$ instances), and an equal number of negative training instances are generated by randomly coupling word pairs that belong to different relation types.

I run the pattern extraction algorithm described in section 3.2.1 on the contexts in my dataset to extract lexical patterns. Experimentally, I set the values for the various parameters in the pattern extraction algorithm: $L = 5$, $g = 2$, and $G = 4$. The proposed pattern extraction algorithm identifies numerous lexical patterns. For example, for ENT training data, the algorithm extracts 473910 unique patterns. However, of those, only 148655 (31.36% of the total) occur more than twice. Patterns that only occur once contain misspellings or badly formatted text. I only select the patterns that occur at least twice to filter-out this noise. The remaining experiments described in this chapter are performed using those patterns. I first compute the distribution of Euclidean distances over the training data to determine the values for distance thresholds u and l in Algorithm 4. I then respectively select the 5-th and 95-th percentiles of distance distribution as u (1.96) and l (0.22).

[4] http://wikipedia.org/

[5] http://www.forbes.com/

[6] http://developer.yahoo.com/search/boss/

Slack parameter γ is set to 0.01 experimentally.

Relation Classification

I evaluate the proposed relational similarity measure in a relation classification task. Given an entity pair, the goal is to select a relation out of the five relation types in the ENT dataset that describes the relation between the two entities. This is a multi-class classification problem. I use k-nearest neighbor classification to assign a relation to a given entity pair. Specifically, given an entity pair (a, b), for which a relation R holds, I compute the relational similarity between (a, b) and the remaining entity pairs in the dataset. I then sort the word pairs in the descending order of relational similarity with (a, b), and select the most similar k entity pairs. I then find the relation that is given to most of those k entity pairs and assign this relation to (a, b). Ties are resolved randomly. This procedure is repeated for each entity pair in the ENT dataset. Overall accuracy of relation classification is computed as

$$\mathrm{Accuracy} = \frac{\text{No. of correctly classified entity pairs}}{\text{Total no. of entity pairs}}. \tag{3.10}$$

A good relational similarity measure must assign higher similarity scores to word pairs with similar implicit relations. However, the classification accuracy does not evaluate the relative rankings of similarity scores. I use average precision [132] to evaluate the top k most similar entity pairs to a given entity pair. Average precision integrates the precision at different ranks. It is frequently used as an evaluation measure in retrieval tasks. The average precision for a particular relation type R is defined as

$$\mathrm{AveragePrecision} = \frac{\sum_{r=1}^{k} \mathrm{Pre}(r) \times \text{Rel(r)}}{\text{No of relevant word pairs}}. \tag{3.11}$$

Here, $\mathrm{Rel}(r)$ is a binary valued function that returns 1 if the entity pair at rank r has the same relation (i.e., R) as in (a, b). Otherwise, it returns zero. Furthermore, $\mathrm{Pre}(r)$ is the precision at rank r, which is given as

$$\mathrm{Pre}(r) = \frac{\text{no. of entity pairs with relation R in top r pairs}}{r}. \tag{3.12}$$

The number of relevant entity pairs is 20 for all five relation types in my dataset. I consider the 10 most similar entity pairs (i.e., $k = 10$) for nearest neighbor classification. The average precision is computed for those top 10 entity pairs.

I use ENT training data to investigate the effect of clustering threshold θ (Algorithm 3) on relation classification performance. Results are summarized in Figure 3.6. Overall, in Figure 3.6, we see that performance increases with θ. This is because higher values of θ result in highly similar pattern clusters that represent specific semantic relations. However, a slight drop of performance can be observed for high θ values, because it produces a large number of pattern clusters (i.e., increased model complexity), which results in over fitting the data. The best performance is reported for $\theta = 0.905$. The remaining experiments described in this section use this value of theta.

In Table 3.8, I show the top 10 clusters with the largest number of lexical patterns. The number of patterns in each cluster is shown within brackets in the first column. For each cluster in Table 3.8, I show the top four patterns that occur in the greatest number of entity pairs. For explanatory purposes, I label the clusters with the five relation types as clusters 1 and 4 (acquirer-acquiree); clusters 2, 3, 6, and 7 (person-field); cluster 5 (CEO-company); cluster 8 and 10 (company-headquarters); cluster 9 (person-birthplace). Table 3.8 clarifies that patterns representing various semantic relations are extracted by the proposed pattern extraction algorithm. Moreover, we see that each cluster contains different lexical patterns that express a specific semantic relation. We can also find multiple clusters even among the top few clusters shown in Table 3.8 that represent a particular relation type. For example, cluster 1 and 4 both represent an *acquirer-acquiree* relation, although the patterns in cluster 1 are derived from the verb *acquire*, whereas the patterns in cluster 4 are derived from the verbs *buy* and *purchase*. We can expect a certain level of correlation among such clusters, which justifies the use of the Mahalanobis distance instead of Euclidean distance when computing relational similarity.

I compare the proposed relational similarity measure (**PROP**) to the Euclidean distance baseline (**EUC**), vector space model-based relational similarity [152] (**VSM**) and the state-of -the-art Latent Relational Analysis [150] (**LRA**). Next, I explain each of those relational similarity measures in detail.

VSM: This is the vector space model-based approach proposed by Turney et al. [152].

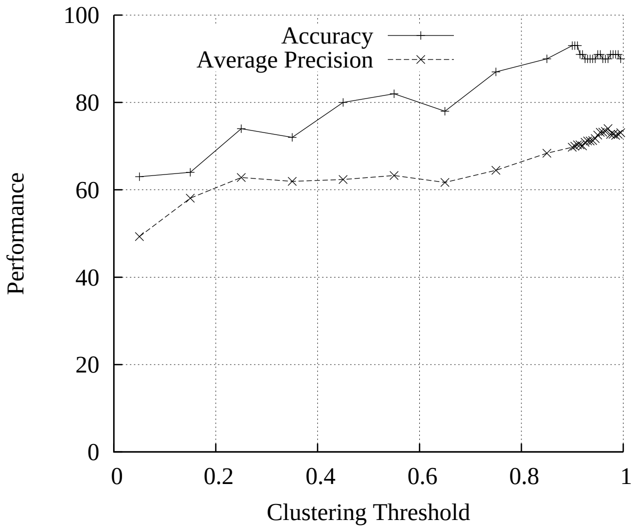


Figure 3.6: Performance of the proposed method against the clustering threshold (θ)

First, each word-pair is represented using a vector of pattern frequencies. Then the relational similarity between two word pairs is computed as the cosine of the angle between the two vectors representing the two word pairs. This approach is equivalent to computing relational similarity using Formula 3.4, if I define feature vectors as pattern frequency vectors and take the identity matrix as $\mathbf{A}$.

LRA: This is the Latent Relational Analysis (LRA) proposed by Turney [150]. First, a matrix is created, in which the rows represent word pairs and the columns represent lexical patterns. An element of the matrix corresponds to the frequency of occurrence of a word-pair in a particular lexical pattern. Next, singular value decomposition (SVD) is performed on this matrix to reduce the number of columns. Finally, the relational similarity between two word pairs is computed as the cosine of the angle between the corresponding row vectors. I re-implemented LRA as described in the original paper. However, I do not use related word thesauri to find additional word pairs, because such resources are not available for named entities. Following, Turney's proposal, I used the most frequent 4000 lexical patterns in the matrix and reduced the number of

Table 3.8: Most frequent patterns in the largest clusters.

cluster 1 (2868)	**X** acquires **Y** **X**, acquisition, **Y**	**X** has acquired **Y** **Y** goes **X**	**X**'s **Y** acquisition
cluster 2 (2711)	**Y** legend **X** was **X** autographed **Y** ball	**X**'s championship **Y** **Y** star **X** robbed	**Y** star **X** was
cluster 3 (2615)	**Y** champion **X** **X**'s greatest **Y**	world **Y** champion **X** **Y** players like **X**	**X** teaches **Y**
cluster 4 (2008)	**X** to buy **Y** **Y** purchase to boost **X**	**X** and **Y** confirmed **X** is buying **Y**	**X** buy **Y** is
cluster 5 (2002)	**Y** founder **X** **X** says **Y**	**Y** founder and CEO **X** **X** talks up **Y**	**X**, founder of **Y**
cluster 6 (1364)	**X** revolutionized **Y** ago, **X** revolutionized **Y**	**X** professor of **Y** **X**'s contribution to **Y**	in **Y** since **X**
cluster 7 (845)	**X** and modern **Y** on **Y** by **X**	genius: **X** and modern **Y** **X**'s lectures on **Y**	**Y** in DDDD, **X** was
cluster 8 (280)	**X** headquarters in **Y** the **X** conference in **Y**	**X** offices in **Y** **X** headquarters in **Y** on	past **X** offices in **Y**
cluster 9 (144)	**X**'s childhood in **Y** **Y** born **X** introduced the	**X**'s birth in **Y** sobbing **X** left **Y** to	**Y** born **X**
cluster 10 (49)	**X** headquarters in **Y** **X** works with the **Y**	**X**'s **Y** headquarters **Y** office of **X**	**Y** - based **X**

columns to 300 via SVD (i.e., eigenvectors corresponding to the largest 300 eigenvalues are used to approximate the matrix). I used Scientific Python's SVD library[7] for the computation of SVD. LRA is the current state-of-the-art relational similarity measure.

EUC: I set **A** in Formula 3.4 to the identity matrix and compute relation similarity using pattern clusters. This is equivalent to computing relational similarity between two word pairs as the Euclidean distance between the corresponding two feature vectors created using pattern clusters. This baseline is a cut-down version of the proposed method, where all clusters are assumed to be independent. This baseline is expected to show the decrease in the performance when I do not use Mahalanobis distance

[7] www.scipy.org

Table 3.9: Performance of the proposed method and baselines.

Relation	VSM	LRA	EUC	PROP
acquirer-acquiree	0.9227	0.9224	0.9147	$\mathbf{0.9415}$
comp.-headquarters	0.8455	0.8254	0.7986	$\mathbf{0.8653}$
person-field	0.4470	0.4396	0.5195	$\mathbf{0.5715}$
CEO-comp.	0.9582	$\mathbf{0.9612}$	0.9058	0.9578
person-birthplace	0.2747	0.2795	0.3343	$\mathbf{0.3648}$
Overall Average Precision	0.6896	0.6856	0.6946	$\mathbf{0.7403}$
Classification Accuracy	0.86	0.88	0.90	0.93

learning.

PROP: This is the proposed relational similarity measure, defined in Formula 3.4. For both **EUC** and **PROP**, I used the same set of clusters. Therefore, any difference in performance can be attributable to using the Mahalanobis distance when computing relational similarity. I used the 10447 clusters derived by setting the clustering threshold θ to the value 0.905.

The four methods described above presented for comparison in Table 3.9. For each relation type, Table 3.9 shows the average precision scores computed using Formula 3.11. Moreover, the overall performance is reported using both average precision and classification accuracy. The proposed method (**PROP**) reports the highest overall average precision (0.7403) in Table 3.9. In fact, **PROP** has the best average precision scores for four out of the five relation types. Analysis of variance (ANOVA) reveals that the average precision scores in Table 3.9 are statistically significant. Moreover, paired t-tests conducted between the proposed method (**PROP**) and each of the remaining three methods in Table 3.9, reveal that the improvement shown by **PROP** over **VSM**, **EUC**, and **LRA** is statistically significant ($\alpha = 0.01$). **PROP** has the highest classification accuracy (0.93), followed by **EUC**, **LRA**, and **VSM**, in that order. It is noteworthy that the **EUC** baseline that does not consider inter-cluster correlation performs better than the **VSM** method. This result shows that clustering similar patterns prior to computing relational similarity indeed improves performance. Among the five relation types compared in Table 3.9, high average precision scores are reported for the following three relation types: acquirer-acquiree, company-headquarters,

Table 3.10: Performance on the SAT dataset.

Algorithm	score	Algorithm	score
Random guessing	20.0%	LSA+Predictation [90]	42.0%
Jiang & Conrath [150]	27.3%	Veale (WordNet) [157]	43.0%
Lin [150]	27.3%	Bicici & Yuret [14]	44.0%
Leacock & Chodrow [150]	31.3%	**VSM** [151]	47.1%
Hirst & St.-Onge [150]	32.1%	**PROPOSED**	51.1%
Resnik [150]	33.2%	Pertinence [149]	53.5%
PMI-IR [150]	35.0%	**LRA** [150]	56.1%
SVM [19]	40.1%	Human	57.0%

and CEO-company. Lowest performance is reported for the person-birthplace relation. A closer look into the snippets extracted for the person-birthplace pairs reveals that there were many snippets that convey information related to places that people associate with places other than their place of birth. For example, regarding actors, the locations where they gave their first performance are incorrectly extracted as contexts for the person-birthplace relation.

Solving SAT Word Analogy Questions

Following the previous work on relational similarity measures, I use the proposed method to solve SAT word-analogy questions. I split the SAT dataset (374 questions) randomly into five partitions and select four partitions as training data and the remainder as test data. The procedure is repeated with different partitions. Then the experimental results are reported for five-fold cross-validation. For all word pairs in the SAT dataset, I download contexts from the Web (section 3.3.1), and extract lexical patterns (section 3.2.1). I then cluster the extracted patterns using Algorithm 3. Next, for each SAT question in the training dataset, I create a positive training instance by coupling the stem word-pair with the correct answer. Similarly, negative training instances are created by coupling the stem word-pair with incorrect answers. I then use Algorithm 4 to learn a Mahalanobis distance matrix from the training data. To solve an SAT question in the test dataset, I compute the relational similarity (alternatively distance) between the stem word-pair and each choice word pairs using Formula 3.4, and select the choice with the highest relational similarity (lowest distance) as the correct answer. The SAT score is computed as the percentage of correctly answered questions to the total questions in the dataset.

As shown in Table 3.10 the proposed method reports an SAT score of 51.1%; it is ranked 3rd among 16 systems. The average SAT score reported by high-school students is 57%. Randomly guessing one out of five choices gives the lower bound of 20%. The proposed method outperforms WordNet-based relational similarity measures (Veale [157]) as well as various corpus-based approaches. The two systems that perform better than the proposed method (i.e., Pertinence and LRA) use a synonym dictionary to find similar word pairs. However, the proposed method requires no external resources such as synonym dictionaries to compute relational similarity. In fact, synonym dictionaries for named entities are either not available or incomplete. Moreover, as stated in the original paper, LRA takes over 9 days to answer the 374 questions in the SAT dataset, whereas the proposed method requires less than 6 hours to answer the same set of questions. The gain in processing time can be attributable to two factors. First, unlike LRA and Pertinence, the proposed method requires no singular value decomposition (SVD). Performing SVD on large matrices is time consuming. For example, in LRA the data matrix consists of 2176 word pairs (rows) and 4000 patterns (columns). Second, compared to LRA, the proposed method requires much fewer search engine queries. In LRA, to compute the feature vector for a word-pair we must issue a query for each pattern extracted. For example, with 4000 patterns, LRA requires at least 8000 (4000×2 word pairs) search engine queries to compute the relational similarity between two word-pairs. On the other hand, the proposed method searches for patterns only within the snippets downloaded for a word-pair. Because multiple snippets can be downloaded by issuing a single query, the proposed method requires only two search engine queries to compute the relational similarity between two word pairs. Moreover, the number of search engine queries is independent of the number of patterns. Therefore, the proposed method is more appropriate in an online setting (e.g., web search), in which we must quickly compute relational similarity for unseen word pairs.

The definition of relational similarity, as given in Formula 3.4 can be viewed as a general framework into which all existing relational similarity measures can be integrated. The existing approaches differ in their definition of matrix $\mathbf{A}$. For example, in **VSM**, $\mathbf{A}$ is the identity matrix, and in **LRA** it is computed via SVD. The proposed method learns a Mahalanobis distance matrix as $\mathbf{A}$ using training data. The task of designing relational similarity measures can be modeled as searching for a matrix $\mathbf{A}$ that best reflects the notion of relational similarity possessed by humans.

The full potential of relational similarity measures in Semantic Web is largely unexplored. Measuring the similarity between relations is important for Semantic Web search. A relational similarity measure can be used to find similar relations in an ontology to improve the recall in semantic search. For example, consider the two analogous SPARQL queries *?X isCapitalOf ?Y* and *?X isStateCapitalOf ?Y*. We can expect a high degree of relational similarity between the entity pairs that satisfies each of those relations. Consequently, given the result set (entity pairs) for one query, we can use a relational similarity measure to find results for the other.

So far in this thesis I have studied semantic (attributional) similarity (Chapter 2) and relational similarity (Chapter 3) separately. As described in Section 1.8, attributes and relations have a dual connection. Therefore, a natural question that arises from our discussion is whether there is any relationship between the two types of similarities. For example, "given a relational similarity measure can we use it to measure attributional similarity?", or vice versa. I will discuss the relationship between the two types of similarity measures in detail in Chapter 7, where I propose a relational model of similarity.

Chapter 4

Personal Name Disambiguation

In this thesis, so far I have proposed and experimentally evaluated both semantic and relational similarity measures that use web search engines. As already discussed in chapter 1, similarity measures are useful in a wide range of numerous related tasks such as document/term clustering, solving analogy questions, query expansion, automatic thesauri, and word sense disambiguation. A thesis on similarity measures would not be complete if it does not consider the potential applicability of the proposed similarity measure. I dedicate this and next chapters to evaluate the contribution of similarity measures in real-world web applications. Specifically, I investigate the problem of person name disambiguation on the web. Disambiguating a person on the web can be difficult because of two main problems: *namesakes* (i.e. different people with the same name), and *name aliases* (i.e. a particular individual being referred to by more than one name). This chapter concentrates on the namesake problem. The name alias problem is investigated in chapter 5.

The problem of *measuring similarity* and the problem of *entity resolution* are closely related. By entity resolution, I collectively refer to both namesake disambiguation problem and name alias detection problem. In natural language processing field these two tasks are commonly known as *word sense disambiguation* (WSD) and *cross-document co-reference resolution*. If we consider an entity e, it can be represented by the set of attributes it has. In general, an entity can have multiple appearances on the web. What we find on the web are these appearances of an entity and not the entity itself. To differentiate the appearance of an entity from the actual entity itself, I refer to the former as an *instance* of an entity. Let us illustrate this distinction by an example. Consider the case where e is a person whose name is n. The set of web pages that contain n are potential instances of e. However, a name is only one of the many attributes of an entity. For example, for a person there can be other attributes beside a name such as the *date of birth*, *nationality*, *affiliation*, etc. As we shall see in Chapter 6, extracting attributes of an entity is a challenging task. However, to facilitate our discussion here let us assume that given an entity, we can represent it by its set of attributes. Then, the entity resolution problem can be solved by measuring the attributional similarity between instances of entities. In particular, if we consider the feature model of similarity, then the set of attributes acts as the set of features in similarity computations. The namesake disambiguation problem is where the attribute *name* takes the exact value (a string) for two or more entities. The alias detection problem is where the attribute *name* takes multiple values in different instances of the same entity.

The Internet has grown into a collection of billions of web pages. Web search engines are important interfaces to this vast information. We send simple text queries to search engines and retrieve web pages. However, due to the ambiguities in the queries, a search engine may return a lot of irrelevant pages. In the case of personal name queries, we may receive web pages for other people with the same name (*namesakes*). For example, if we search *Google* [1] for *Jim Clark*, even among the top 100 results we find at least eight different *Jim Clarks*. The two popular namesakes; *Jim Clark* the Formula one world champion (46 pages), and *Jim Clark* the founder of Netscape (26 pages), cover the majority of the pages. What if we are interested only in the Formula one world champion and want to filter out the pages for the other *Jim Clarks*? One solution is to modify our query by including a phrase such as *Formula one* or *racing driver* with the name, *Jim Clark*. This paper presents an unsupervised method to extract such phrases from the Web.

Identifying proper names is a vital first step in information integration. The same proper name could appear across different information sources. An information integration system needs to resolve these ambiguities in order to correctly integrate information regarding a particular entity. IJCAI held the workshop on Information Integration on the Web (IIWeb) in 2003, Web People Search Task (WePS) workshops[2] held in conjunction with ACL 2008

[1] www.google.com

[2] http://nlp.uned.es/weps/

and WWW 2009 are some of the recent attempts by the research community to address the name disambiguation problem.

Two tasks that can readily benefit from automatically extracted key phrases to disambiguate personal names are *query suggestion* and *social network extraction*. In query suggestion [50], a search engine returns a set of phrases to the user alongside with the search results. The user can then modify the original query using these phrases to narrow down the search. Query suggestion helps the users to easily navigate through the result set. For personal name queries, the key phrases extracted by the proposed algorithm can be used as suggestions to reduce ambiguity and narrow down the search on a particular namesake.

Social networking services (SNSs) have been given much attention on the Web recently as an application of Semantic Web. SNSs can be used to register and share personal information among friends and communities. There have been recent attempts to extract social networks using the information available on the Web[3] [102, 96]. In both Matsuo's [96] and Mika's [102] algorithms, each person is represented by a node in the social network and the strength of the relationship between two people is represented by the length of the edge between the corresponding two nodes. As a measure of the strength of the relationship between two people A and B, these algorithms use the number of hits obtained for the query *A AND B* in a web search engine. However, this approach fails when A or B has namesakes because the number of hits in these cases includes the hits for the namesakes. To overcome this problem, we could include phrases in the query that uniquely identify A and B from their namesakes.

In this chapter, I follow a three-stage approach to extract phrases that identify people with the same name. In the first stage I represent each document containing the ambiguous name by a *term-entity* (TE) model, as described in section 4.2.3. I define a pair wise contextual similarity metric based on snippets returned by a search engine, to calculate the similarity between term-entity models. In the second stage, I cluster the TE-models using the similarity metric. In the final stage, I select key phrases from the clusters that uniquely identify each namesake.

[3] http://flink.semanticweb.org/. The system won the 1st place at the Semantic Web Challenge in ISWC2004.

Personal Name Disambiguation in Social Networks

Social networks have grown both in size and popularity over the recent years. *mixi*[4], a popular social network system in Japan, reported over ten million registered users at the turn of the year 2007. As more and more people join these social networks, it is highly likely that more than one person with identical names exist in the network. In order to identify a particular person in a social network by his or her name, we must first resolve the ambiguity for that name. The proposed method can be used to annotate people using automatically extracted keywords, thereby reducing the ambiguity in the social network.

Disambiguating personal names is an essential first step in many social network extraction algorithms [102, 96]. Given a pair of names, social network extraction algorithms attempt to capture the degree of association between the two people from the Web. Various association measures such as the Jaccard coefficient [102], and Overlap coefficient [96] have been proposed to find the relationships between personal names on the Web. They use page-counts returned by a web search engine for the individual names and the conjunctive (AND) query to compute association measures. Page-count of a query is the number of different web pages in which the query words appear. Most major web search engines provide page-counts (or approximated page-counts) for user queries. However, if one or both of the names are ambiguous (i.e. if there are other people with the same names), then page-counts do not accurately reflect the association of the two persons that we are interested. One solution to this problem is to include a keyword that uniquely identifies the person under consideration from his or her namesakes. The keywords extracted by the proposed method have been successfully utilized to disambiguate real-world large-scale social networks [95].

The context of a person in a social network can be useful to disambiguate that person. For example, if you know one or more friends of a person, then you can use that information to disambiguate a person. Bekkerman and McCallum [10] showed that it is possible to disambiguate a group of people collectively in this manner. The underlying assumption here is that two ambiguous people are unlikely to have the same set of friends. However, to utilize the social context of a person to disambiguate that person, we must first either

[4] http://mixi.jp/

obtain a social network covering that person (i.e. Facebook search page indicating some of the friends of the target person) or must mine her social network from the web. The first approach has privacy issues and might not be possible when a person does not disclose her friends to the public. The second approach (as discussed in the previous paragraph) requires name disambiguation to be performed before hand.

The friend of a friend (FOAF) project[5] is an initiative to create an annotated web of people [101]. In FOAF, users can describe themselves using keywords, provide links to their home pages and introduce their friends. FOAF uses RDF (Resource Description Framework) to represent the information provided by the users. We can boost the manual annotation process in FOAF by automatically extracting keywords from the Web that describe individuals.

Personal Name Disambiguation in Ontologies

Ontologies are an important tool for formally specifying the vocabulary and relationships between concepts used on the Semantic Web. However, manually creating large-scale ontologies from scratch can be time consuming and tedious. Several methods have been proposed to boost the process of ontology creation by aligning and merging existing ontologies to create larger ontologies [110, 71, 56], and extracting ontologies from the Web [30]. Moreover, proper alignment of ontologies is important for evaluating ontologies [44], where an ontology is compared against a gold-standard using various evaluation metrics[6]. However, ambiguity among concepts in different ontologies lead to inaccurate alignments. For example, consider merging two ontologies representing employees in two companies. A particular personal name might appear multiple times in the two ontologies. A single person can be associated with different projects in an internal ontology of a company. Before merging the two ontologies one must first resolve the ambiguities for the concepts. Annotating concepts (in this Chapter I focus on personal names) with extra keywords is a useful way to disambiguate entries in an ontology.

[5] http://www.foaf-project.org/

[6] http://oaei.ontologymatching.org/2007/

4.1 Related Work

Personal name disambiguation can be seen as a special case of word sense disambiguation (WSD) [135, 98] problem which has been studied extensively in Natural Language Processing. However, there are several fundamental differences between WSD and personal name disambiguation. WSD typically concentrates on disambiguating between 2-4 possible meanings of a word, all of which are a priori known. However, with personal names on the Web, the number of different namesakes can be much larger and unknown. Moreover, WSD algorithms have access to sense annotated dictionaries and corpora. However, it is not feasible to create or maintain such resources for ambiguous personal names on the Web.

In large citation databases, author names can easily become ambiguous. In order to efficiently search for a particular publication, one must first disambiguate the author names. Besides the author names, a citation usually contains information such as the title of the publication, conference or journal name, year of publication and the number of pages. Such information have been utilized in previous work on citation disambiguation to design both supervised and unsupervised algorithms [58, 72, 141, 66, 67]. However, compared to a web page, a citation has a short and semi-structured format. One must first extract salient information related to the ambiguous name, to disambiguate personal names on the Web, This extra step involved when disambiguating names on the Web makes it a more challenging task.

Research on multi-document personal name resolution [6, 91, 47, 125] focuses on the related problem of determining if two instances with the same name and from different documents refer to the same individual. Bagga and Baldwin [6] first perform within-document co-reference resolution to form co-reference chains for each entity in each document. They then use the text surrounding each reference chain to create summaries about each entity in each document. These summaries are then converted to a bag-of-words feature vector and are clustered using the standard vector space model, often employed in information retrieval. The use of simplistic bag-of-words clustering is an inherently limiting aspect of their methodology. On the other hand, Mann and Yarowsky [91] propose a richer document representation involving automatically extracted features. However, their clustering technique can be basically used only for separating two people with the same name. Fleischman and Hovy [47] constructs a maximum entropy classifier to learn distances between documents that are subsequently clustered. However, their method requires a large training set.

Pedersen et al. [118, 117] propose an unsupervised approach to resolve name ambiguity by representing the context of an ambiguous name using second order context vectors derived using singular value decomposition (SVD) on a co-occurrence matrix. They then agglomeratively cluster the vectors using cosine similarity. Li et al. [83] propose two approaches to disambiguate entities in a set of documents: a supervisedly trained pairwise classifier and an unsupervised generative model. They do not assign keywords to different people with an ambiguous name.

Bekkerman and McCallum [10] present two unsupervised methods for finding web pages referring to a particular person: one based on link structure and another using Agglomerative/Conglomerative Double Clustering (A/CDC). Their scenario focuses on simultaneously disambiguating an existing social network of people, who are closely related. Therefore, their method cannot be applied to disambiguate an individual whose social network (for example, friends, colleagues) is not known.

Guha and Grag [54] present a re-ranking algorithm to disambiguate people. The algorithm requires a user to select one of the returned pages as a starting point. Then, through comparing the person descriptions, the algorithm re-ranks the entire search results in such a way that pages referring to the same person described in the user-selected page are ranked higher. A user needs to browse the documents in order to find which one matches the user's intended referent, which puts an extra burden on the user.

4.2 Automatic annotation of ambiguous personal names

4.2.1 Problem definition

Definition 1 *Given an entity e which has the name n, I call it* ***ambiguous*** *if there is at least one other entity $e\prime$ which has the same name n.*

For example, in the case of people, if two or more people have the same personal name n,

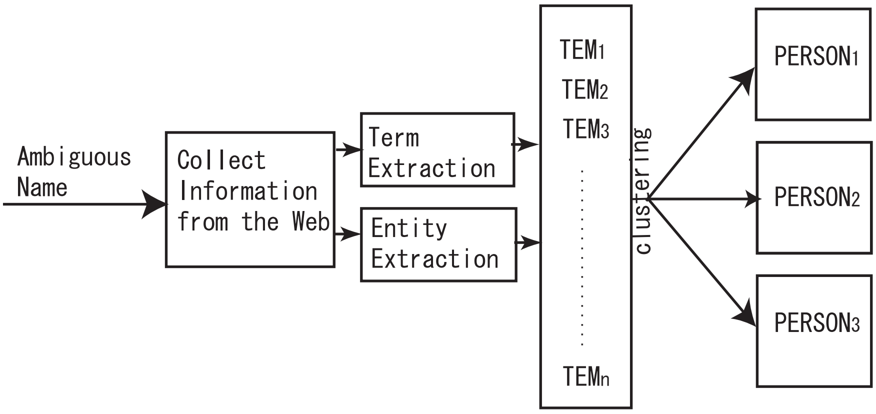


Figure 4.1: A TEM is created from each search result downloaded for the ambiguous name. Next, TEMs are clustered to find the different namesakes. Finally, discriminative keywords are selected from each cluster and used for annotating the namesakes.

then they are collectively called as *namesakes* of the ambiguous name n.

Definition 2 *Given an ambiguous entity e, the problem of automatic annotation of e is defined as the task of finding a set of words (or multiword expressions) $W(e)$, that uniquely identify e from his or her namesakes.*

For example, in our example of *Jim Clark*, the set of words (or multiword expressions) racing driver, formula one, scotsman can identify the Formula One racing champion Jim Clark from the other Jim Clarks in the Web. It is noteworthy that the name string (i.e. *jim clark* itself is not unique and does not belong to the set of words. In practice, whether a particular word (or a multiword expression) can uniquely identify a namesake of a given ambiguous name, can be difficult to decide. In this thesis, I take a pragmatic approach and decide a word (or a multiword expression) can uniquely identify a person, if that word together with the person's name (e.g. in a conjunctive query) can return results only for that person in a web search engine.

4.2.2 Outline

The proposed method is outlined in Figure 4.1. Given an ambiguous personal name, the first step in my algorithm is to collect information regarding the different people with that name from the Web. Retrieving a relevant set of documents for a query, is a fundamental problem that has been studied extensively in information retrieval [5]. I assume the availability of a web search engine and query for a given ambiguous personal name to collect information regarding the different people who are represented on the web by that name. Specifically, given an ambiguous personal name n, I download the top N ranked search results and only consider the namesakes with name n that appear in this set of N results.

How to represent information about people and their web activities, is an important problem that any web-based identity disambiguation method must consider. For this purpose, I propose *Term-Entity Models* (TEMs). TEMs are sets of terms and/or named entities that are closely associated with the people with an ambiguous name on the web. I formally define TEMs in Section 4.2.3, and describe an unsupervised method to create TEMs for a given personal name in Section 4.2.4. To determine whether two documents disclose information about the same person, we must compare the TEMs created for the two documents. However, measuring the similarity between TEMs is not a trivial task. For example, consider the two phrases *Formula One* and *Racing Championship*. The two phrases are closely related because Formula One is a racing championship. However, there are no words in common (i.e. zero word overlap) between those two phrases. To infer that the two TEMs in this example correspond to two documents that are about the racing champion Jim Clark, we must accurately measure the similarity between phrases (terms and/or named-entities). I employ a contextual similarity measure (Section 4.2.5) for this purpose.

I make the assumption that all occurrences of a given ambiguous personal name within a document (e.g. a web page) refer to the same individual. Under this assumption, identifying the different namesakes for a given name can be modeled as a document clustering problem. Initially, a TEM is created from each downloaded document. Next, I cluster those TEMs to identify the different people (namesakes) for the ambiguous personal name. I use group-average agglomerative hierarchical clustering for this purpose. A document is assigned to only one cluster (i.e. hard clustering). Ideally, each final cluster formed by this process must represent a different namesake. Therefore, the number of clusters must be equal to the number of different namesakes of the given name. However, in reality it is not possible to know in advance the number of different namesake of a given name on the web. Therefore, I terminate the agglomerative clustering process when the overall cluster quality (defined in Section 4.2.6) drops below a pre-defined threshold value. The threshold is determined using a development dataset. Finally, I select unique keywords from each cluster, and annotate the different namesakes using the extracted keywords.

4.2.3 Term-Entity Models

When searching for an individual who has numerous namesakes on the Web, one quick solution that we frequently adopt is to append the query with one or two keywords that identify the person we are interested in from his or her namesakes. For example, if we want to search for *Jim Clark* the *racing driver* we could append the query with keywords such as *Racing Driver, Formula One* or *Scotsman*. These keywords enable us to filter-out the search results for other *Jim Clarks*. I extend this idea and propose a keyword-based model to represent individuals on the Web.

Definition 3 *A* ***Term-Entity Model*** *(TEM) of a person p is a set of terms or named-entities that uniquely describes that person from his or her namesakes. Terms and/or named entities that construct a TEM are called* ***elements*** *of the TEM.*

I use the notation $T(p)$ to denote the TEM of a person p. Then with the conventional set notation we can write,

$$T(p) = \{e_1, e_2, \ldots, e_n\}. \tag{4.1}$$

Here, $e_1, e_2, \ldots, e_n$ are the elements of $T(p)$ and can be terms or named entities.

For example, TEM for $\mathrm{JimClark}_{\mathrm{driver}}$, the racing champion, could be,

$$T(\mathrm{JimClark}_{\mathrm{driver}}) = \{\mathrm{Formula\ One}, \mathrm{Racing\ Driver}, \mathrm{Champion}\}. \tag{4.2}$$

In this example, *Racing Driver* and *Champion* are terms whereas, *Formula One* is a named-entity. I use the subscript notation here to indicate a namesake of a name.

TEMs capture the essence of the keyword-based boolean web queries we are accustomed to. For simplicity, if we limit ourselves to conjunctive queries (*AND* queries), then the elements of an TEM act as the literals of the boolean query that identifies a person with the ambiguous name. Moreover, TEMs can be considered as a scaled down version of the bag-of-words (BOW) model [92], which is commonly used in information retrieval. Bag-of-words model represents a document as a set of words. However, considering all the words as identifiers of a person is noisy and inefficient. The reasons for using both terms and named-entities in TEMs are two fold. Firstly, there are multi-word phrases such as the *secretary of state, chief executive officer, racing car driver* which are helpful when identifying people on the web but are not recognized as named-entities. Secondly, automatic term extraction [48] can be carried out using statistical methods, and does not require extensive linguistics resources such as named entity dictionaries, which might not be readily available for some domains. It is noteworthy that I do not distinguish terms from named-entities once I have cerated a TEM for a person. All elements in a TEM are considered equally, irrespective of whether they are terms or named-entities in the subsequent processing. In practice, some terms (e.g. secretary of state) can also be extracted by certain named-entity recognizers (NERs) (i.e. secretary of state as a PERSON entity), and vice-versa. The motivation of using both named-entities as well as terms in the model is to obtain a better representation of a person.

4.2.4 Creating Term-Entity Models from the Web

Given an ambiguous personal name, I extract terms and named entities from the *contexts* retrieved from a web search engine for the name to create its TEM. If the ambiguous name n appears in a document D, then the context of n could be for example a paragraph containing n, a fixed window of words including n, or the entire text in document D. Other than for some exceptional cases, such as a search results page for an ambiguous name from a search engine or a disambiguation entry in Wikipedia, usually a single web page does not include more than one namesake of an ambiguous name. In fact, all previous work in web-based name disambiguation have modeled this problem as a web page clustering problem, where each cluster represents a different person of the given ambiguous name. Following these lines, I consider the entire document where an ambiguous name appears as its context.

For automatic multi-word term extraction, I use the *C-value* measure proposed by Frantzi et al. [48]. The C-value approach combines linguistic and statistical information, emphasis being placed on the statistical part. The linguistic information consists of the part-of-speech tagging of the document being processed, the linguistic filter constraining the type of terms extracted, and a stop word list. First, the context from which we need to extract terms is tagged using a part of speech tagger. Next a set of pre-defined POS patterns are used to extract candidate terms. For example, the POS pattern $(NN+)$ extracts noun phrases, and $(AdJ)(NN+)$ extracts noun phrases modified by an adjective. Here, NN represents a single noun, Adj represents a single adjective, and $+$ matches one or more occurrences of the preceding term. A list of stop words can be used to prevent extracting common words that are not considered as terms in a particular domain. Having a stop words list improves the precision of term extraction. However, in my experiments I did not use a stop words list because it is not possible to determine in advance the domain which a namesake belongs to.

The sequences of words that remain after this initial filtering process (here onwards referred to as candidates) are evaluated for their *termhood* (likeliness of a candidate to be a term) using the *C-value* measure which is defined as,

$$\text{C-value}(a) = \begin{cases} \log_2 |a| \cdot f(a) & a \text{ is not nested,} \\ \log_2 |a| (f(a) - \frac{1}{P(T_a)} \sum_{b \in T_a} f(b)) & \text{otherwise} \end{cases} . \tag{4.3}$$

Here, a is the candidate string, $f(a)$ is its frequency of occurrence in a corpus, $|a|$ is the length of the candidate string in words, T_a is the set of extracted candidate terms that contain a, $P(T_a)$ is the number of candidate terms.

C-value is built using statistical characteristics of the candidate string, such as the total frequency of occurrence of the candidate string in the document, the frequency of the candidate string as part of other longer candidate strings, the number of these longer candidate terms, and the length of the candidate string (measured in the number of words). The higher the C-value of a candidate, more likely it is a term. In my experiments I select candidates with C-value greater than 2 as terms. (see [48] for more details on C-value-based multi-word term extraction). However, there are cases where the terms extracted from the C-value method tend to be exceedingly longer and meaningless. For example, I get the term *Search Archives Contact Us Table Talk Ad* from a page about the Netscape founder, Jim Clark. This term is a combination of words extracted from a navigation menu and is not a genuine term. To avoid such terms I use two heuristics. First, I ignore any terms which are longer than four words. Second, for the remaining terms, I check their page-counts I obtain from a web search engine. The assumption here is that, if a term is a meaningful, then it is likely to be used in many web pages. I ignore any terms with less than five page-counts. Those heuristics allow us to extract more expressive and genuine terms.

To extract entities for TEMs, the contexts are tagged using a named entity tagger developed by the Cognitive Computation Group at UIUC[7]. From the tagged contexts I select personal names, organization names and location names to include in TEMs. UIUC named-entity tagger has an F1-score of 90.80% on CoNLL03 evaluation dataset [123].

4.2.5 Contextual Similarity

We must calculate the similarity between TEMs derived from different contexts, in order to decide whether they represent the same namesake or not. WordNet[8] based similarity metrics have been widely used to compute the semantic similarity between words in sense disambiguation tasks [7, 98]. However, most of the terms and entities are proper names or multi-word expressions which are not listed in the WordNet.

Sahami et al. [131] proposed the use of snippets returned by a Web search engine to calculate the semantic similarity between words. A snippet is a brief text extracted from a document around the query term. Many search engines provide snippets alongside with a link to the original document. Snippets help a web search engine user to decide whether a search result is relevant without actually having to click the link. Because snippets capture the immediate surrounding of the query in a document, we can consider a snippet to be the context of the query term. Using snippets is also efficient because it obviates the need to download the source documents from the web, which can be time consuming depending on the size of the page. To calculate the contextual similarity between elements (terms and entities) in TEMs, I first collect snippets for each element by querying a web search engine for that element. I then represent each snippet S_i as a vector $\vec{v_i}$ of words that appear in S_i. Each word in a snippet is weighted using TF-IDF weighting method [5]. Weight w_{ij} of a

[7] http://l2r.cs.uiuc.edu/˜cogcomp/

[8] http://wordnet.princeton.edu/perl/webwn

word T_j that appears in a snippet S_i is defined as follows,

$$w_{ij} = tf_{ij} \log_2 \frac{N}{df_j}. \tag{4.4}$$

Here, tf_{ij} is the term-frequency of word T_j in the snippet S_i (i.e. the number of times that T_j occurs in S_i). N is the total number of snippets retrieved from the search engine for the query, and df_j is the document-frequency of word T_j (i.e. the number of snippets that contained the word T_j). I then compute the centroid vector, $\vec{C(a)}$, for a query a by averaging all the snippet-word vectors $\vec{v_i}$ as follows,

$$\vec{C(a)} = \frac{1}{N} \sum_{i=1}^{N} \vec{v_i}. \tag{4.5}$$

Moreover, I normalize centroid vectors such that their L_2 norm is 1. Normalizing snippet-word vectors to unit length enables us to compare snippets with different numbers of words. I define the contextual similarity, $\mathrm{ContSim}(a, b)$, between two elements a, b, in TEMs as the inner product between their centroid vectors $\vec{C(a)}$, $\vec{C(b)}$.

$$\mathrm{ContSim}(a, b) = \vec{C(a)} \cdot \vec{C(b)} \tag{4.6}$$

Let us illustrate the above mentioned contextual similarity measure by an example. Consider computing the association between the two phrases *“George Bush”* and the *“President of the United States”*. First, I issue the query *“George Bush”* to a web search engine and download snippets. In this example, I download the top 100 ranked snippets by Google for the query. I then use TF-IDF method (Equation 4.4) to weight the words in snippets. Each snippet is represented by a vector of words weighted by TF-IDF. Because we have 100 snippets in this example we obtain 100 vectors. Next, the centroid vector of those 100 vectors is computed using Equation 4.5. Similarly, a centroid vector is computed for the query *“President of the United States”*. Finally, the similarity between the two phrases is computed as the inner product between the corresponding centroid vectors using Equation 4.6.

Figure 4.2 shows the distribution of most frequent words in snippets for these two

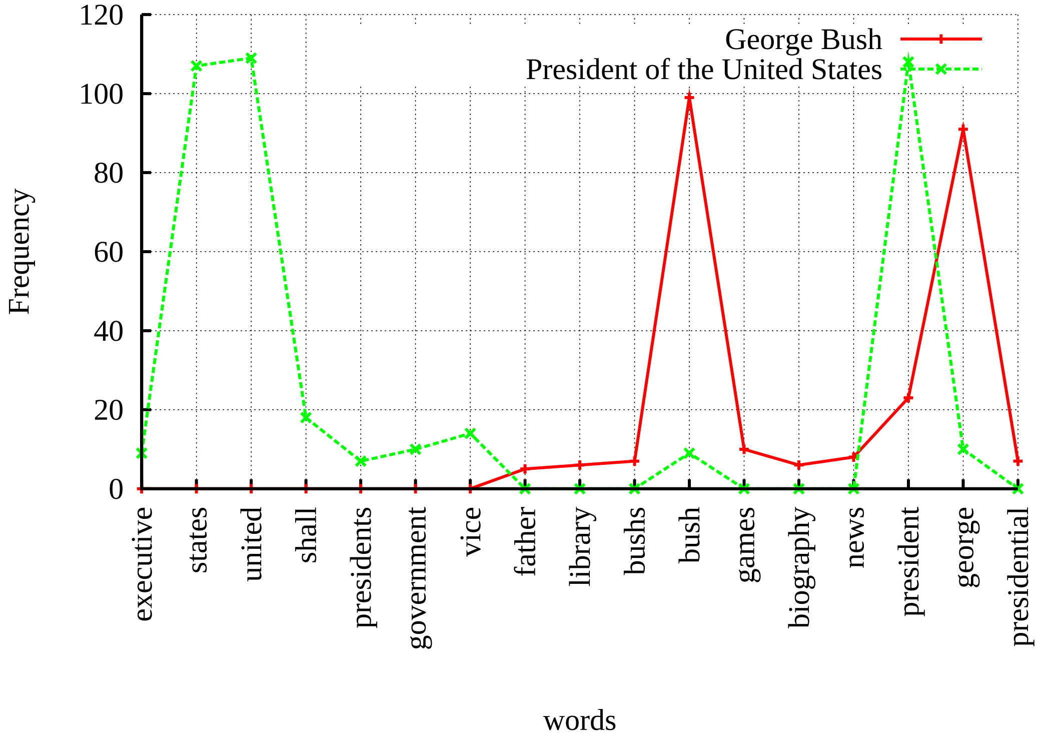


Figure 4.2: Distribution of words in snippets for "George Bush" and "President of the United States"

queries. We can observe a high overlap between the two distributions. Interestingly, the words *george* and *bush* appear with a high frequency among the snippets for the query *"President of the United States"* and word *president* appears with a high frequency for the query *"George Bush"*. The contextual similarity between the two queries is 0.2014.

On the other hand, if we compare snippets for the queries *"Tiger Woods"* and *"President of the United States"* (as shown in Figure 4.3) we get a relatively low similarity score of 0.0691. This indicates *"George Bush"* is more closely related to the phrase the *"President of the United States"* than *"Tiger Woods"* is.

Using the snippet-based contextual similarity measure, I define the similarity $\mathrm{sim}(T(A), T(B))$, between two TEMs $T(A) = \{a_1, \ldots, a_n\}$ and $T(B) = \{b_1, \ldots, b_m\}$ of contexts A and B

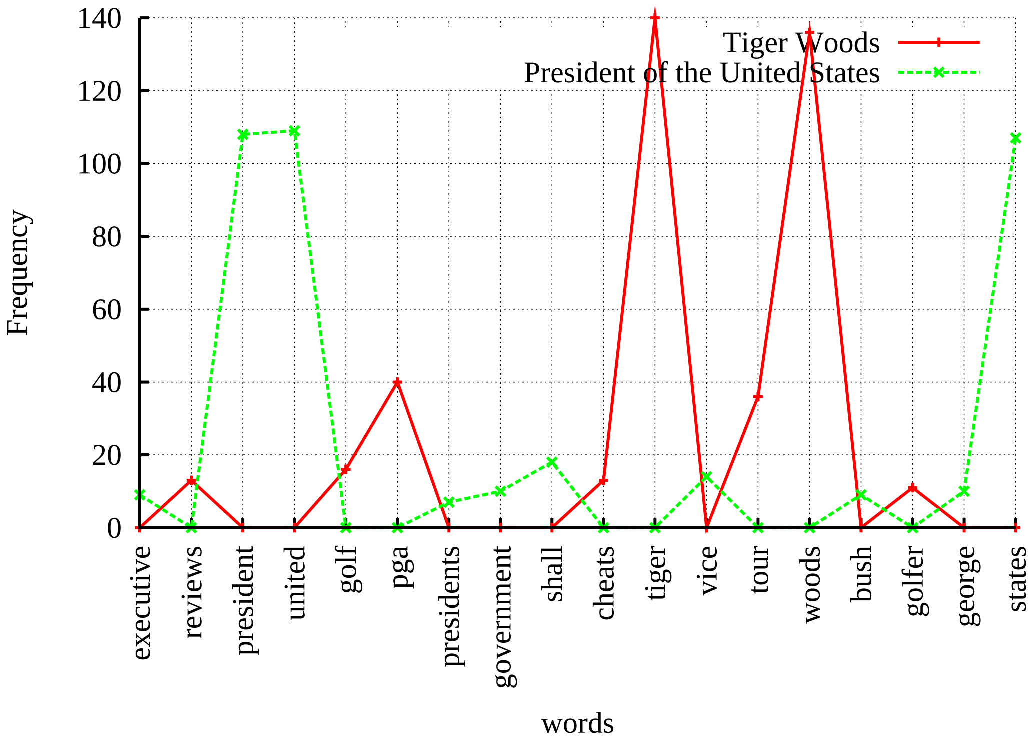


Figure 4.3: Distribution of words in snippets for "Tiger Woods" and "President of the United States"

as follows,

$$\mathrm{sim}(T(A), T(B)) = \frac{1}{nm} \sum_{i,j} \mathrm{ContSim}(a_i, b_j). \tag{4.7}$$

Therein; $\mathrm{ContSim}(a_i, b_j)$ is the contextual similarity between elements a_i and b_j, and it is given by Equation 4.6.

4.2.6 Clustering

I use Group-Average Agglomerative Clustering (GAAC) [35], a hybrid of single-link and complete-link clustering, to cluster the contexts that belong to a particular namesake. Initially, I assign a separate cluster for each of the contexts in the collection. Then, GAAC in each iteration executes the merger that gives rise to the cluster Γ with the largest average correlation $C(\Gamma)$ where,

$$C(\Gamma) = \frac{1}{2}\frac{1}{|\Gamma|(|\Gamma|-1)}\sum_{u\in\Gamma}\sum_{v\in\Gamma}\mathrm{sim}(T(u),T(v)). \tag{4.8}$$

Here, $|\Gamma|$ denotes the number of contexts in the merged cluster Γ; u and v are two contexts in Γ, and $\mathrm{sim}(T(u),T(v))$ is given by Equation 4.7.

Ideally, the number of clusters formed by the GAAC process must be equal to the number of different namesakes for the ambiguous name. However, in reality it is impossible to exactly know the number of namesakes that appear on the Web for a particular name. Moreover, the distribution of pages among namesakes is not even. For example, among the top 100 results retrieved for the name "Jim Clark" from Google, 78 belong to the two famous namesakes; *Founder of Netscape* and *Formula One world champion*. The remaining 22 search results (web pages) are distributed among six other namesakes. If these outliers get attached to the otherwise pure clusters, both disambiguation accuracy and keywords selection deteriorate. Therefore, I monitor the *quality* of clustering and terminate further agglomeration when the cluster quality drops below a pre-set threshold value. Numerous metrics have been proposed for evaluating the quality of clustering [73]. I use normalized cuts measure proposed by Shi and Malik [137].

Let V denote the set of contexts for a name. Consider, $A \subseteq V$ to be a cluster of contexts taken from V. For two contexts x,y in V, $\mathrm{sim}(x,y)$ represents the contextual similarity between the contexts (Equation 4.7). Then, the normalized cut $N_{cut}(A)$ of cluster A is defined by,

$$N_{cut}(A) = \frac{\sum_{x\in A\; y\in(V-A)}\mathrm{sim}(x,y)}{\sum_{x\in A\; y\in V}\mathrm{sim}(x,y)}. \tag{4.9}$$

For a set, $\{A_1,\ldots,A_n\}$ of non-overlapping n clusters A_i, I define the *quality* of clustering, $\mathrm{Quality}(\{A_1,\ldots,A_n\})$, as follows,

$$\mathrm{Quality}(\{A_1,\ldots,A_n\}) = \frac{1}{n}\sum_{i=1}^{n}N_{cut}(A_i). \tag{4.10}$$

For the set of clusters formed at each iteration of the agglomerative clustering process, I compute the cluster quality using Equation 4.10. I terminate the clustering process, if the cluster quality drops below a fixed threshold θ. Finally, I assign the remaining contexts (singletons) to the already formed clusters based on the correlation (Equation 4.8) between a context and a cluster. I experimentally determine the cluster stopping threshold θ using a development dataset as described later in Section 4.4.4.

4.2.7 Automatic annotation of namesakes

GAAC process produces a set of clusters representing each of the different namesakes of the ambiguous name. To annotate the namesakes represented by the formed clusters, I select elements (terms and entities) from TEMs in each cluster. To select appropriate keywords to annotate a person represented by a cluster, I first compute the union of all TEMs in that cluster. I then remove any elements that appear in other clusters. This process yields a set of elements that uniquely represents each cluster. Finally, I rank each element in a cluster according to its similarity with the given ambiguous name. I use Formula 4.6 to compute the similarity between the given name and an element. The context of a name is approximated by the top ranking snippets. I used the top ranked 100 snippets in my experiments. For example, in Section 4.2.5, I computed the similarity between the name *George Bush* and the element (a term) *President of the United States* to be 0.2014. The motivation behind ranking elements is to identify the keywords which are closely related to the namesake. Each namesake is annotated using the top ranking elements in his or her cluster.

Alternatively, we can first rank all the elements in each cluster using the similarity between the name and the element using Equation 4.6, and subsequently remove any elements that are ranked below a certain rank. Optionally, we can remove elements that appear in more than one cluster to obtain a set of keywords that uniquely identify a cluster. This alternative approach is particularly useful when there are multiple namesakes who are popular in a particular field. However, this approach requires more web search queries compared to the previous approach, because we must first compare *all* elements in a cluster with the given name in order to rank them. On the other hand, first removing common elements in different clusters can significantly reduce the number of comparisons (thereby the web search queries). Furthermore, during my preliminary experiments with this second approach I did not notice any significant improvement in the quality of keywords obtained at the cost of additional web queries. Therefore, I adopted the first approach which require comparatively lesser number of web search queries, where I first remove elements that appear in multiple clusters and subsequently rank the remaining elements.

4.3 Evaluation Datasets

To evaluate the ability to disambiguate and annotate people with the same name, I create a dataset for ambiguous personal names; *Jim Clark* and *Michael Jackson*. For each of those names, I query Google and download the top ranking search results. I then manually annotate each search result by reading the content in each downloaded web page. I exclude pages that only contain non-textual data, such as images. A web page is assigned to only one namesake of the ambiguous personal name under consideration. Moreover, I evaluate on two datasets created in previous work on namesake disambiguation: Pedersen and Kulkarni [116, 115]'s dataset (5 ambiguous names: *Richard Alston, Sarah Connor, George Miller, Michael Collins* and *Ted Pedersen*), and Bekkerman and McCallum [10]'s dataset (12 ambiguous names: *Adam Cheyer, William Cohen, Steve Hardt, David Israel, Leslie Pack Kaelbling, Bill Mark, Andrew McCallum, Tom Mitchell, David Mulford, Andrew Ng, Fernando Pereira* and *Lynn Voss*. By using the same datasets used in previous work, I can directly compare the proposed method with previous work on namesake disambiguation.

To create a gold standard for namesake disambiguation one must first manually annotate each search result retrieved for an ambiguous personal name. All datasets mentioned above take this approach. As an alternative approach that does not require manual annotation of search results, Pedersen et al. [118] propose the use of *pseudo ambiguous names*. In this approach, first a set of unambiguous personal names are manually selected. For each of the names in this set there must be only one individual in the web. Next, a web search engine is queried with each of the unambiguous names separately and search results are downloaded. Finally, each occurrence of the queried name is replaced by an identifier (e.g. person-X) and the search results retrieved for all the unambiguous names are conflated to create a single dataset. This conflated dataset can be considered as containing namesakes for the pseudo-ambiguous name, person-X. Moreover, we know which search result belongs to which namesake without any manual annotation because we replace a name with an identifier by ourselves. Although this process obviates the need for manual annotation, thereby enabling us to easily create a large dataset, it is sometimes criticized because it does not reflect the natural distribution of namesakes for real-world ambiguous personal names. Following the previous work on this line, for automated pseudo-name evaluation purposes, I select the four names (*Bill Clinton*, *Bill Gates*, *Tom Cruise* and *Tiger Woods*) for conflation. I download the top 100 ranking search results from Google for each of these names and manually confirmed that the search results did not contain any namesakes of the selected names. I then replace the ambiguous personal name by the string "person-X" in the collection, thereby artificially introducing ambiguity. The complete dataset that I used for experiments is shown in Table 4.1. I have grouped the names in Table 4.1 according to the datasets that they belong to. Moreover, names within a particular dataset are sorted alphabetically.

4.4 Experiments and Results

4.4.1 Outline

In this section I present the numerous experiments I conduct to evaluate the proposed personal name disambiguation algorithm. In my experiments, I query Google[9] for a given ambiguous personal name and download the top ranked 100 web pages. I eliminate pages that do not contain any text. I use Beautiful Soup[10], an HTML parser, to extract text from HTML pages. Next, I create a TEM from each resulting web page as described in Section 4.2.4. The set of web pages downloaded for the given ambiguous personal name is then clustered using the clustering algorithm described in Section 4.2.6. I use contextual similarity (Equation 4.6) to compute the similarity between elements (i.e. terms or named-entities that appear in a term-entity model) in TEMs created for web pages. In Formula 4.6, I use the top ranking 100 snippets returned by Google for an element as its context.

[9] http://code.google.com/

[10] http://www.crummy.com/software/BeautifulSoup/

Table 4.1: Experimental Dataset

Name	number of namesakes	number of contexts
person-X	4	137
Jim Clark	8	100
Michael Jackson	2	82
Pedersen and Kulkarni's dataset [116, 115]		
George Miller	3	286
Michael Collins	4	359
Richard Alston	2	247
Sarah Connor	2	150
Ted Pedersen	4	333
Bekkerman and McCallum's dataset [10]		
Adam Cheyer	2	97
Andrew McCallum	8	94
Andrew Ng	29	87
Bill Mark	8	94
David Israel	16	92
David Mulford	13	94
Fernando Pereira	19	88
Leslie Pack Kaelbling	2	89
Lynn Voss	26	89
Steve Hardt	6	81
Tom Mitchell	37	92
William Cohen	10	88
Total	205	2779

In Section 4.4.2, I describe *disambiguation accuracy*, the evaluation measure used in the experiments. In Section 4.4.3, I compare disambiguation accuracy with cluster quality introduced in Section 4.2.6. I determine the cluster stopping threshold θ using development data in Section 4.4.4. In Section 4.4.5, I compare the performance of the proposed method against baseline methods and previous work on namesake disambiguation. Moreover, in Section 4.4.6, I employ the keywords selected for different namesakes of an ambiguous personal name in an information retrieval task

4.4.2 Evaluation Measure

To evaluate the clusters produced by the proposed method I compare them with the gold-standard clusters for a name in a dataset. For each ambiguous personal name, the gold standard contains a set of contexts (web pages) downloaded and assigned to a namesake. In the gold standard a web page is assigned to only one of the namesakes of the given name. Therefore, we can consider the set of contexts in the gold standard for a particular name as a set of non-overlapping clusters. I compare the set of clusters in the gold standard with the set of clusters produced the proposed method. If the two sets of clusters are similar, then we can conclude that the proposed method can accurately disambiguate namesakes for a given personal name.

First, I assign each cluster to the namesake that has the most number of contexts (web pages) for him or her in that cluster. If there is more than one namesake in a cluster with the highest number of contexts, then I randomly select one of those namesakes and assign to the cluster. This process assigns a cluster to one of the namesakes for the given personal name. The purpose of this assignment is to automatically *label* a cluster by the namesake that it represents. Identifying a cluster with a namesake enables us to compare the set of clusters produced by a namesake disambiguation algorithm against a set of gold standard clusters. Next, I evaluate experimental results based on the confusion matrix A, where $A[i.j]$ represents the number of contexts for "person i" predicted as "person j". $A[i, i]$ represents the number of correctly predicted contexts for "person i". I define *disambiguation accuracy* as the sum of diagonal elements divided by the sum of all elements in the matrix as follows,

$$\text{Disambiguation Accuracy} = \frac{\sum_i A(i,i)}{\sum_{i,j} A(i,j)}. \tag{4.11}$$

If all contexts are correctly assigned for their corresponding namesakes then the confusion matrix A becomes a diagonal matrix and the disambiguation accuracy becomes 1. In practice, the number of clusters produced by a namesake disambiguation system might not necessarily be equal to the number of namesakes for an ambiguous personal name. The above mentioned cluster assignment procedure can assign multiple clusters to a particular namesake, or not assign any cluster for some namesakes, depending on the set of clusters produced by a system. However, it is noteworthy that disambiguation accuracy can still be

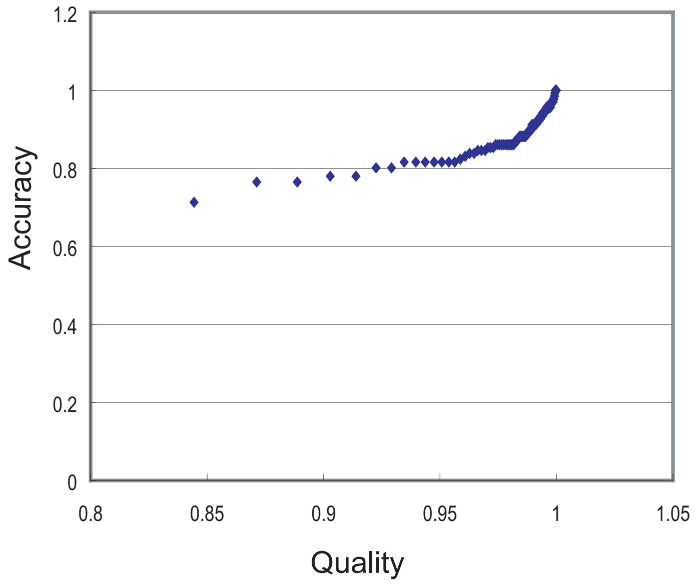


Figure 4.4: Accuracy vs Cluster Quality for person-X data set.

computed using the definition in Equation 4.11 even under such circumstances.

4.4.3 Correlation between cluster quality and disambiguation accuracy

In Section 4.2.6, I proposed the use of cluster quality (which can be computed unsupervisedly without using the gold standard clustering) to determine when to stop the agglomerative clustering process. However, it remains unknown whether the cluster quality can accurately approximate the actual accuracy of a clustering algorithm. In order to evaluate how well does normalized cuts-based cluster quality reflects the accuracy of clustering, I compare disambiguation accuracy (computed using the gold-standard) with cluster quality (computed using Equation 4.10) for person-X collection as shown in Figure 4.4. For the data points shown in Figure 4.4, we observe a high correlation between accuracy and quality (Pearson correlation coefficient between accuracy and quality is 0.865). This result enables us to guide the clustering process and determine the optimal number of clusters using cluster quality.

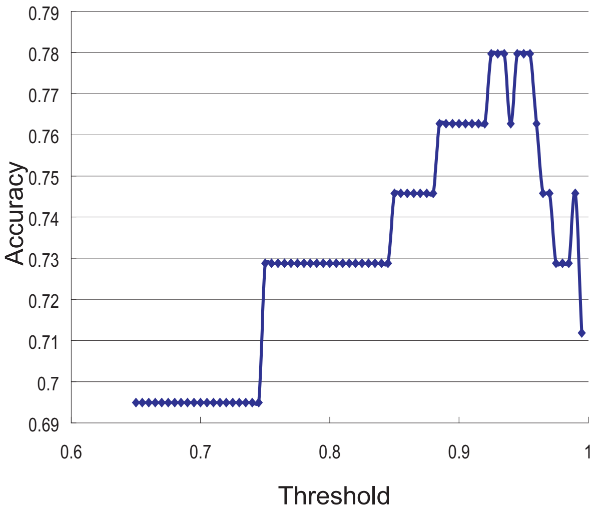


Figure 4.5: Accuracy vs Threshold value for person-X data set.

4.4.4 Determining the Cluster Stopping Threshold θ

In Section 4.2.6 I described a group-average agglomerative hierarchical clustering algorithm to cluster the contexts (i.e. web pages). I empirically determine the cluster stopping threshold θ using person-X collection as a development dataset. Figure 4.5 shows the accuracy of clustering against various threshold values. According to Figure 4.5 I set the threshold at 0.935 where accuracy maximizes for person-X collection. Threshold θ is fixed at this value (0.935) for the remainder of the experiments described in the paper.

4.4.5 Clustering Accuracy

Table 4.2 summarizes experimental results for the accuracy of the clustering. For each ambiguous name in the dataset, the second column in Table 4.2 shows the number of different people in the collection with that name. Moreover, I have visualized the experimental results in Figure 4.6 to show the overall trend. I compare the proposed method against the following three baselines.

Jaccard (Jaccrad coefficient-based clustering) : This method computes the similarity between two TEMs using the Jaccard coefficient. Jaccard coefficient between two sets

Table 4.2: Comparing the proposed method against baselines.

Name	Namesakes	Jaccard	Overlap	Proposed	Majority
person-X	4	0.8382(4)	0.7941(4)	0.7941(4)	0.6985(1)
Jim Clark	8	0.8475(3)	0.8305(3)	0.8475(3)	0.6949(1)
Michael Jackson	2	0.9706(2)	0.9706(2)	1.0000(2)	0.6765(1)
Pedersen and Kulkarni's dataset [116, 115]					
George Miller	3	0.9441(3)	0.9231(3)	0.9895(3)	0.7762(1)
Michael Collins	4	0.9777(4)	0.9861(4)	0.9889(4)	0.8357(1)
Richard Alston	2	0.9109(2)	0.9069(2)	0.9960(2)	0.7368(1)
Sarah Connor	2	0.9333(2)	0.9333(2)	0.9867(2)	0.7267(1)
Ted Pedersen	4	0.9820(4)	0.9670(4)	0.9850(4)	0.8378(1)
Bekkerman and McCallum's dataset [10]					
Adam Cheyer	2	0.9897(1)	0.9897(1)	0.9897(1)	0.9897(1)
Andrew McCallum	8	0.7447(3)	0.7340(3)	0.7766(4)	0.7660(1)
Andrew Ng	29	0.5172(7)	0.4943(6)	0.5747(5)	0.6437(1)
Bill Mark	8	0.6702(2)	0.6383(2)	0.8191(4)	0.6064(1)
David Israel	16	0.5217(3)	0.5217(4)	0.6739(4)	0.5435(1)
David Mulford	13	0.6702(3)	0.6809(2)	0.7553(4)	0.7128(1)
Fernando Pereira	19	0.4886(5)	0.4886(6)	0.6364(6)	0.5455(1)
Leslie Pack Kaelbling	2	0.9888(1)	0.9888(1)	0.9888(1)	0.9888(1)
Lynn Voss	26	0.4607(7)	0.4045(4)	0.6404(9)	0.5056(1)
Steve Hardt	6	0.8642(2)	0.8148(2)	0.8148(2)	0.8272(1)
Tom Mitchell	37	0.3478(9)	0.3696(8)	0.4891(13)	0.5870(1)
William Cohen	10	0.7955(2)	0.7841(2)	0.8295(3)	0.7955(1)
Overall Average		0.7732	0.7610	0.8288	0.7247

A and B is defined as follows,

$$\text{Jaccard} = \frac{|A \cap B|}{|A \cup B|}. \tag{4.12}$$

Here, $|A \cap B|$ denotes the number of elements in the intersection of sets A and B, $|A \cup B|$ is the number of elements in the union of sets A and B. If two TEMs share many elements then the Jaccard coefficient computed over the two TEMs will be high. Using the Jaccard coefficient as the similarity measure, I perform group average agglomerative clustering with cluster stopping enabled to discriminate the

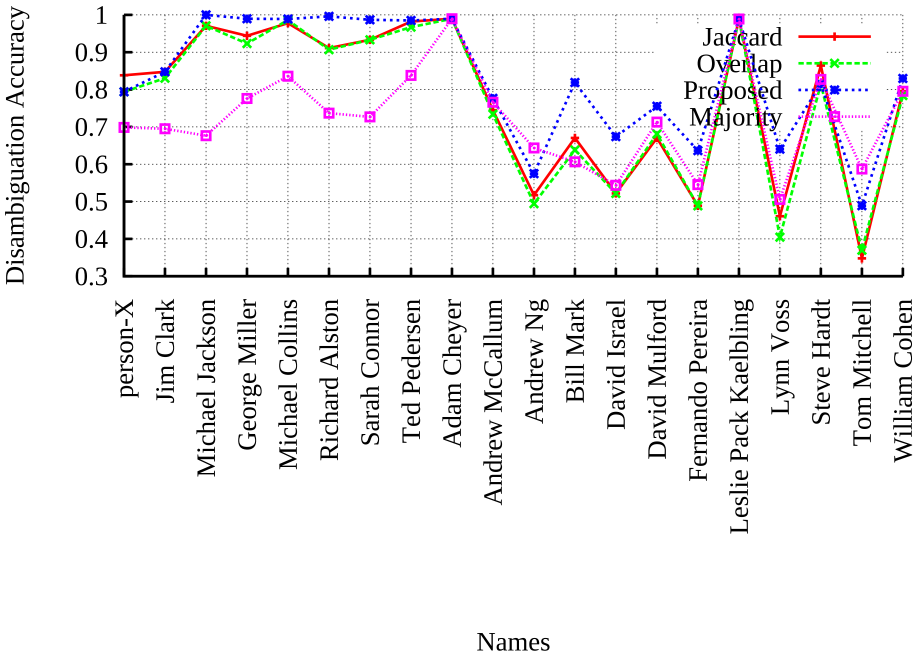


Figure 4.6: Comparing the proposed method against baselines.

namesakes. This baseline shows the effect of the contextual similarity measure (Section 4.2.5) on the proposed method.

Overlap (Overlap coefficient-based clustering): This approach computes the similarity between two TEMs using the overlap (Simpson) coefficient between them. Overlap coefficient between two sets A and B is defined as follows,

$$\text{Overlap} = \frac{|A \cap B|}{\min(|A|, |B|)}. \tag{4.13}$$

If one of the TEMs that we compare contains a lot of elements, then it is likely to share many elements in common with smaller TEMs. Overlap coefficient attempts to normalize the bias due to the difference in size (i.e., number of elements in a TEM) when computing similarity. Using the overlap coefficient as the similarity measure I perform group average agglomerative clustering with cluster stopping enabled to discriminate the namesakes. Overlap coefficient has been used in previous work on social network mining to measure the association between two names on the web [96]. Likewise the **Jaccard** baseline, **Overlap** baseline is expected to show the effect of using contextual similarity measure (Section 4.2.5) on the proposed method.

Proposed: This is the proposed namesake disambiguation algorithm. This approach uses the contextual similarity measure described in Section 4.2.5 to compute the similarity between TEMs. The clustering is performed using group average agglomerate clustering with cluster stopping enabled.

Majority: Majority sense clustering assigns all the contexts in a collection to the person that has the most number of contexts in the collection (dominant sense). Majority sense acts as a baseline for sense disambiguation. In personal name disambiguation on web, although there are lots of people with the same name, only a few are very popular. Assigning all the documents to this popular namesake can still report high clustering accuracies. For this reason, majority sense has been used as a baseline in previous work on name disambiguation [47, 115, 116].

Disambiguation accuracies and the number of correctly identified namesakes (shown within brackets) for the different approaches are reported in Table 4.2. From Table 4.2, we see that the proposed method (**Proposed**) reports the highest disambiguation accuracy of 0.8288. Moreover, all three methods; **Jaccard**, **Overlap** and **Proposed** report significant improvements (pair-wise t-tests with $\alpha = 0.05$) over the **Majority** sense baseline. The proposed method, which uses contextual similarity, outperforms both Jaccard and Overlap baselines because those similarity measures are computed using exact matches between elements. They do not utilize the snippet-based contextual similarity described in section 4.2.5. Therefore, both Jaccard and Overlap baselines suffer from data sparseness (i.e. only few elements appear in common for two TEMs). It is interesting to note that the majority sense baseline has similar or better performance to the proposed method for the names *Adam Cheyer, Leslie Pack Kaelbling, Andrew McCallum, Steve Hardt, Tom Mitchell, Andrew Ng*. All those names appear in the dataset proposed by Bekkerman and McCallum [10]. The datasets for those names are highly skewed and the majority of the documents collected for a name belong to one namesake. For example, in the case of *Adam Cheyer*, 96 out of the total 97 documents are about the founder of Siri Inc. However, the majority sense baseline can only find one namesake and performs poorly when there are more than one popular namesake for an ambiguous personal name.

For collections *person-X, Michael Jackson, Richard Alston, Sarah Connor, George Miller, Michael Collins,Ted Pedersen* all three methods: Jaccard, Overlap, and Proposed, correctly identify all the different namesakes. Correct identification of the number of namesakes is essential because the selection of keywords depends on it. However, Jaccard and Overlap do not perform well with ambiguous names with lots of different namesakes, such as *Tom Mitchell* (37 namesakes), *Andrew Ng* (29 namesakes), *Lynn Voss* (26 namesakes) and *Fernando Pereira* (19 namesakes). In particular, *Tom Mitchell* is a very ambiguous name in the dataset containing 37 namesakes and only 15 out of 92 contexts is for the dominant sense (CMU professor). The quality and the length of text in a document affects the performance of term extraction and named-entity extraction. In particular, the statistical computations in the C-value method depends on the length (i.e. number of words) in a document. Moreover, the named-entity tagger, which is trained using newspaper articles, produces invalid entities when tested on web documents, which are noisy. Better term and entity extraction methods can produce more accurate TEMs, thereby improving overall performance of the proposed method.

Table 4.3 compares the number of clusters produced by the proposed cluster stopping approach against the number of namesakes for a name in the gold standard. For each name in the dataset, Table 4.3 shows the number of namesakes in the gold standard dataset, the number of clusters produced by the proposed method, the number of namesakes correctly detected by the proposed method, and the number of undetected namesakes (i.e. total namesakes in the dataset minus no. of correctly detected namesakes). From Table 4.3 we see that for 11 out of the 20 names in our dataset the cluster stopping approach produces exactly the same number of clusters as the number of namesakes. Moreover, for 6 of those names, all namesakes are accurately detected. In particular, for Pedersen and Kulkarni's dataset, I have perfectly detected all namesakes.

Table 4.4 compares the proposed method against the best F-scores reported by Pedersen and Kulkarni [115] for the names in their dataset. Although there are differences in definitions of the evaluation metrics, both F-score and disambiguation accuracy compare a clustering produced by a system against a gold-standard. From Table 4.4 we can see that

Table 4.3: The number of namesakes detected by the proposed method

Name	no. of namesakes	no. of clusters	detected	undetected
person-X	4	4	4	0
Jim Clark	8	8	3	5
Michael Jackson	2	2	2	0
Pedersen and Kulkarni's dataset [116, 115]				
George Miller	3	3	3	0
Michael Collins	4	4	4	0
Richard Alston	2	2	2	0
Sarah Connor	2	2	2	0
Ted Pedersen	4	4	4	0
Bekkerman and McCallum's dataset [10]				
Adam Cheyer	2	2	1	1
Andrew McCallum	8	8	4	4
Andrew Ng	29	8	5	24
Bill Mark	8	4	4	4
David Israel	16	7	4	12
David Mulford	13	10	4	9
Fernando Pereira	19	13	6	13
Leslie Pack Kaelbling	2	2	1	1
Lynn Voss	26	12	9	17
Steve Hardt	6	5	2	4
Tom Mitchell	37	17	13	24
William Cohen	10	5	3	7

the proposed method performs better than the method proposed by Pedersen and Kulkarni [115].

In Table 4.5, I compare the proposed method against the previous work on namesake disambiguation by Bekkerman and McCallum [10]. Bekkerman and McCallum consider the namesake disambiguation problem as a one of separating a set of given documents collected for an ambiguous personal name into two clusters: a cluster with all documents relevant to a particular namesake of the given name, and a cluster with all other documents. I compute disambiguation accuracy for those two clusters using Formula 4.11 for each name as shown in Table 4.5. However, it must be emphasized that their method can find *only one namesake* of the given ambiguous personal name. Moreover, they assume the

Table 4.4: Comparison with results reported by Pedersen and Kulkarni [115]

Name	Pedersen and Kulkarni (F-score)	Proposed (disambiguation accuracy)
George Miller	0.7587	0.9895
Michael Collins	0.9304	0.9889
Richard Alston	0.9960	0.9960
Sara Connor	0.9000	0.9867
Ted Pedersen	0.7658	0.9850

Table 4.5: Comparison with results reported by Bekkerman and McCallum [10] using disambiguation accuracy

Name	Bekkerman and McCallum [10]	Proposed
Adam Cheyer	0.6495	0.9897
Andrew McCallum	0.9787	0.7766
Andrew Ng	0.9080	0.5747
Bill Mark	0.8511	0.8191
David Israel	0.9456	0.6739
David Mulford	1.0000	0.7553
Fernando Pereira	0.7159	0.6364
Leslie Pack Kaelbling	0.9438	0.98888
Lynn Voss	0.9888	0.6404
Steve Hardt	0.3827	0.8148
Tom Mitchell	0.9348	0.4891
William Cohen	0.9545	0.8295

availability of information regarding the social network of the person that they attempt to disambiguate. In contrast, the proposed method attempts to disambiguate *all namesakes* of a given personal name and does not require any information regarding the social network of a person. Despite the fact that the proposed method does not require external information regarding a namesake, such as his or her social network, and attempts to identify all namesakes, it has comparative performance with Bekkerman and McCallum's method.

Tables 4.6 and 4.7 shows the top ranking keywords extracted for Michael Jackson and Jim Clark. First cluster for Michael Jackson represents the singer while the second cluster stands for the expert on beer. The two Michael Jacksons are annotated with very different

Table 4.6: Clusters for Michael Jackson

CLUSTER 1	CLUSTER 2
fan club	beer hunter
trial	ultimate beer FAQ
world network	christmas beer
superstar	great beer
new charity song	pilsner beer
neverland ranch	bavaria

Table 4.7: Clusters for Jim Clark

CLUSTER 1	CLUSTER 2
racing driver	entrepreneur
rally	story
scotsman	silicon valley
driving genius	CEO
scottish automobile racer	silicon graphics
british rally news	SGI/Netscape

TEMs. Jim Clark the Formula one champion is represented by the first cluster in Table 4.7, whereas the second cluster stands for the founder of Netscape.

4.4.6 Information Retrieval Task

I conduct an information retrieval task to evaluate the ability of the extracted keywords to uniquely identify an individual. I use a keyword k, selected for a namesake p, of an ambiguous name n to retrieve documents that contain k from a collection of documents D downloaded from the web using n as the query. If a document $d \in D$ contains the keyword k then I retrieve the document. I use the top 5 ranked keywords selected by the proposed method for all the namesakes it identifies for the 19 names in our gold standard datasets shown in Table 4.1 (person-X is not considered in this evaluation because it is not an actual name). If a keyword can accurately retrieve documents regarding a particular namesake, then such keywords are useful when searching for that person. I measure the ability of a keyword to retrieve documents related to a particular namesake using precision, recall and

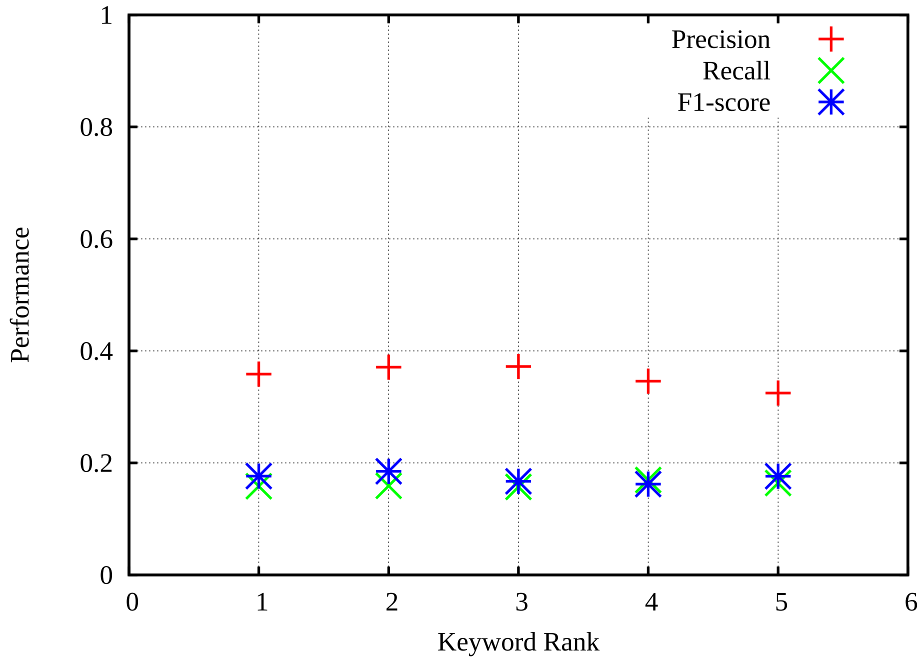


Figure 4.7: Performance vs. the keyword rank

F1-score. The precision of a keyword k, precision(k), is defined as follows,

$$\text{precision}(k) = \frac{\text{no. of documents that contain } k\text{, and belongs to } p}{\text{no. of documents that contain } k}. \tag{4.14}$$

Likewise, recall of a keyword k, recall(k), is defined as follows,

$$\text{recall}(k) = \frac{\text{no. of documents that contain } k\text{, and belongs to } p}{\text{no. of documents that belong to } p}. \tag{4.15}$$

The F1-score of a keyword k, F1-score(k), can then be computed as follows,

$$\text{F1-score}(k) = \frac{2 \times \text{precision}(k) \times \text{recall}(k)}{\text{precision}(k) + \text{recall}(k)}. \tag{4.16}$$

For the top 5 ranked keywords extracted for each detected namesake by the proposed method, I compute their precision, recall and F1-score using the above mentioned equations

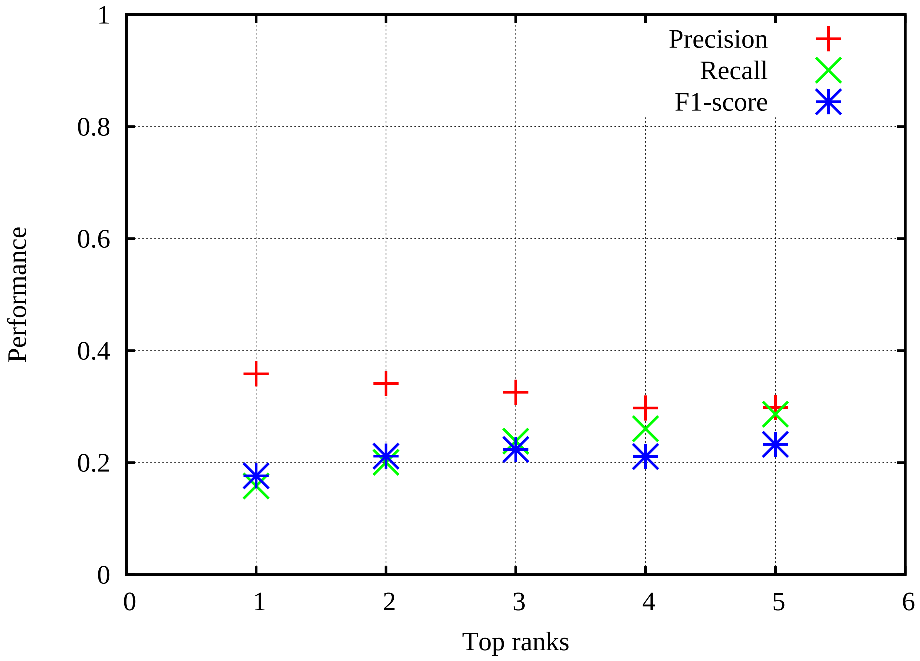


Figure 4.8: Performance vs. combinations of top ranking keywords

and take the average over the 19 names selected from the gold standard. Experimental results are shown in Figure 4.7. From Figure 4.7, we see that using any one of the top ranking keywords, on average, we obtain a precision of around 0.38. Moreover, a slight decrease in precision can be observed with the rank of the keywords used. However, the low recall (therefore the F1-score) indicates that using a single keyword alone is not sufficient to retrieve all the documents related to a namesake. A combination of top ranking keywords can be useful to improve recall. To evaluate the effect of using multiple keywords on retrieval performance, I combined the top r ranked keywords in a disjunctive (OR) query. Specifically, I retrieve a document for a namesake, if any one of the top k ranked keywords selected for that namesake appears in that document. I experiment with top 1-5 ranks as shown in Figure 4.8. From Figure 4.8 we see that combining the top ranked keywords indeed improve the recall. Although a slight drop in precision can be seen when we combine lower ranked keywords, overall, the F1-score improves as a result of the gain in recall.

As a specific example of how the keywords extracted by the proposed method can

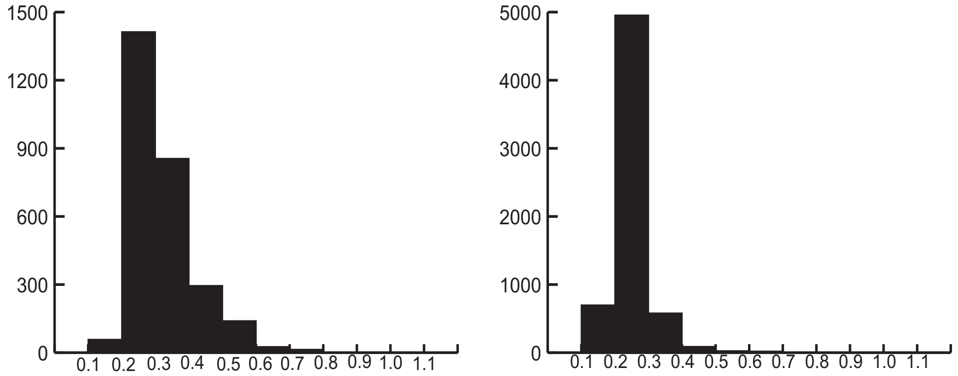
Table 4.8: Effectiveness of the extracted keywords to identify an individual on the web.

Keyword	person-1	person-2	others	Hits
NONE	41	26	33	1,080,000
racing driver	81	1	18	22,500
rally	42	0	58	82,200
scotsman	67	0	33	16,500
entrepreneur	1	74	25	28,000
story	17	53	30	186,000
silicon valley	0	81	19	46,800

be used in a real-world web search scenario, I search Google for the namesakes of the ambiguous personal name *Jim Clark* using the extracted keywords. I first search Google only using the name *Jim Clark*. I then modify the query by including a keyword selected for a particular namesake. I manually check the top 100 ranked search results and determine how many results are relevant for the namesake that I am searching for. Experimental results are summarized in Table 4.8.

In Table 4.8 I classify Google search results into three categories. “person-1” is the formula one racing world champion, “person-2” is the founder of Netscape, and “other” category contains remainder of the pages that I could not classify to previous two groups (some of these pages were on other namesakes, and some were not sufficiently detailed to properly classify). I first search in Google without adding any keywords to the ambiguous name. Including the keywords *rally* and *scotsman*, which are selected from the cluster for *Jim Clark* the formula one champion, return no results for the other popular namesake. Likewise, the keywords *entrepreneur* and *silicon valley* yield results largely for the founder of Netscape. However, the keyword *story* returns results for both namesakes. A close investigation revealed that, the keyword *story* is extracted from the title of the book “The New New Thing: A Silicon Valley Story”, a book on the founder of Netscape.

On a Intel Core2Duo 2.8GHz, 2GB RAM desktop, the proposed method requires approximately one minute to disambiguate a given name. The major portion of time is spent on querying a web search engine to compute contextual similarity. I cache the search results to reduce the amount of web accesses. The proposed method is used to disambiguate people in a social network system with more than $200,000$ people [95].



(a) Within-class similarity distribution in ”person-X” dataset

(b) Cross-class similarity distribution in ”person-X” dataset

Figure 4.9: The histogram of within-class and cross-class similarity distributions in ”person-X” dataset. X axis represents the similarity value. Y axis represents the number of document pairs from the same class (within-class) or from different classes (cross-class) that have the corresponding similarity value.

4.4.7 Evaluating Contextual Similarity

In section 4.2.5, I defined the similarity between documents (i.e., term-entity models created from the documents) using a web snippets based contextual similarity (Formula 4.6). However, how well such a metric represents the similarity between documents, remains unknown. Therefore, to evaluate the contextual similarity among documents, I group the documents in ”person-X” dataset into four classes (each class representing a different person) and use Formula 4.7 to compute within-class and cross-class similarity histograms, as illustrated in Figure 4.9.

Ideally, within-class similarity distribution should have a peak around 1 and cross-class similarity distribution around 0, whereas both histograms in Figure 4.9(a) and 4.9(b) have their peaks around 0.2. However, within-class similarity distribution is heavily biased toward to the right of this peak, and cross-class similarity distribution to the left. Moreover, there are no document pairs with more than 0.5 cross-class similarity. The experimental results guarantees the validity of the contextual similarity metric.

Chapter 5

Name Alias Detection

Precisely identifying entities in web documents is necessary for various tasks such as relation extraction [96], search and integration of data [55] and entity disambiguation [91]. Nevertheless, identification of entities on the web is difficult for two fundamental reasons: first, different entities can share the same name (lexical ambiguity); secondly, a single entity can be designated by multiple names (referential ambiguity). As an example of lexical ambiguity the name *Jim Clark* is illustrative. Aside from the two most popular namesakes, the formula-one racing champion and the founder of Netscape, at least 10 different people are listed among the top 100 results returned by Google for the name. On the other hand, referential ambiguity occurs because people use different names to refer to the same entity on the web. For example, the American movie star *Will Smith* is often called the *the Fresh Prince* in web contents. Although lexical ambiguity, particularly ambiguity related to personal names, has been explored extensively in the previous studies of name disambiguation [91, 10], the problem of referential ambiguity of entities on the web has received much less attention. In this Chapter, I specifically examine on the problem of automatically extracting the various references on the web to a particular entity.

For an entity e, I define the set A of its aliases to be the set of all words or multi-word expressions that are used to refer to e on the web. For example, *Godzilla* is a one-word alias for *Hideki Matsui*, whereas the alias *the Fresh Prince* contains three words and refers to *Will Smith*. Various types of terms are used as aliases on the web. For instance, in the case of an actor, the name of a role or the title of a drama (or a movie) can later become an alias for the person (e.g., *Fresh Prince, Knight Rider*). Titles or professions such as *president*, *doctor*, *professor*, etc. are also frequently used as aliases. Variants or abbreviations of names such as *Bill* for *William* and acronyms such as *J.F.K.* for *John Fitzgerald Kennedy* are also types of name aliases that are observed frequently on the web.

Identifying aliases of a name is important for extracting relations among entities. For example, Matsuo et al. [96] propose a social network extraction algorithm, in which they compute the strength of the relation between two individuals *A* and *B* by the web hits for the conjunctive query, *"A" AND "B"*. However, both persons *A* and *B* might also appear in their alias names in web contents. Consequently, by expanding the conjunctive query using aliases for the names, a social network extraction algorithm can accurately compute the strength of a relationship between two persons.

Searching for information about people on the web is an extremely common activity of Internet users. Around 30% of search engine queries include personal names [2]. However, retrieving information about a person merely using his or her real names is insufficient when that person has nicknames. Particularly with keyword-based search engines, we will only retrieve pages which use the real name to refer to the person about whom we are interested in finding information. In such cases, automatically extracted aliases of the name are useful to expand a query in a web search, thereby improving recall.

The Semantic Web is intended to solve the entity disambiguation problem by providing a mechanism to add semantic metadata for entities. However, an issue that the Semantic Web currently faces is that insufficient semantically annotated web contents are available. Automatic extraction of metadata [31] can accelerate the process of semantic annotation. For named entities, automatically extracted aliases can serve as a useful source of metadata, thereby providing a means to disambiguate an entity.

Along with the recent rapid growth of social media such as blogs, extracting and classifying sentiment on the web has received much attention [144]. Typically, a sentiment analysis system classifies a text as positive or negative according to the sentiment expressed in it. However, when people express their views about a particular entity, they do so by referring to the entity not only using the real name but also using various aliases of the name. By aggregating texts that use various aliases to refer to an entity, a sentiment analysis system can produce an informed judgment related to the sentiment.

I propose a fully automatic method to discover aliases of a given personal name from the web. Our contributions can be summarized as follows.

- I propose a lexical pattern-based approach to extract aliases of a given name using snippets returned by a web search engine. The lexical patterns are generated automatically using a set of real-world name-alias data. I evaluate the confidence of extracted lexical patterns and retain the patterns that can accurately discover aliases for various personal names. Our pattern generation algorithm does not assume any language specific pre-processing such as part-of-speech tagging or dependency parsing etc., which can be both inaccurate and computationally costly in web-scale data processing.

- To select the best aliases among the extracted candidates, I propose numerous ranking scores based upon three approaches: lexical pattern frequency, word-cooccurrences in an anchor text graph, and page-counts on the web. Moreover, using real world name alias data, I train a ranking support vector machine to learn the optimal combination of individual ranking scores to construct a robust alias extraction method.

- I conduct a series of experiments to evaluate the various components of the proposed method. I compare the proposed method against numerous baselines and previously proposed name alias extraction methods on three datasets: an English personal names dataset, an English place names dataset, and a Japanese personal names dataset. Moreover, I evaluate the aliases extracted by the proposed method in an information retrieval task and a relation extraction task.

5.1 Related work

The problem of extracting aliases of a given name can be considered as a special case of the more general problem of extracting the words Y that have a given relation R with a word X. For example, extracting hyponyms [60], synonyms [86], meronyms [12] are specific instances of this general problem of relation extraction. Manually created or automatically extracted lexico-syntactic patterns have been successfully used to identify various relations between words [124, 140]. For example, patterns such as *X is a Y* and *X such as Y* are typically used to introduce hypernyms, whereas, *X of a Y* and *X's Y* are frequently used with meronyms. However, alias extraction poses several unique challenges that separates it from the more general relation extraction problem. Firstly, personal names and their aliases are not typically listed in manually created dictionaries. Therefore, an alias extraction algorithm must first extract a possible set of candidate aliases for a given name and then verify each extracted candidate. Secondly, names and aliases can be multi-word expressions. For example, in the case of *Will Smith*, who has a two-word alias *fresh prince*, it is inaccurate to extract *fresh* as an alias. Thirdly, unlike hypernyms or meronyms, it is not obvious as to which lexical patterns convey useful clues related to aliases of a given name. This makes it difficult to manually create a sufficiently large list of lexical patterns to cover various types of name aliases. In addition to above mentioned challenges, the lack of evaluation benchmark dataset for aliases makes it difficult to compare and evaluate different approaches. Although it is relatively easy to manually verify whether an extracted candidate is a correct alias of a given name, it is not always possible to obtain a list of all the aliases of a name, which makes it difficult to compute the recall or coverage of an alias extraction algorithm.

Alias identification is closely related to the problem of cross-document coreference resolution, in which the objective is to determine whether two mentions of a name in different documents refer to the same entity. Bagga and Baldwin [6] proposed a cross-document coreference resolution algorithm by first performing within-document coreference resolution for each individual document to extract coreference chains, and then clustering the coreference chains under a vector space model to identify all mentions of a name in the document set. However, the vastly numerous documents on the web render it impractical to perform within-document coreference resolution to each document separately and then cluster the documents to find aliases.

In personal name disambiguation the goal is to disambiguate various people that share the same name (*namesakes*) [91, 10]. Given an ambiguous name, most name disambiguation algorithms have modeled the problem as one of document clustering, in which all documents that discuss a particular individual of the given ambiguous name are grouped into a single cluster. The web people search task (WEPS) at SemEval 2007 [1] provided

[1] http://nlp.uned.es/weps

a dataset and evaluated various name disambiguation systems. However, the name disambiguation problem differs fundamentally from that of alias extraction because, in name disambiguation the objective is to identify the different entities that are referred by the same ambiguous name; in alias extraction, we are interested in extracting all references to a single entity from the web.

Approximate string matching algorithms have been used for extracting variants or abbreviations of personal names (e.g. matching *Will Smith* with the first name initialized variant *W. Smith*) [49]. Rules in the form of regular expressions and edit-distance-based methods have been used to compare names. Bilenko and Mooney [15] proposed a method to learn a string similarity measure to detect duplicates in bibliography databases. However, an inherent limitation of such string matching approaches is that they cannot identify aliases which share no words or letters with the real name. For example, approximate string matching methods would not identify *Fresh Prince* as an alias for *Will Smith*.

Hokama and Kitagawa [63] propose an alias extraction method that is specific to the Japanese language. For a given name p, they search for the query *"* koto p"* and extract the context that matches the asterisk. The Japanese word *koto*, roughly corresponds to *also known as* in English. However, *koto* is a highly ambiguous word in Japanese that can also mean *incident, thing, matter, experience* and *task*. As reported in their paper, many noisy and incorrect aliases are extracted using this pattern, which requires various post-processing heuristics that are specific to Japanese language to filter-out the incorrect aliases. Moreover, manually crafted patterns do not cover various ways that convey information about name aliases. In contrast, I propose a method to leverage such lexical patterns automatically using a training dataset of names and aliases.

5.2 Method

The proposed method is outlined in Figure 5.1 and comprises two main components: pattern extraction, and alias extraction and ranking. Using a seed list of name-alias pairs, I first extract lexical patterns that are frequently used to convey information related to aliases on the web. The extracted patterns are then used to find candidate aliases for a given name.

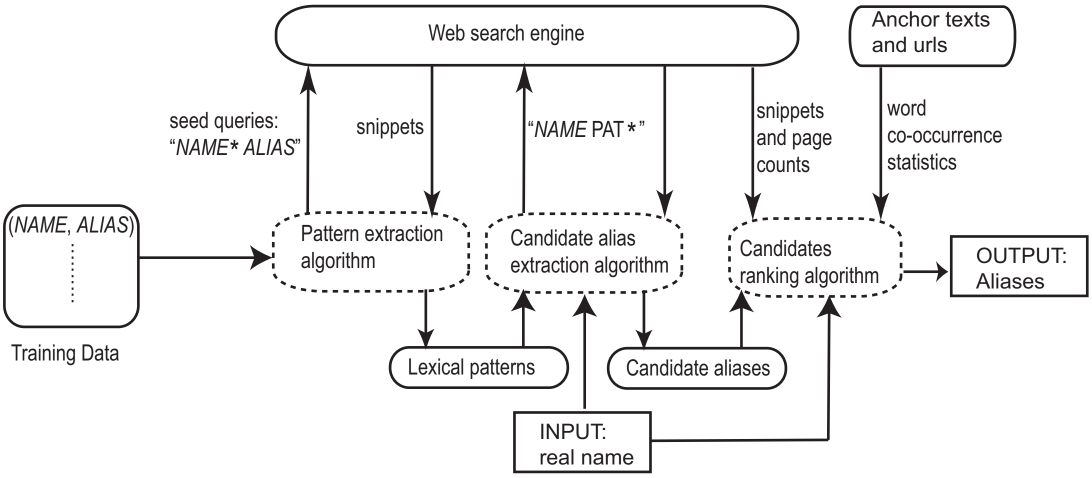


Figure 5.1: Outline of the proposed method

.

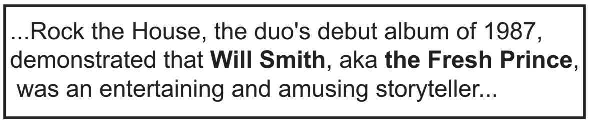

...Rock the House, the duo's debut album of 1987, demonstrated that **Will Smith**, aka **the Fresh Prince**, was an entertaining and amusing storyteller...

Figure 5.2: A snippet returned for the query *"Will Smith * The Fresh Prince"* by Google

I define various ranking scores using the hyperlink structure on the web and page counts retrieved from a search engine to identify the correct aliases among the extracted candidates.

5.2.1 Extracting Lexical Patterns from Snippets

Many modern search engines provide a brief text snippet for each search result by selecting the text that appears in the web page in the proximity of the query. Such snippets provide valuable information related to the local context of the query. For names and aliases, snippets convey useful semantic clues that can be used to extract lexical patterns that are frequently used to express aliases of a name. For example, consider the snippet returned by Google[2] for the query *"Will Smith * The Fresh Prince"*.

Here, I use the wildcard operator * to perform a *NEAR* query and it matches with one or more words in a snippet. In Figure 5.2 the snippet contains *aka* (i.e. *also known as*),

[2] www.google.com

which indicates the fact that *fresh prince* is an alias for *Will Smith*. In addition to *a.k.a.*, numerous clues exist such as *nicknamed, alias, real name is, nee*, which are used on the web to represent aliases of a name. Consequently, I propose the shallow pattern extraction method illustrated in Algorithm 5 to capture the various ways in which information about aliases of names is expressed on the web. Lexico-syntactic patterns have been used in numerous related tasks such as extracting hypernyms [60] and meronyms [12].

Algorithm 5 ExtractPatterns(S)

```
Input: set $S$ of (NAME, ALIAS) pairs
Output: set $P$ of patterns
$P \leftarrow$ null
for $(NAME, ALIS) \in S$ do
    $D \leftarrow \mathrm{GetSnippets}(``NAME\ *\ ALIAS'')$
    for snippet $d \in D$ do
        $P \leftarrow P + \mathrm{CreatePattern}(d)$
    end for
end for
return $P$
```

Algorithm 6 ExtractCandidates(NAME,P)

```
Input: set of patterns $P$, real name $NAME$
Output: set $C$ of candidates
$C \leftarrow$ null
for pattern $p \in P$ do
    $D \leftarrow \mathrm{GetSnippets}(``NAME\ p\ *'')$
    for snippet $d \in D$ do
        $C \leftarrow C + \mathrm{GetNgrams}(d, NAME, p)$
    end for
end for
return $C$
```

Given a set S of (NAME, ALIAS) pairs, the function *ExtractPatterns* returns a list of lexical patterns that frequently connect names and their aliases in web-snippets. For each (NAME, ALIAS) pair in S, the *GetSnippets* function downloads snippets from a web search engine for the query *"NAME * ALIAS"*. Then, from each snippet, the *CreatePattern* function extracts the sequence of words that appear between the name and the alias.

Results of our preliminary experiments demonstrated that consideration of words that fall outside the name and the alias in snippets did not improve performance. Finally, the real name and the alias in the snippet are respectively replaced by two variables **[NAME]** and **[ALIAS]** to create patterns. For example, from the snippet shown in Figure 5.2, I extract the pattern **[NAME]** *aka* **[ALIAS]**. I repeat the process described above for the reversed query, *"ALIAS * NAME"* to extract patterns in which the alias precedes the name.

Once a set of lexical patterns is extracted, I use the patterns to extract candidate aliases for a given name as portrayed in Algorithm 6. Given a name, $NAME$ and a set, P of lexical patterns, the function *ExtractCandidates* returns a list of candidate aliases for the name. I associate the given name with each pattern, p in the set of patterns, P and produce queries of the form: *"NAME p *"*. Then the *GetSnippets* function downloads a set of snippets for the query. Finally, the *GetNgrams* function extracts continuous sequences of words (n-grams) from the beginning of the part that matches the wildcard operator $*$. Experimentally, I select up to 5-grams as candidate aliases. Moreover, I remove candidates that contain only stop words such as *a, an*, and *the*. For example, assuming that we retrieved the snippet in Algorithm 5 for the query *"Will Smith aka *"*, the procedure described above extracts *the fresh* and *the fresh prince* as candidate aliases.

5.2.2 Ranking of Candidates

Considering the noise in web-snippets, candidates extracted by the shallow lexical patterns might include some invalid aliases. From among these candidates, we must identify those which are most likely to be correct aliases of a given name. I model this problem of alias recognition as one of ranking candidates with respect to a given name such that the candidates which are most likely to be correct aliases are assigned a higher rank. First, I define various ranking scores to measure the association between a name and a candidate alias using three different approaches: lexical pattern frequency, word co-occurrences in an anchor text graph, and page-counts on the web. Next, I describe those three approaches in detail.

5.2.3 Lexical Pattern Frequency

In Section 5.2.1 I presented an algorithm to extract numerous lexical patterns that are used to describe aliases of a personal name. As we will see later in Section 2.3, the proposed pattern extraction algorithm can extract a large number of lexical patterns. If the personal name under consideration and a candidate alias occur in many lexical patterns, then it can be considered as a good alias for the personal name. Consequently, I rank a set of candidate aliases in the descending order of the number of different lexical patterns in which they appear with a name. The lexical pattern frequency of an alias is analogous to the document frequency (DF) popularly used in information retrieval.

5.2.4 Co-occurrences in Anchor Texts

Anchor texts have been studied extensively in information retrieval and have been used in various tasks such as synonym extraction, query translation in cross-language information retrieval, and ranking and classification of web pages [25]. Anchor texts are particularly attractive because they not only contain concise texts, but also provide links that can be considered as expressing a citation. I revisit anchor texts to measure the association between a name and its aliases on the web. Anchor texts pointing to a url provide useful semantic clues related to the resource represented by the url. For example, if the majority of *inbound anchor texts* of a url contain a personal name, it is likely that the remainder of the inbound anchor texts contain information about aliases of the name. Here, I use the term inbound anchor texts to refer the set of anchor texts pointing to the same url.

I define a name p and a candidate alias x as *co-occurring*, if p and x appear in two *different* inbound anchor texts of a url u. Moreover, I define *co-occurrence frequency* (**CF**) as the number of different urls in which they co-occur. It is noteworthy that I do not consider co-occurrences of an alias and a name in the same anchor text. For example, consider the picture of Will Smith shown in Figure 5.3. Figure 5.3 shows a picture of Will Smith being linked to by four different anchor texts. According to our definition of co-occurrence, *Will Smith* and *fresh prince* are considered as co-occurring. Similarly, *Will Smith* and *Hancock* is also co-occurring in this example. It is noteworthy that we do not require the resource that is linked to by an anchor text to be a picture (or any resource related to the person under

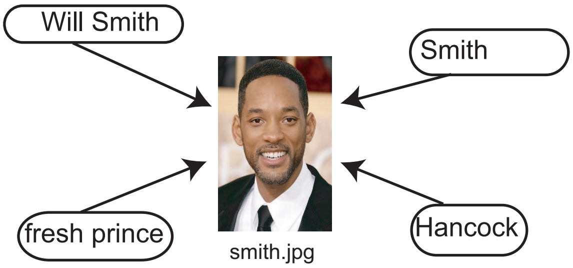


Figure 5.3: A picture of Will Smith being linked by different anchor texts on the web.

	x	$C - \{x\}$	C
p	k	$n - k$	n
$V - \{p\}$	$K - k$	$N - n - K + k$	$N - n$
V	K	$N - K$	N

Table 5.1: Contingency table for a candidate alias x

consideration for that matter). p and x are considered as co-occurring as long as they are linked to the same url, irrespective of the resource. Moreover, an anchor text can contain other texts beside a real name or an alias. I consider all the words in an anchor text and their bi-grams as potential candidate aliases if they co-occur with the real name according to our definition.

We can use this definition to create a contingency table like the one shown in Table 5.1. Therein, C is the set of candidates extracted by the algorithm described in Algorithm 6, V is the set of all words that appear in anchor texts, $C - \{x\}$ and $V - \{p\}$ respectively denote all candidates except x and all words except the given name p, k is the co-occurrence frequency between x and p. Moreover, K is the sum of co-occurrence frequencies between x and all words in V, whereas n is the same between p and all candidates in C. N is the total co-occurrences between all word pairs taken from C and V.

To measure the strength of association between a name and a candidate alias, using Table 5.1 I define nine popular co-occurrence statistics: co-occurrence frequency (**CF**), tfidf measure (**tfidf**), chi-squared measure (**CS**), Log-likelihood ratio (**LLR**), hyper-geometric distributions (**HG**), cosine measure (**cosine**), overlap measure (**overlap**), and Dice coefficient (**Dice**). Next, I describe the computation of those association measures in detail.

Co-occurrence Frequency

This is the simplest of all association measures and was defined already in the previous section. The value k in Table 5.1 denotes the co-occurrence frequency of a candidate alias x and a name p. Intuitively, if there are many urls which are pointed to by anchor texts that contain a candidate alias x and a name p, then it is an indication that x is indeed a correct alias of the name p.

tfidf

The co-occurrence frequency is biased towards highly frequent words. A word that has a high frequency in anchor texts can also report a high co-occurrence with the name. For example, so-called *stop words* such as particles and articles appear in various anchor texts and have an overall high frequency. The *tfidf* measure [133], which is popularly used in information retrieval, is useful to normalize this bias. In fact, the tfidf measure reduces the weight that is assigned to words that appear across various anchor texts. The tfidf score of a candidate x as an alias of name p, $\text{tfidf}(p, x)$, is computed from Table 5.1 as

$$tfidf(p, x) = k \log \frac{N}{K+1}. \tag{5.1}$$

Chi-square Measure (CS)

The χ^2 measure has been used as a test for dependence between two words in various natural language processing tasks including collocation detection, identification of translation pairs in aligned corpora, and measuring corpus similarity [92]. Given a contingency table resembling that shown in Table 5.1, the χ^2 measure compares the observed frequencies in the table with the frequencies expected for independence. Then it is likely that the candidate x is an alias of the name p if the difference between the observed and expected frequencies is large for some candidate x.

The χ^2 measure sums the differences between observed and expected values in the contingency table and is scaled by the magnitude of the expected values. Actually, the χ^2

measure is given as

$$\chi^2 = \sum_{i,j} \frac{(O_{ij} - E_{ij})^2}{E_{ij}}. \tag{5.2}$$

Here, O_{ij} and E_{ij} respectively denote the observed and expected frequencies.

I apply Formula 5.2 to Table 5.1 and define χ^2 ranking score, $\mathrm{CS}(p, x)$, of a candidate x as an alias of a name p as follows:

$$\mathrm{CS}(p, x) = \frac{N\{k(N - K - n + k) - (n - k)(K - k)\}^2}{nK(N - K)(N - n)}. \tag{5.3}$$

The CS score is used to rank candidate aliases of a name.

Log Likelihood Ratio (LLR)

The log likelihood ratio (LLR) [42] is defined as the ratio between the likelihoods of two alternative hypotheses: that the name p and the candidate alias x are independent or the name p and that the candidate alias x are dependent. Likelihood ratios have been used often in collocation discovery [92]. Unlike the χ^2 measure, likelihood ratios are robust against sparse data and have a more intuitive definition.

The LLR-based alias ranking score, $\mathrm{LLR}(p, x)$, is computed using values in Table 5.1 as

$$\begin{aligned} \mathrm{LLR}(p, x) &= k \log \frac{kN}{nK} + (n - k) \log \frac{(n - k)N}{n(N - K)} \\ &+ (K - k) \log \frac{N(K - k)}{K(N - n)} \\ &+ (N - K - n + k) \log \frac{N(N - K - n + k)}{(N - K)(N - n)}. \end{aligned} \tag{5.4}$$

Pointwise Mutual Information (PMI)

Pointwise mutual information [28] is a measure that is motivated by information theory; it is intended to reflect the dependence between two probabilistic events. For values of two random variables y and z, their pointwise mutual information is defined as

$$\mathrm{PMI}(y,z) = \log_2 \frac{P(y,z)}{P(y)P(z)}. \tag{5.5}$$

Here, $P(y)$ and $P(z)$ respectively denote the probability of random variables y and z. Consequently, $P(y,z)$ is the joint probability of y and z. The probabilities in Formula 5.5 can be computed as marginal probabilities from Table 5.1 as

$$\begin{aligned} \mathrm{PMI}(p,x) &= \log_2 \frac{\frac{k}{N}}{\frac{K}{N}\frac{n}{N}} \qquad (5.6)\\ &= \log_2 \frac{kN}{Kn}. \end{aligned}$$

Hypergeometric Distribution

Hypergeometric distribution [62] is a discrete probability distribution that describes the number of successes in a sequence of draws from a finite population without replacement. For example, the probability of the event that "k red balls are contained among n balls which are arbitrarily chosen from among N balls containing K red balls", is given by the hypergeometric distribution, $hg(N,K,n,k)$, as

$$hg(N,K,n,k) = \frac{\binom{K}{l}\binom{N-k}{n-l}}{\binom{N}{n}}. \tag{5.7}$$

I apply the definition 5.7 of hypergeometric distribution to the values in Table 5.1 and compute the probability $HG(p,x)$ of observing more than k number of co-occurrences of the name p and candidate alias x. The value of $HG(p,x)$ is give by,

$$\begin{gathered} \mathrm{HG}(p,x) = -\log_2\Big(\sum_{l \geq k} hg(N,K,n,l)\Big) \qquad (5.8)\\ \max\{0, N+K-n\} \geq l \geq \min\{n,K\}. \end{gathered}$$

The value $HG(p,x)$ indicates the significance of co-occurrences between p and x. I use $HG(p,x)$ to rank candidate aliases of a name.

Cosine Measure

The cosine measure is widely used to compute the association between words. For strength of association between elements in two sets, X and Y can be computed using the cosine measure:

$$\text{cosine}(X, Y) = \frac{|X \cap Y|}{\sqrt{|X|}\sqrt{|Y|}}. \tag{5.9}$$

Here, $|X|$ denotes the number of elements in set X.

Letting X be the occurrences of candidate alias x and Y the occurrences of name p, I define $\text{cosine}(p, x)$ as a measure of association between a name and a candidate alias as

$$\text{cosine}(p, x) = \frac{k}{\sqrt{n} + \sqrt{K}}. \tag{5.10}$$

Overlap Measure

The overlap between two sets X and Y is defined as

$$\text{overlap}(X, Y) = \frac{|X \cap Y|}{\min(|X|, |Y|)}. \tag{5.11}$$

Assuming that X and Y respectively represent occurrences of name p and candidate alias x, I define a ranking score based on the overlap measure to evaluate the appropriateness of a candidate alias.

$$\text{overlap}(p, x) = \frac{k}{\min(n, K)} \tag{5.12}$$

Dice Coefficient

Smadja [138] proposes the use of the Dice coefficient to retrieve collocations from large textual corpora. The Dice coefficient is defined over two sets X and Y as

$$\text{Dice}(X, Y) = \frac{2|X \cap Y|}{|X| + |Y|}. \tag{5.13}$$

As with cosine and overlap measures, using the co-occurrence values in Table 5.1, I

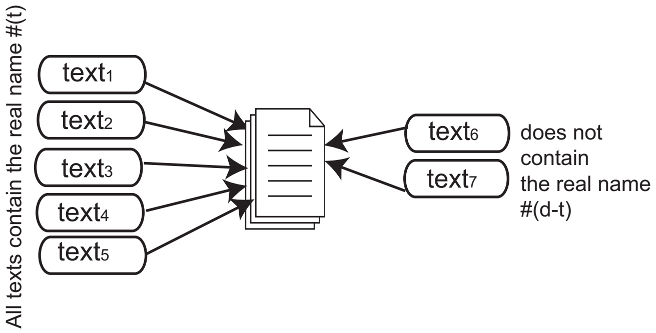

All texts contain the real name #(t)
text1
text2
text3
text4
text5
text6
text7
does not contain the real name #(d-t)

Figure 5.4: Discounting the co-occurrences in hubs.

define a ranking score based on the Dice coefficient as

$$\mathrm{Dice}(p, x) = \frac{2k}{n + K}.$$

The Jaccard coefficient, which is monotonic with the Dice coefficient, was not considered because it gives the exact ranking of nodes as that given by the Dice coefficient. In general, if the Dice coefficient is g, then the corresponding Jaccard coefficient is given by $g/(2 - g)$.

5.2.5 Hub discounting

A frequently observed phenomenon related to the web is that many pages with diverse topics link to so-called *hubs* such as Google, Yahoo, or MSN. Two anchor texts might link to a hub for entirely different reasons. Therefore, co-occurrences coming from hubs are prone to noise. Consider the situation shown in Figure 5.4 where a certain web page is linked to by two sets of anchor texts. One set of anchor texts contains the real name for which we must find aliases, whereas the other set of anchor texts contains various candidate aliases. If the majority of anchor texts linked to a particular web site use the real name to do so, then the confidence of that page as a source of information regarding the person whom we are interested in extracting aliases increases. I use this intuition to compute a simple discouting measure for co-occurrences in hubs as follows.

To overcome the adverse effects of a hub h when computing co-occurrence measures, I multiply the number of co-occurrences of words linked to h by a factor $\alpha(h,p)$, where

$$\alpha(h,p) = \frac{t}{d}. \tag{5.14}$$

Here, t is the number of inbound anchor texts of h that contain the real name p, and d is the total number of inbound anchor texts of h. If many anchor texts that link to h contain p (i.e. larger t value), then the reliability of h as a source of information about p increases. On the other hand, if h has many inbound links (i.e. larger d value), then it is likely to be a noisy hub and gets discounted when multiplied by $\alpha(<< 1)$. Intuitively, Formula 5.14 boosts hubs that are likely to contain information related to p, while penalizing those that contain various other topics.

5.2.6 Page-count-based Association Measures

In section 5.2.4 we defined various ranking scores using anchor texts. However, not all names and aliases are equally well represented in anchor texts. Consequently, in this section, I define word association measures that consider co-occurrences not only in anchor texts but in the web overall. Page counts retrieved from a web search engine for the conjunctive query, *"p AND x"*, for a name p and a candidate alias x can be regarded as an approximation of their co-occurrences in the web. I compute popular word association measures using page counts returned by a search engine.

WebDice

I compute the Dice coefficient, $\mathrm{WebDice}(p,x)$ (Section 5.2.4) between a name p and a candidate alias x using page counts as

$$\mathrm{WebDice}(p,x) = \frac{2 \times \mathrm{hits}(\text{“}p\ AND\ x\text{”})}{\mathrm{hits}(p) + \mathrm{hits}(x)}. \tag{5.15}$$

Here, $\mathrm{hits}(q)$ is the page counts for the query q.

WebPMI

I compute the pointwise mutual information, $\mathrm{WebPMI}(p, x)$ using page counts as follows:

$$\mathrm{WebPMI}(p, x) = \log_2 \frac{L \times \mathrm{hits}(\text{“}p\ AND\ x\text{”})}{\mathrm{hits}(p) \times \mathrm{hits}(x)}. \tag{5.16}$$

Here, L is the number of pages indexed by the web search engine, which I approximated as $L = 10^{10}$ according to the number of pages indexed by Google. It should be noted however that the actual value of L is not required for ranking purposes because it is a constant and can be taken out from the definition of WebPMI (Equation 5.16) as an additive term. Both WebDice and WebPMI measures are described in detail in [18].

Conditional Probability

Using page counts, I compute the probability of an alias, given a name, as

$$Prob(x|p) = \frac{\mathrm{hits}(\text{“}p\ AND\ x\text{”})}{\mathrm{hits}(p)}. \tag{5.17}$$

Similarly, the probability of a name, given an alias, is

$$Prob(p|x) = \frac{\mathrm{hits}(\text{“}p\ AND\ x\text{”})}{\mathrm{hits}(x)}. \tag{5.18}$$

Unlike pointwise mutual information and the Dice coefficient, conditional probability is an asymmetric measure.

5.2.7 Training

Using a dataset of name-alias pairs, I train a ranking support vector machine [70] to rank candidate aliases according to their strength of association with a name. For a name-alias pair, I define three types of features: anchor text-based co-occurrence measures, web page-count-based association measures, and frequencies of observed lexical patterns. The nine co-occurrence measures described in Section 5.2.4 (CF, tfidf, CS, LLR, PMI, HG, cosine, overlap, Dice) are computed with and without weighting for hubs to produce $18(2 \times 9)$ features. Moreover, the four page-count-based association measures defined in Section 5.2.6 and the frequency of lexical patterns extracted by algorithm 5 are used as features in training the ranking SVM. During training, ranking SVMs attempt to minimize the number of discordant pairs in the training data, thereby improving the average precision. The trained SVM model is used to rank the set of candidates that were extracted for a name. Specifically, for each extracted candidate alias and the name under consideration, I compute the above mentioned feature and represent by a feature vector. The trained SVM model can then be used to assign a ranking score to each candidate alias. Finally, the highest-ranking candidate is selected as the correct alias of the name.

5.2.8 Dataset

To train and evaluate the proposed method, I create three name-alias datasets[3]: the English personal names dataset (50 names), the English place names dataset (50 names), and the Japanese personal names (100 names) dataset. Both our English and Japanese personal name datasets include people from various fields of cinema, sports, politics, science, and mass media. The place name dataset contains aliases for the 50 U.S. states. Aliases were manually collected after referring various information sources such as Wikipedia and official home pages.

I crawled Japanese web sites and extracted anchor texts and the urls linked by the anchor texts. A website might use links for purely navigational purposes which do not convey any semantic clues. In order to remove navigational links in our dataset I prepare a list of words that are commonly used in navigational menus such as *top, last, next, previous, links*, etc and ignore anchor texts that contain these words. Moreover, I remove any links that point to pages within the same site. After removing navigational links our dataset contains $24,456,871$ anchor texts pointing to $8,023,364$ urls. All urls in the dataset contain at least two inbound anchor texts. The average number of anchor texts per url is 3.05 and the standard deviation is 54.02. Japanese texts do not use spaces to separate words. Therefore, tokenizing text is an important pre-processing step. I tokenize anchor texts

[3] www.miv.t.u-tokyo.ac.jp/danushka/aliasdata.zip

using the Japanese morphological analyzer MeCab [77].

5.3 Experiments

5.3.1 Pattern Extraction

Algorithm 5 extracts over 8000 patterns for the 50 English personal names in our dataset. However, not all patterns are equally informative about aliases of a real name. Consequently, I rank the patterns according to their F scores to identify the patterns that accurately convey information about aliases. F score of a pattern s is computed as the harmonic mean between the precision and recall of the pattern. First, for a pattern s I compute its precision and recall as follows:

$$\text{Precision}(s) = \frac{\text{No. of correct aliases retrieved by } s}{\text{No. of total aliases retrieved by } s},$$
$$\text{Recall}(s) = \frac{\text{No. of correct aliases retrieved by } s}{\text{No. of total aliases in the dataset}}.$$

Then, its F-score can be computed as:

$$F(s) = \frac{2 \times \text{Precision}(s) \times \text{Recall}(s)}{\text{Precision}(s) + \text{Recall}(s)}.$$

Table 5.2 shows the patterns with the highest F scores extracted using the English personal names dataset. As shown in Table 5.2, unambiguous and highly descriptive patterns are extracted by the proposed method. Likewise, Table 5.3 shows the patterns with the highest F scores extracted using the English location names dataset. The same pattern extraction algorithm (Algorithm 5) can be used to extract patterns indicating aliases of personal names as well as aliases of location names, provided that we have a set of name, alias pairs for each entity type. This is particularly attractive because we do not need to modify the pattern extraction algorithm to handle different types of named entities. Experimentally, I select the top ranked 200 patterns as features for training. Interestingly, among the extracted pattens I found patterns written in languages other than English, such as *de son vrai nom* (French for *his real name*) and *vero nome* (Italian for *real name*).

Pattern	F-score
* aka [NAME]	0.335
[NAME] aka *	0.322
[NAME] better known as *	0.310
[NAME] alias *	0.286
[NAME] also known as *	0.281
* nee [NAME]	0.225
[NAME] nickname *	0.224
* whose real name is [NAME]	0.205
[NAME] aka the *	0.187
* was born [NAME]	0.153

Table 5.2: Lexical patterns with the highest F-scores for personal names.

Patterns	F-score
[NAME] nickname the *	0.739
[NAME] is nicknamed the *	0.723
[NAME] employment nickname *	0.627
[NAME] state flag or *	0.589
[NAME] nicknamed the *	0.5567
[NAME] is called the *	0.3199

Table 5.3: Lexical patterns with the highest F-scores for location names.

5.3.2 Alias Extraction

In Table 5.4, I compare the proposed SVM-based method against various individual ranking scores (baselines) and previous studies of alias extraction (Hokama and Kitagawa [63]) on Japanese personal names dataset. I use linear, polynomial (quadratic), and radial basis functions (RBF) kernels for ranking SVM. Mean reciprocal rank (MRR) [5] is used to evaluate the different approaches. MRR is defined as follows,

$$\mathrm{MRR} = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{R_i}. \tag{5.19}$$

Therein: R_i is the rank assigned to a correct alias and n is the total number of aliases. MRR is widely used in information retrieval to evaluate the ranking of search results. Formula 5.19 gives high MRR to ranking scores which assign higher ranks to correct aliases.

Method	MRR	Method	MRR
SVM (Linear)	0.6718	Prob$(p\|x)$	0.1414
SVM (Quad)	0.6495	CS(h)	0.1186
SVM (RBF)	0.6089	CF	0.0839
Hokama & Kitagawa	0.6314	cosine	0.0761
tfidf(h)	0.3957	tfidf	0.0757
WebDice	0.3896	Dice	0.0751
LLR(h)	0.3879	overlap(h)	0.0750
cosine(h)	0.3701	PMI(h)	0.0624
CF(h)	0.3677	LLR	0.0604
HG(h)	0.3297	HG	0.0399
Dice(h)	0.2905	CS	0.0079
Prob$(x\|p)$	0.2142	PMI	0.0072
WebPMI	0.1416	overlap	0.0056

Table 5.4: Comparison with baselines and previous work.

If a method ranks the correct aliases of a name on top, then it receives a higher MRR value. As shown in Table 5.4, the best results are obtained by the proposed method with linear kernels (SVM(Linear)). Both ANOVA and Tukey HSD tests confirm that the improvement of SVM(Linear) is statistically significant ($p<0.05$). A drop in MRR occurs with more complex kernels, which is attributable to over-fitting. Hokama and Kitagawa's method which uses manually created patterns, can only extract Japanese name aliases. Their method reports an MRR value of 0.6314 on our Japanese personal names dataset. In Table 5.4, I denote the hub-weighted versions of anchor text-based co-occurrence measures by (h). Among the numerous individual ranking scores, the best results are reported by the hub-weighted tfidf score (tfidf(h)). It is noteworthy that, for anchor text-based ranking scores, the hub-weighted version always outperforms the non-hub-weighted counterpart, which justifies the proposed hub-weighting method. Among the four page-count-based ranking scores, WebDice reports the highest MRR. It is comparable to the best anchor text-based ranking score, tfidf(h). The fact that Prob$(x|p)$ gives slightly better performance over Prob$(p|x)$ implies that we have a better chance in identifying an entity given its real name than an alias.

In Table 5.5, I evaluate the overall performance of the proposed method on each dataset

Table 5.5: Overall performance

Dataset	MRR	AP
English Personal Names	0.6150	0.6865
English Place Names	0.8159	0.7819
Japanese Personal Names	0.6718	0.6646

Table 5.6: Aliases extracted by the proposed method

Real Name	Extracted Aliases
David Hasselhoff	**hoff**, **michael knight**, **michael**
Courteney Cox	**dirt lucy**, **lucy**, **monica**
Al Pacino	**michael corleone**
Teri Hatcher	**susan mayer**, **susan**, mayer
Texas	**lone star state**, lone star, lone
Vermont	**green mountain state**, green,
Wyoming	**equality state**, **cowboy state**
Hideki Matsui	**Godzilla**, **nishikori**, **matsui**

using MRR and average precision (AP) [5]., which is defined as follows,

$$\mathrm{AveragePrecision} = \frac{\sum_{r=1}^{k} \mathrm{Pre}(r) \times \mathrm{Rel(r)}}{\text{No of correct aliases}}. \tag{5.20}$$

Here, $\mathrm{Rel}(r)$ is a binary valued function that returns 1 if the candidate at rank r is a correct alias for the name. Otherwise, it returns zero. Furthermore, $\mathrm{Pre}(r)$ is the precision at rank r, which is given by,

$$\mathrm{Pre}(r) = \frac{\text{no. of correct aliases in top r candidates}}{r}. \tag{5.21}$$

Different from the mean reciprocal rank, which focuses only on rank, average precision incorporates consideration of both precision at each rank and the total number of correct aliases in the dataset. Both MRR and average precision have been used in rank evaluation tasks such as evaluating the results returned by a search engine. With each dataset, I perform a 5-fold cross validation. As shown in Table 5.5, the proposed method reports high scores for both MRR and average precision on all three datasets. Best results are achieved for the place name alias extraction task.

Table 5.6 presents the aliases extracted for some entities included in our datasets. Overall, the proposed method extracts most aliases in the manually created gold standard (shown in bold). It is noteworthy that most aliases do not share any words with the name nor acronyms, thus would not be correctly extracted from approximate string matching methods. It is interesting to see that, for actors the extracted aliases include their roles in movies or television dramas (e.g. *Michael Knight* for *David Hasselhoff*).

Table 5.7 shows the top three ranking aliases extracted for *Hideki Matsui* by the proposed SVM (Linear) measure and the various baseline ranking scores. For each candidate I give its English translation within brackets alongside with the score assigned by the measure. The correct alias, *Godzilla*, is ranked first by SVM (RBF). Moreover, the correct alias is followed by his last name *Matsui* and the team which he plays for, *New York Yankees*. tfidf(h), LLR(h), CF(h) all have the exact ranking for the top three candidates. *hide*, which is an abbreviated form of *Hideki* is ranked second by these measures. However, none of them contain the alias *Godzilla* among the top three candidates. The non-hub weighted measures have a tendency to include general terms such as *Tokyo*, *Yomiuri* (a popular Japanese newspaper), *Nikkei* (a Japanese business newspaper) and *Tokyo stock exchange*. A close analysis revealed that such general terms frequently co-occur with a name in hubs. Without adjusting the co-occurrences coming from hubs, such terms receive high ranking scores as shown in Table 5.7. It is noteworthy that the last name *Matsui* of the baseball player is ranked by most of the baseline measures as the top candidate.

Method	First		Second		Third	
	candidate	score	candidate	score	candidate	score
SVM (Linear)	Godzilla	8.50	Matsui	8.43	Yankees	8.11
tfidf(h)	Matsui	415.24	*hide*	147.78	Yankees	131.42
LLR(h)	Matsui	218.74	*hide*	61.39	Yankees	59.74
cosine(h)	Matsui	0.67	Yankees	0.28	*hide*	0.25
CF(h)	Matsui	43.99	*hide*	15.40	Yankees	12.27
HG(h)	Matsui	173.01	Yankees	49.21	*hide*	47.78
Dice(h)	Matsui	0.04	Yankees	0.02	*hide*	0.01
CS(h)	Matsui	5912.66	Major league	2262.52	player	2144.81
CF	Tokyo	171	Yomiuri	111	Nikkei	95
cosine	Yomiuri	0.79	Tokyo stock exchange	0.68	Matsui	0.60
tfidf	Yomiuri	1040.75	Tokyo	995.09	Tokyo stock exchange	884.64
Dice	Yomiuri	0.019	Tokyo stock exchange	0.017	Matsui	0.015
overlap(h)	play	1.25	Godzilla	1.20	Steinbrenner	0.80
PMI(h)	play	11.15	Godzilla	11.10	Steinbrenner	10.70
LLR	Yomiuri	243.58	Tokyo stock exchange	219.15	jiji.com	172.32
HG	Yomiuri	592.36	Tokyo stock exchange	500.84	Matsui	427.32
CS	jiji.com	104943.29	Tokyo stock exchange	53222.85	Yomiuri	51420.51
PMI	Komdatzien	11.62	picture	10.92	contents	10.90
overlap	Komdatzien	1.00	picture	1.00	contents	1.00

Table 5.7: Top ranking candidate aliases for Hideki Matsui

Real name only			Real name and top alias		
Precision	Recall	F	Precision	Recall	F
.4812	.7185	.4792	.4833	.9083	.5918

Table 5.8: Effect of aliases on relation detection

5.3.3 Relation Detection

I evaluate the effect of aliases on a real-world relation detection task as follows. First, I manually classified 50 people in the English personal names dataset, depending on their field of expertise, into four categories: *music, politics, movies,* and *sports*. Following earlier research on web-based social network extraction [96, 102], I measure the association between two people using the pointwise mutual information (Equation 5.16) between their names on the web.

I then use group average agglomerative clustering (GAAC) [92] to group the people into four clusters. Initially, each person is assigned to a separate cluster. In subsequent iterations, group average agglomerative clustering process, merges the two clusters with the highest correlation. Correlation, $\mathrm{Corr}(\Gamma)$, between two clusters X and Y is defined as

$$\mathrm{Corr}(\Gamma) = \frac{1}{2}\frac{1}{|\Gamma|(|\Gamma|-1)}\sum_{(u,v)\in\Gamma}\mathrm{sim}(u,v). \tag{5.22}$$

Here, Γ is the merger of the two clusters X and Y. $|\Gamma|$ denotes the number of elements (persons) in Γ and $\mathrm{sim}(u,v)$ is the association between two persons u and v in Γ.

Ideally, people who work in the same field should be clustered into the same group. I used the B-CUBED metric [6] to evaluate the clustering results. The B-CUBED evaluation metric was originally proposed for evaluating cross-document coreference chains. I compute the precision, recall and F-score for each name in the dataset and average the results over the number of people in the dataset. For each person p in our dataset, let us denote the cluster that p belongs to as $C(p)$. Moreover, I use $A(p)$ to represent the affiliation of person p, e.g. $A($“Bill Clinton”$)$ $=$“politics”. Then I calculate the precision and recall for person p as

$$\mathrm{Precision}(p) = \frac{\text{No. of people in C(p) with affiliation A(p)}}{\text{No. of people in C(p)}}, \tag{5.23}$$

$$\mathrm{Recall}(p) = \frac{\text{No. of people in C(p) with affiliation A(p)}}{\text{Total No. of people with affiliation A(p)}}. \tag{5.24}$$

Then, the F-score of person p is defined as

$$\mathrm{F}(p) = \frac{2 \times \mathrm{Precision}(p) \times \mathrm{Recall}(p)}{\mathrm{Precision}(p) + \mathrm{Recall}(p)}. \tag{5.25}$$

The overall precision (**P**), recall (**R**) and F-score (**F**) are computed by taking the averaged sum over all the names in the dataset.

$$\mathrm{Precision} = \frac{1}{N} \sum_{p \in DataSet} \mathrm{Precision}(p) \tag{5.26}$$

$$\mathrm{Recall} = \frac{1}{N} \sum_{p \in DataSet} \mathrm{Recall}(p) \tag{5.27}$$

$$F\mathrm{-Score} = \frac{1}{N} \sum_{p \in DataSet} \mathrm{F}(p) \tag{5.28}$$

Here, $DataSet$ is the set of 50 names selected from the English personal names dataset. Therefore, $N = 50$ in our evaluations.

I first conduct the experiment only using real names (i.e. only using the “real name” as the query to obtain web hits). Next, I repeat the experiment by expanding the query with the top ranking alias extracted by the proposed algorithm (i.e.“real name” OR “alias”). Experimental results are summarized in Table 5.8. From Table 5.8, we can see that F-scores have increased as a result of including aliases with real names in relation identification. Moreover, the improvement is largely attributable to the improvement in recall. The inclusion of aliases has boosted recall by more than 20%. By considering not only real names but also their aliases, it is possible to discover relations that are unidentifiable solely using real names.

5.3.4 Web Search Task

In order to retrieve information about a particular person from a web search engine, it is common practice to include the name of the person in the query. In fact it has been reported

No	First Name	Last Name	Alias	F	L	A	F+A	L+A	F+L	F+L+A
1	Hideki	Matsui	godzilla	48	1	3	50	50	50	50
2	Minako	Nakano	nakami	8	0	4	50	38	50	50
3	Maki	Goto	gomaki	16	6	24	48	44	48	50
4	Hidetoshi	Nakata	*hide*	44	6	12	50	50	50	50
5	Takuya	Kimura	kimutaku	34	2	30	50	50	50	50
6	Yuichi	Kimura	brother kimu	29	0	45	49	45	50	50
7	Takafumi	Horie	horiemon	29	5	49	48	50	50	50
8	Hikaru	Utada	hikki	30	48	5	50	48	50	50
9	Ryuichi	Sakamoto	professor	22	3	1	36	17	49	50
10	Miki	Ando	mikiti	41	8	4	47	35	48	50

Table 5.9: Effect of aliases in personal name search

that approximately one third of all web queries contain a person name [54]. A name can be ambiguous in the sense that there might exist more than one individual for a given name. Under such circumstances, searching only by the name is insufficient to locate information regarding the person we are interested in. However, by including an alias that uniquely identify a person from his or her namesakes, it might be possible to filter out irrelevant search results. I set up an experiment to evaluate the effect of aliases in a web search task.

The experiment is conducted as follows. For a given individual, I search Google with the name as the query and collect top 50 search results. I then manually go through the search results one by one and decide whether they are are relevant for the person we searched for. I count the total number of relevant results. I then append the name query with an alias of the person and repeat the above mentioned process. Table 5.9 summarizes the experimental results for 10 Japanese names in our dataset. First, I will briefly describe each person shown in Table 5.9.

Hideki Matsui is a major league baseball player playing for New York Yankees.

Minako Nakano is an announcer for the Fuji TV corporation.

Maki Goto is a J-pop singer and part of *Hello! Project*. She is a former member of the music group *Morning Musume*.

Hidetoshi Nakata was a member of the Japanese national football team and retired from professional football in 2006.

Takuya Kimura a member of Japanese idol group *SMAP*, is a leading actor as well as a singer.

Yuichi Kimura is a member of the comedian group *oldies*.

Takafumi Horie a Japanese entrepreneur, is the founder and former CEO of Japanese internet portal *Livedoor*. He was accused of security frauds in 2006 and sentenced to a 2 years and 6 months imprisonment.

Hikaru Utada is a New York born J-pop singer and song writer.

Ryuichi Sakamoto is a Japanese musician, composer, producer and actor. He has won Academy Awards, Grammy and Golde Globe for his performances.

Miki Ando , the Japanese figure skater, became the 2007 world figure skating champion.

In Table 5.9, I use **F**, **L**, and **A** to denote the first name, last name and alias respectively. The notation X+Y stands for the conjunctive query, *"X"* AND *"Y"* of two terms X and Y. For example, for Hideki Matsui, F+A represents the query *"Hideki"* AND *"Godzilla"*. The numbers in Table 5.9 are the relevant results out of 50 top ranking results returned by Google for each query. Because I search for Japanese names, I search in google.co.jp for pages written in Japanese language.

In Table 5.9, the number of relevant results have improved for both first name (F) and last name (L) only queries when the aliases was added to the query (F vs F+A and L vs L+A). In particular, last names alone produce very poor results. This is because most Japanese last names are highly ambiguous. For example, the last name Nakano (person no. 2) is a place name as well as a person name and does not provide any results for Minako Nakano, the television announcer. Compared to last names, first names seem to have better precision for the people tested. It is also evident from Table 5.9 that aliases alone are insufficient to retrieve relevant results. Despite the popular alias *godzilla* for Hideki Matsui, searching alone by the alias produces just 3 relevant results. The highly ambiguous alias, *professor*, for Ryuichi Sakamoto (person no. 8) returns only the wikipedia disambiguation entry for the musician. However, the combination of last name and alias significantly improves relevant results for all individuals.

We see that searching by the full name (F+L) returns perfect results for 7 out of the 10 people. However, it is noteworthy that including the aliases still improve relevancy even in the remaining 3 cases. If we only consider the last names, *Takuya Kimura* (person no. 5) and *Yuichi Kimura* (person no 6) corresponds to namesakes for the last name *kimura*. Searching only by the last name retrieves very few (almost zero) relevant results. However, they have unique name aliases. Even searching only by their aliases improves relevancy by a high margin (30 and 45 results respectively).

5.4 Implementation considerations

In order to create a contingency table like the one shown in Table 5.1, we require anchor texts pointing to an url. In this section, I describe two independent approaches to efficiently extract anchor texts that point to a url, which is also pointed by the given real name that we are interested in extracting aliases for. First, I describe a technique to extract this information from a large web crawl. Second, I propose an algorithm to retrieve the required information directly from an existing web search engine.

5.4.1 Extracting inbound anchor texts from a web crawl

In the case where we have a large web crawl at our disposal, then extracting all the inbound anchor texts of a url can be accomplished using a hash table, where the hash key is a url and the corresponding value contains a list of anchor texts that point to the url specified by the key. Algorithm 7 illustrates the pseudo code to achieve this task. Given a web crawl, I extract the set L of links from the crawled data, where L contains tuples (a_i, u_i) of anchor texts a_i and the urls u_i pointed by the anchor texts. The function **ANCHORS** in algorithm 7 processes the links in L and returns a hash H where keys are urls u_i and values, $H[u_i]$ are lists of anchor texts pointing to u_i. H is initialized to an empty hash and for each tuple (a_i, u_i) in L, if H already contains the key u_i then the function $\mathrm{Append}(H[u_i], a_i)$ appends the anchor text a_i to the list $H[u_i]$. If u_i does not exist in H then a new list containing a_i is created and assigned to key u_i. If the number of links to be processed is large, then H can be implemented as a distributed hash table.

Algorithm 7 ANCHORS(p)

```
Input: name $p$
Output: H
$H = \{\ \}$
$S \leftarrow \mathrm{InAnchor}(p)$
for url $u_i \in S$ do
    $H[u_i] \leftarrow [\ ]$
    $T \leftarrow \mathrm{Link}(u_i)$
    for page $t \in T$ do
        $L \leftarrow \mathrm{ExtractLinks}(t)$
        for $(a_k, q_k) \in L$ do
            if $q_k = u_i$ then
                $\mathrm{Append}(H[u_i], a_k)$
            end if
        end for
    end for
end for
return H
```

5.4.2 Retrieving anchor texts and links from a web search engine

Crawling and indexing a large set of anchor texts from the web requires both computation power and time. However, there already exist large web indexes created by major web search providers such as Google, Yahoo and MSN, Most web search engines provide query APIs and search operators to search in anchor texts. For example, Google[4] provides the search operator *inanchor* to retrieve urls pointed by anchor texts that contain a particular word. Moreover, the search operator *link* returns urls which are linked to a given url. In Algorithm 8 I describe an algorithm to extract a set of anchor texts that points to urls which are also pointed by a given name using the basic search operators provided by a web search engine.

Given a name p, the function $\mathrm{ANCHORS}(p)$ described in algorithm 8 returns a hash H where keys of H are urls and the corresponding values contain lists of anchor texts pointing to the url specified by the key. Moreover, for each url in H, at least one of the retrieved anchor texts contains the name p. Therefore, all the words that appear in anchor texts

[4] `http://code.google.com/apis/`

Algorithm 8 ANCHORS(S)

```
Input: A list $L$ of tuples $(a_i, u_i)$
Output: H
H = { }
for $(a_i, u_i) \in L$ do
    if $u_i \in H$ then
        $Append(H[u_i], a_i)$
    else
        $H[u_i] \leftarrow [\,]$
    end if
end for
return H
```

extracted by the algorithm contains candidate aliases of the name p. The algorithm first initializes the hash H to empty hash. Then the function $InAnchor(p)$ retrieves urls that are pointed by anchor texts that contain p. For example, in Google, the search operators *inanchor* or *allinanchor* can be used for this purpose. For each url u_i retrieved by the function InAnchor I initialize the list of anchor texts to empty list and retrieve the set of pages T that link to u_i using the function $\mathrm{Link}(u_i)$. For example, in Google, the search operator *link* provides the desired functionality. Then for each page t in T the function $ExtractLinks$ extracts anchor texts and links from t. Finally I accumulate anchor texts that point to u_i and store them as a list in H.

5.5 Discussion

I utilize both lexical patterns extracted from snippets retrieved from a web search engine as well as anchor texts and links in a web crawl. Lexical patterns can only be matched within the same document. In contrast, anchor texts can be used to identify aliases of names across documents. The use of lexical patterns and anchor texts respectively, can be considered as an approximation of within document and cross-document alias references. Experimentally, in Section 5.3.2 I showed that by combining both lexical patterns-based features and anchor text-based features we can achieve better performance in alias extraction. The contingency table introduced in Section 5.2.4 can only incorporate first order co-occurrences.

An interesting future research direction would be to create a word co-occurrence graph using the definition of anchor texts-based co-occurrences given in the paper and run graph mining algorithms to identify second or higher order associations between a name and candidate aliases.

An alias might not always uniquely identify a person. For example, the alias *Bill* is used to refer many individuals who has the first name *William*. The namesake disambiguation problem focuses on identifying the different individuals who have the same name. The existing namesake disambiguation algorithms assume the real name of a person to be given, and does not attempt to disambiguate people who are referred only by aliases. In Section 5.3.4, I showed experimentally that the knowledge of aliases is helpful to identify a particular person from his or her namesakes on the web. Aliases are one of the many attributes of a person that can be useful to identify that person on the web. Extracting common attributes such as date of birth, affiliation, occupation, and nationality have been shown to be useful for namesake disambiguation on the web [136].

Consider the problem of detecting whether a particular relation R holds between two entities A and B. One approach to solve this problem is to find contexts in which A and B co-occur and decide whether the relation R pertains between the entities. For example, if A and B are two researchers, then we can expect a high co-occurrence on the web if they publish their mutual works together or work on the same project. In fact, previous studies of social network extraction [96, 102] have considered co-occurrences on the web as a measure of the social association among people. However, if A and B have name aliases, then it is not possible to collect all the contexts in which they co-occur merely by searching using the real names. To illustrate this point, let us assume the aliases of A and B to be a, b. Then there exists four possible co-occurrences: (A,B), (A,b), (a, B) and (a,b). The query which contains only real names, A **AND** B, covers only one of the four outcomes. Moreover, the number of possible combinations grows exponentially along with the number of aliases for each entity. As seen from the relation detection experiment in Section 5.3.3, knowledge related to aliases can improve a relation detection system by providing more accurate information related to the co-occurrences of entities.

Chapter 6

Attribute Extraction

A person is associated with numerous attributes on the Web. Accurate extraction of attributes for a particular person is important to uniquely identify that person on the web. For example, in the case of namesakes (i.e. people with identical names), although they share the same name, the other attributes such as affiliation, nationality, date of birth, place of birth, etc. might be different. Consequently, extracting various attributes has shown to be useful for personal name disambiguation. For example, consider the two namesakes of the ambiguous name *Jim Clark*: one is a racing driver whereas the other Jim Clark is the founder of Netscape corporation and also a university professor. The attribute *occupation* can separate the two Jim Clarks because one is a sports car driver and the other is a university professor. In this Chapter I describe a preliminary study that I conducted in attribute extraction from the web. In particular, I concentrate on extracting attributes related to people. The motivation of this work is that extracting attributes related to people is useful for both namesake disambiguation (Chapter 4) and name alias extraction (Chpater 5) problems that I already studied in this thesis. In this Chapter, I describe the alias extraction system (**MIVTU**) that I developed which took part in the second Web People Search Task (WePS) workshop.

Web People Search Task (WePS) is aimed at searching for people on the web. The first WePS introduced a name disambiguation task where given a collection of web documents retrieved for a particular name, the objective is to identify the documents that belong to different people with the queried name. The problem can be conveniently modeled as a one of document clustering where each cluster represents a different person of the given ambiguous name. It was found that attributes such as date of birth, nationality, affiliation, occupation, etc. are particularly useful as features to identify namesakes [4]. Consequently, in the second WePS [3], an attribute extraction subtask was introduced. Given a web document, the objective of this attribute extraction task is to extract a pre-defined set of attribute values for a given person name. The WePS attribute extraction task focuses on extracting values for the following 18 attributes: date of birth, birth place, other name, occupation, affiliation, work, award, school, major, degree, mentor, location, nationality, relatives, phone, fax, e-mail, and web site. The definition of each attribute can be found in the task description guide [1]. However, not at attributes are equally represented in the WePS dateset. The most frequent attributes are Work (3770), Occupation (3292), and Affiliation (3105). Here, the total number of occurrences are shown within brackets. Attributes such as fax (65), web site (154), and major (173) are the least frequent attributes in the dataset.

A system that attempts to extract attributes for a given person from web documents must solve several sub-problems. First, it must identify the different occurrences of the given person name. This is challenging because of two main problems: namesakes and name aliases on the web. Although a web page might contain the given person name it could be a page for a different person who has the identical name. In attribute extraction task at the second WePS workshop only documents relevant to the person under consideration are given. Therefore the problem of namesake disambiguation does not occur. In fact, the objective of attribute extraction in WePS is to use the extracted attributes to disambiguate people on the web. However, a particular individual can be represented by more than one name on the web. For example, *William Gates* is more commonly called as *Bill Gates* in web contexts. Moreover, abbreviated variants of names are common. For example, *John Fitzgereld Kennedy* has the variants *J.F.K.*, *John F. Kennedy*, and *J. F. Kennedy*. Although it is relatively easy to cover the above mentioned variants using dictionaries of common name aliases (i.e. Bill vs. William, Jim vs. James, etc.) and regular expressions, some name aliases such as *Fresh Price* for *Will Smith* or *Godzilla* for *Hideki Matsui* are difficult to identify automatically [20].

Once the occurrences of the given name is identified in a set of documents, an attribute

[1] http://nlp.uned.es/weps/weps2/WePS2_Attribute_Extraction.pdf

extraction system must extract attributes and their values. Attribute extraction step can be further divided into two parts – the attribute extraction system must first identify the values for the given set of attributes and then decide which attribute values are relevant to the person under consideration. Attributes such as *e-mail addresses, urls, telephone numbers, fax numbers*, and *birth dates* follow a specific format and are easy to detect. However, attributes such as *relative, mentor, school, award, degree*, and *affiliation* have many variations and are difficult to detect. For example, a mentor of a person can be introduced in a text as the *teacher*, *advisor*, *professor*, *supervisor*, etc. The set of values such attributes can take is open, and cannot be completely enumerate using pre-compiled lists. For example, the attribute *mentor* can take any person name as its value. Named entity recognition tools can solve this problem partially. However, most named entity recognition tools only cover more common entities such as personal names, locations, and organizations. They do not annotate awards, majors, nationalities or classify organizations into schools. Moreover, named entity recognition tools are usually trained on noise-free text corpora such as news articles and do not show optimal performance on relatively noisy web documents with numerous markups such as HTML and Javascript. Therefore, identifying which strings can be potential attributes for a person is an important task that an attribute extraction system must perform.

Finally, an attribute extraction system must select the attribute values relevant to the given person. A document might contain information regarding more than one person. Consequently, not all attributes that appear in a document might be relevant to the person under consideration. A simple yet effective heuristic is to associate attributes closest to an occurrence of the given person name. It is likely that a person denote his or her contact information such as e-mail, telephone and fax close to the name in a home page. However, this simple heuristic cannot cover the cases where a name and a relevant attribute appear in distant parts in a document. Moreover, it is not clear how to handle cases where more than one person name appear in a document – should we go beyond an occurrence of a different name and associate attributes or not.

This chapter describes the **MIVTU** system that participated in the attribute extraction subtask at the second WePS workshop. According to the official results, MIVTU was ranked 5^{th} among the 15 systems that participated in the attribute extraction task at the second WePS. However, the highest overall F-scores reported by all participating systems is 12.2. MIVTU system reported an F-score of 8.3. The low performance reprted by the pariticipants in WePS-2 attribute extraction task, highlights the difficulty of accurately extracting attributes for people. The challenges discussed above are not yet fully solved by any participating system. However, the applications that can benifit from an accurate attribute extraction system are numerous. I believe that we will see further research on this field in near future. In Chapter 8, I discuss some aspects of attribute extraction that can be improved.

6.1 Related Work

Extracting attribute values for an entity has wide applications in information extraction and retrieval. Pasca [114] proposed a method to extract born year of people from web text. For example, from the text *Mozart was born in 1756*, this method extracts the pair (*Mozart*, 1756). The extraction algorithm starts from as few as 10 seed facts, and is capable of extracting facts from over 100 million web documents. Seed facts are searched on web texts and lexical patterns are generated. Then the generated lexical patterns are searched in web texts and new facts are extracted. The process is repeated with newly extracted facts as seeds. Although this method was used to extract birth year of people, a similar boot strapping approach can be followed to extract other types of attributes such as birth place, nationality and occupation.

Bellare et al. [11] proposed a lightly-supervised approach to attribute extraction from the web. They first tag a given text corpus with part-of-speech information and then from each tagged sentence extract all proper noun and noun pairs. They consider each extracted pair as a candidate entity-attribute instance. Each candidate instance is assigned with a set of features. They select the left, right and middle contexts that appear around the entity and candidate attribute as features. They use two learning methods: decision lists by co-training using a mutual information-based measure, and a maximum-entropy classifier by self-training. They evaluate their algorithm on two tasks: extracting the set of attributes for companies, and extracting the set of attributes for countries. However, they do not extract the values for those attributes.

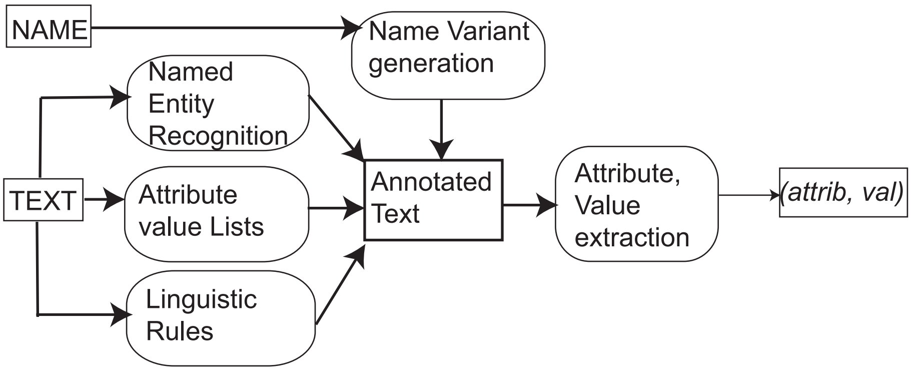


Figure 6.1: Outline of the proposed system

6.2 Attribute Extraction from the Web

6.2.1 Outline

The proposed method is illustrated in Figure 6.1 and it can be seen as consisting of two fundamental steps. First, I mark potential attribute values in a given text. Second, I decide which candidate values correspond to which attributes of the given person name.

To mark the potential values of attributes I use three approaches: lists of candidate attribute values, a named entity recognizer, and a set of manually created rules in the form of regular expressions. For example, attributes such as nationalities (e.g. Japanese, British) , universities (e.g. The University of Tokyo), majors (e.g. Master of Science, Bachelor of Arts) and professional titles (e.g. professor, general) can be marked using candidate lists. These lists were created manually referring online information sources such as Wikipeida. However, lists cannot completely enumerate all attribute values. In addition to using precompiled lists of attribute values, we used a named entity recognition tool to mark three types of named entities: personal names, organization names, and location names. Attributes such as dates, telephone numbers, fax numbers, e-mail addresses and urls usually follow a fixed format and can be efficiently annotate in a text using rules in the form of regular expressions.

Once the given text is annotated following the above mentioned procedure, I mark all potential variants of the given name for which we must extract attribute values. I generate abbreviated forms, last name and first name inter-changed forms, middle name initialized forms, middle name dropped forms, name followed by titles, and combinations of all the above. We then mark those variants in the given text. For example, if the given name is *John Fitzgereld Kennedy* then this process will generate variants such as *J. F. Kennedy*, *John F. Kennedy*, *Kennedy J. F.*, and *John Kennedy*. To find the attributes of the given person, I find the distance for each marked attribute value from a name variant. I then select the closest attribute value as the correct candidate. However, I do not go beyond a different person name when computing distances. Moreover, I assign higher confidence score to an extracted attribute value if certain cue phrases appear in close proximity. For example, the cue phrases *born* and *birth* increase the confidence of an extracted date being the date of birth of the person under consideration. Likewise, cue phrases *mentor*, *supervisor*, and *advisor* increase the confidence of a value extracted as the mentor of the person under consideration. The cue phrases are selected manually after reviewing the test data in the WePS attribute extraction dataset. Each sub-component of the proposed attribute extraction system including examples of candidate value lists, linguistics rules, cue phrases, and attribute extraction method will be further explained in the sections to follow.

6.2.2 Pre-processing

WePS attribute extraction task dataset contains HTML documents for a set of person names. However, named entity recognition tools have difficulties in operating on HTML marked texts. Therefore, I first remove all HTML markups using an external tool[2]. I then use Stanford named entity recognizer[3] and annotate the text for person names, locations and organizations. The remainder of the processing described in the paper use this annotated text version of the dataset and does not use the original HTML version. I use a set of rules to generate probable variants of the given person name. First, I split the given name into first name and last name. I then generate the following variants: first name followed by last name, last name followed by first name, a comma appearing between the two names, a word appearing between the two names, first name initialized and immediately followed by the last name, last name followed by a comma and the first name initialized, and first

[2] http://www.oluyede.org/files/htmlstripper.py

[3] http://nlp.stanford.edu/software/CRF-NER.shtml

Benjamin [VARIANT,1] Snyder and Phedora [ORGANIZATION,3] Blazer Benjamin Snyder and Phoebe Ann Blazer Husband: Benjamin [VARIANT,1] Snyder born 12 DEC 1827 in Dayton, [LOCATION,0] Montgomery [ORGANIZATION,1] Co., OH died 6 JUL 1873 in Montreal, [LOCATION,0] Camden [ORGANIZATION,1] Co., MO.. [LOCATION,0] buried: Freedom [ORGANIZATION,5] Church, Linn Creek, Camden Co., MO [LOCATION,0] married: Phoebe Ann BLASER Bef 1855 in OH Wife: Phoebe Ann BLASER born 25 JUL 1838 in OH died: 20-Feb-1896 in MO [LOCATION,0] buried: Freedom Cemetery -LRB- West side -RRB- Children of Benjamin [VARIANT,1] Snyder and Phoebe [ORGANIZATION,2] Ann Blazer 1. Andrew Jackson Snyder [VARIANT,0] born: JUL 1855 in OH died: 16 OCT 1935 in Tulsa, [LOCATION,0] Tulsa [ORGANIZATION,1] Co., OK. married: Delilah Caroline MOSBY 1878 in IN born: 20 NOV 1866 in Evansville, [LOCATION,0] Vanderburgh [ORGANIZATION,1] Co., IN daughter of Vincent MOSBY and Manerva SAMUELS died: 8 JUN 1910 in Linn [ORGANIZATION,3] Creek, Camden Co., MO [LOCATION,0] buried: Freedom [ORGANIZATION,1] Cemetery - Montreal, [LOCATION,0] ...

Figure 6.2: Example of an annotated text for the person Benjamin Snyder.

name initialized and followed by a word and the last name. I also consider all combinations of the above variants with the following titles: Mr., Mrs. Miss., Ms, Rev., Prof., President, Minister, Prime Minister, General, Madame, Lady, Dr., King., Queen, Vice President, Senator, Lawyer, Major, Maj., General, Gen., Maj. Gen., Major General, and Jr. For example, given the name *George Bush* the above mentioned process recognizes the string *president George W. Bush* as a variant of the given name. The process is an over generating one and in practice generates a large number of variants that never occur in the corpus. However, once the candidate variants are generated they can be efficiently matched using regular expressions. Figure 6.2 shows an example annotation produced by the pre-processing step. In Figure 6.2 we use the format [TAG_NAME, LENGTH_IN_WORDS] to mark the span of a tag. For example, *Benjamin [VARIANT,1] Snyder* indicates that Benjamin Snyder is a variant of the given name.

6.2.3 Attribute Extraction

I use the HTML markup removed and annotated text produced by the pre-processing to extract attributes. Next, I describe the extraction procedure for each of the attributes in detail.

Date of birth: I use a set of rules in the form of regular expressions to mark all date strings in the text. I then normalize all date strings to YEAR/MONTH/DAY format. The following Perl versions of the regular expressions are used to mark dates.

```
/((month_exp)\s*(\d+),?\s+(\d+))/gi
/((month_exp)\.?\s*(\d+))/gi
/((\d{1,2})\s*($month_exp)\s*(\d{2,4}))/gi
/((\d{1,2})\s+(\d{1,2})\s+(\d{2,4}))/gi
/((\d+)\/(\d\d?)\/(\d\d?))/g
/((\d+)\/(\d{1,2}))/g
/((\d+)\.(\d\d?)\.(\d\d?))/g
/(\d\d\d\d)/g
```

Here, month_exp is a variable that holds the names of months. Once all dates are marked in the text, I assign confidence scores to dates that appear closer to the given name (or its variants) or have cue phrases such as *born* or *birth*.

Birth place: I use the location markups as given by the named entity recognition tool to identify candidates for the birth place. I then assign confidence scores to locations that appear closer to the given name (or its variants) or have cue phrases such as *birth place* or *born in*.

Other name: The name variant generation procedure described in section 6.2.2 annotates name variants. I select those variants as other names of the given name. Moreover, I use cue phrases such as *a.k.a.*, *also known as*, *alias*, and *other name* to identify other names for the person under consideration.

Occupation: I created a list of occupations from Wikipedia[4]. The list contains 666 entries. I then select the occupation closest to the given name or any of its variants in the text. Moreover, we tokenized each entry in the occupation list into words and sorted the

[4] http://en.wikipedia.org/wiki/List_of_occupations

words according to their total frequency within the list. The goal of this is to identify words that are commonly used to describe occupations. If a sequence of words contain any of those high frequency words, I select those sequences as occupations. The most frequent words that occur in occupations are: engineer (11), officer (8), scientist (6), Technologist (5), agent (5), designer (5), Financial (5), and worker (5).

Affiliation: I consider companies and universities as affiliations. I create lists for universities and companies using Wikipedia. The created company names list contains 43040 entries and university list contains 1726 entries. I also find the frequency of words that appear in each of these entity types as we did for occupations. The top 10 most frequent words that appear in company names are: Inc. (16137), Corporation (3932), Ltd. (2277), Limited (2126), Company (1993), LLC (1782), Group (1685), plc (976), and International (835). If a capitalized sequence of continuous words contain those words we mark it as an company. Although words such as *of* (1814), *&* (1814), and *The* (1514) are also highly frequent in company names, I remove such words using a stopwords list because those words are ambiguous and can appear in various contexts not necessarily for companies. Moreover, the named entity recognition tool we used in the pre-processing step also provides some company names. A similar word frequency analysis for university names revealed that the most frequent words that appear in university names to be the following: University (861), College (662), State (234), New (74), Saint (56), and Institute (55).

Work: Works of people are very difficult to extract. Books written by authors and movies created by film directors are such cases. However, what can be a work differs from person to person. It is not feasible to cover all value types for this attribute using lists. MIVTU system does not extract this attribute.

Award: I used Wikipedia to create a list of awards. The list contains 454 entries. Any entry that is found in this list is marked as an occurrence of an award in the given text. I perform a word frequency analysis on this list and found the following words to be the most commonly used in names of awards: Award (85), Prize (62), Medal (58), and Order (41). Any continuous sequence of capitalized words that include these words are marked as awards. However, some awards such as *Common Wealth Award* *of Distinguished Service* and *National Medal of Science* contain the preposition *of* which is not capitalized. I found *of* to appear 98 times in award names. Initially, I had removed *of* because it is a common stopword. But we reinstate *of* in order to facilitate the award names that contains it.

School: A list of high schools was created from Wikipedia's "list of" pages. The compiled list contains 25271 entries. Any entry that is found in this list is marked as an occurrence of a school in the given text. The word frequency analysis shows the most frequent words that appear in names of schools to be: School (18618), High (16112), Academy (1828), Christian (1651), HS (1469), Central (684), and Senior (640). First letter capitalized sequences that contain those high frequent words are also marked as schools. However, the confidence score assigned to such partial matches are lower than that for complete entries in the list. Confidence scores are experimentally determined by manual supervision.

Major: I prepared a list of majors by referring to fields of studies offered by some top universities. The compiled list contains 318 entries. Any entry that appear in this list is marked as an occurrence of a major with a high confidence score. The most frequent words that appear in this list are: Studies (33), Engineering (24), Science (22), Management (17), and Education (13). I assign low confidence scores to first letter capitalized continuous word sequences that contain those words.

Degree: A list of degrees was compiled manually using Wikipeida. Our list contains 175 entries. We have both acronym versions of degrees (e.g., M.Eng) and the corresponding full forms (e.g., Masters of Engineering). The word frequency analysis reveals the most frequent words that appear in degree names to be: of (57), Doctor (31), Master (15), Bachelor (12), Administration (8), Science (8), Engineer (8), Medicine (7), degree (7), in (6), Business (5), and Licentiate (5). We ignore common stop words such as *of* and *in* because they are ambiguous and can occur in other contexts other than degrees. If a first letter capitalized continuous sequence of words contain any one or more of those high frequency words then I increase the confidence score assigned to that sequence being a name of a degree. However, the confidence scores assigned in word frequency analysis are lower than the confidence scores assigned when an entire entry in the list get matches in the text. The exact values of the confidence scores are adjusted manually.

Mentor: The value set for the attribute mentor contains only person names. Therefore, I use all person names given by the named entity recognition tool as potential candidates for the attribute mentor if they appear near some cue phrases such as *studied with*, *worked with*, *coach*, *trainer*, *adviser*, *mentor*, *supervisor*, and *spiritual adviser*. We checked the local contexts of the attribute mentor in WePS training data to determine the above mentioned cue phrases. Concretely, we extract a pre-defined window of text from all occurrences of the attribute mentor in WePS training data and then manually go through these contexts to identify the cue phrases. Once person names are marked as candidates for mentors, we then select the candidate that is closest to any of the variants of the given person name as the mentor for that person.

Location: I use the location annotation provided by the named entity recognition tool to mark potential candidates for this attribute. I then select the mention of location that is closest to any of the variants of the given name.

Nationality: I prepare a list of nationalities. The list contains 442 entries. It has multiple entries for certain nationalities (i.e. both *Englishmen* and *British* are marked for United Kingdom). Entries found in this list are marked as nationalities in the text. I then select the nationality tag that is closest to any variant of the given person name in the text as the correct nationality of the person under consideration.

Relatives: The set of values that the attribute *relatives* can take consists of person names. I mark all person names annotated by the named entity recognition tool as candidates of relatives of the person under consideration if a set of cue phrases that indicate various relationships exist in the immediate context of the candidate. I select a window of 10 words around the candidate as its immediate context. Cue phrases are selected from pages describing relationships in Wikipedia. It contains the following entries: ”spouse”, ”brother”, ”sister”, ”wife”, ”husband”, ”father”, ”mother”, ”married”,”son”, ”daughter”, ”late husband”, ”late wife”, ”widow”, ”grand father”,

”grand mother”, ”aunt”, ”uncle”, ”step father”, ”step mother”, ”brother-in-law”, ”sister-in-law”, ”son-in-law”, ”nees”, ”nephew”, ”father-in-law”, ”mother-in-law”, ”child”, ”children”, ”sibling”, ”parent”, ”meet”, ”met”, ”girl friend”, ”boy friend”, ”finance”, ”ex-girlfriend”, ”ex-boyfriend”, ”ex-husband”, ”ex-wife”.

Phone and Fax: I use the following regular expression to mark strings that are likely to be telephone numbers or a fax numbers.

```
(((\+\d{1,3}(-| )?\(?\d\)?(-| )?\d{1,5})|
(\(?\d{2,6}\)?))(-| )?(\d{3,4})(-| )
?(\d{4})(( x| ext)\d{1,5}){0,1})
```

I then mark those candidate strings as a telephone numbers if the cue phrases *tel*, *telephone*, *phone*, *mobile* occur in the immediate context of the candidates. I set a window of 3 words as the immediate context of a candidate. Likewise, a candidate string is marked as a fax number if the cue phrase *fax* occur in its immediate context. I then select the closest candidate to any variant of the given name as the correct attribute value for the person under consideration.

Email: E-mail addresses are marked using the following regular expression.

```
([\w\-\.]+@(\w[\w\-]+\.)+[\w\-]+)
```

I use a stop list for e-mail addresses that occur frequently on web documents such as webmaster@domain or support@domain. This exclusion list of e-mail addresses is compiled manually. Moreover, I found that people have a tendency to include a substring of their first or last names (or both) in their e-mail addresses. Therefore, I increase the confidence of an extracted candidate string if it satisfies those conditions. Finally, the e-mail address candidate that is closest to any of the variants of the given name is selected as the e-mail address of the person under consideration.

Web site: I use the following regular expression to extract urls.

```
(https?://([-\w\.]+)+(:\d+)?
(/([\w/_\.]*(\?\S+)?)?)?)
```

Table 6.1: Overall results for participating systems.

Rank	System	Precision	Recall	F-score
1	PolyUHK	30.4	7.6	12.2
2	CASIANED	8.5	19.0	11.7
3	ECNU_2	8.0	17.6	11.0
4	ECNU_1	6.8	18.8	10.0
5	MIVTU	5.7	15.5	8.3
6	UvA_2	4.4	27.4	7.6
7	UvA_1	2.7	27.3	5.0
8	UC3M_5	8.0	3.6	5.0
9	UvA_5	3.3	2.8	3.1
10	UC3M_1	2.5	2.2	2.3
11	UC3M_2	2.4	2.2	2.3
12	UC3M_3	2.2	2.0	2.1
13	UC3M_4	2.2	2.0	2.1
14	UvA_3	0.7	0.2	0.2
15	UvA_5	0.2	0.0	0.0

Table 6.2: Performance of MIVTU system by each name.

Name	Matches	Over generations	Misses	Precision	Recall	F-score
Benjamin Snyder	42	771	268	5.166	13.548	7.480
Hao Zang	66	747	279	8.118	19.130	11.399
Amanda Lentz	33	1185	279	2.709	10.577	4.314
Otis Lee	26	495	304	4.990	7.879	6.110
Bertram Brooker	56	1247	380	4.298	12.844	6.440
Jason Hart	174	1015	592	14.634	22.715	17.801

I then select the url that is closest to any of the variants of the given name as the correct web url for the person under consideration.

6.3 Results and Discussion

The proposed attribute extraction system is evaluated on the test dataset created for the second WePS workshop. This dataset contains 3468 web documents retrieved for 30 people names. The average number of documents per name is 115.6. Out of those documents 585 were ignored during the annotation process and only the remaining 2883 were used for testing. 2421 documents in the test dataset have at least one attribute value. There were 462 documents without any attribute values. For further details of the annotation process and datasets refer [136]. Each participating system is evaluated by comparing the attributes produced by that system for a particular name against the gold standard attributes created by the annotators. Comparisons are done using precision, recall and F-score. Those evaluation metrics are computed for each individual name in the test dataset as well as for the overall set of attributes extracted by each system. Evaluation metrics are computed using following formulas,

$$\mathrm{precision} = \frac{\text{no. of correctly identified attribute values by system}}{\text{no. of attribute values produced by the system}},$$

$$\mathrm{recall} = \frac{\text{no. of correctly identified attribute values by system}}{\text{no. of attribute values in gold data}},$$

$$F - \mathrm{score} = \frac{2 \times \mathrm{recall} \times \mathrm{precision}}{\mathrm{recall} + \mathrm{precision}}.$$

Table 3.9 summarizes the results produced by each participating system in the attribute extraction task at the second WePS. We have arranged the systems according to their overall F-scores. Results shown for UC3M_4 and UC3M_5 are for unofficial runs submitted after the results submission deadline. The proposed system is ranked 5^{th} among the 15 systems shown in Table 6.1. It reports an overall F-score of 8.3. The best performing system is PolyUHK. It has an F-score of 12.2. Overall, all the systems report low F-scores. The highest precision reported by any individual system is 30.4 (PolyUHK) and the highest recall reported by any individual system is 19.0 (CASIANED). This fact suggests that the task of attribute extraction is indeed challenging and none of the systems successfully overcome the difficulties described at the begining of this chapter. Among the 18 attributes considered in the task, MIVTU system reported the highest recall for three attributes: date of birth (32.0), birth place (48.5), and affiliation (23.0). All systems had difficulties in extracting the attributes major, mentor and award.

Table 6.2 shows the performance of the proposed (MIVTU) system per each name used in the evaluations. Best results are reported for *Jason Hart*. From Table 6.2 we see that MIVTU system generally has better recall values compared to precision values. This is a side effect of it using various lists and over generating candidates. A more conservative approach to tagging candidates might help to overcome this problem. During our experiments we noticed that the named entity tagger itself has a tendency to mark first letter capitalized consecutive sequences of words as named entities even when they were not.

To improve the accuracy of attribute extraction we must improve both steps: marking candidate attributes in text and finding which attribute values are relevant to the person under consideration. The list-based approach that we used to find candidate attribute values has several limitations. First, one cannot enumerate all attribute values using lists. Attributes such as nationalities can be listed up because the number of countries is a closed set. However, attributes such as awards, occupations, birth place, location, affiliation, school and mentor are typical examples of open sets. In addition to using pre-compiled lists, we must have some form of rules to identify such attributes. Moreover, the use of lists can also introduce a level of ambiguity because an entry in a list can appear in a different context in the text. For example, some entries in a list of schools can also be valid entries for affiliation of a person if that person is actually employed in that school. The second step of determining which attribute values are relevant to the person under consideration could be improved if we can merge results from different documents. For example, an attribute such as birth date or birth place should be the same even if it is extracted from different documents for the same person. By enforcing such constraints we might be able to reduce the number of candidate attribute values and thereby make an accurate decision. However, the task guidelines in WePS does not allow this form of cross-document information integration because the objective is to use the set of extracted attributes to cluster the pages. Therefore, the attribute extraction systems must process each document separately.

Chapter 7

Relational Model of Semantic Similarity

In Section 1.8 I explained the relationship between attributional and relational similarity. I argued that relational similarity is a more primal concept than attributional similarity. However, we lack a theoretical model that can be used to measure attributional (semantic) similarity between words that is motivated by semantic relations. In this Chapter, I present some preliminary work that I conduct on those lines.

Geometric models, such as multi-dimensional scaling has been used in psychological experiments analyzing the properties of similarity [76]. These models represent objects as points in some coordinate space such that the observed dissimilarities between objects correspond to the metric distances between the respective points. Geometric models assume that objects can be adequately represented as points in some coordinate space and that dissimilarity behaves like a metric distance function satisfying minimality, symmetry and triangle inequality assumptions. However, both dimensional and metric assumptions are open to question.

Tversky [153] proposed the *contrast model* of similarity to overcome the problems in geometric models. The contrast model relies on featural representation of objects, and it is used to compute the similarity between the representations of two objects. Similarity is defined as an increasing function of common features (i.e. features in common to the two objects), and as a decreasing function of distinctive features (i.e. features that apply to one object but not the other). The attributes of objects are primal to contrast model and it does not explicitly incorporate the relations between objects when measuring similarity.

I propose a model to compute the semantic similarity between two words a and b using the set of semantic relations $R(a, b)$ that hold between a and b. The proposed model is called the *relational model* of semantic similarity, and it is defined by the following equation,

$$\mathrm{sim}(a, b) = \Xi(R(a, b)). \tag{7.1}$$

Here, $\mathrm{sim}(a, b)$ is the semantic similarity between the two words a and b, and Ξ is a weighting function defined over the set of semantic relations $R(a, b)$. Given that a particular set of semantic relations are known to hold between two words, the function Ξ expresses our confidence on those words being semantically similar.

A semantic relation can be expressed in a number of ways. For example, given a taxonomy of words such as WordNet, semantic relations (i.e. hypernymy, meronymy, synonymy etc.) between words can be directly looked up in the taxonomy. Alternatively, the labels of the edges in the path connecting two words can be used as semantic relations. However, in this paper we do not assume the availability of manually created resources such as dictionaries or taxonomies. We represent semantic relations using automatically extracted lexical patterns. Lexical patterns have been successfully used to represent various semantic relations between words such as hypernymy [60] and meronymy [12]. Following these previous approaches, we represent $R(a, b)$ as a set of lexical patterns.Moreover, we denote the frequency of a lexical pattern r for a word pair (a, b) by $f(r, a, b)$.

So far we have not defined the functional form of Ξ. A straightforward approach is to use a linearly weighted combination of relations as shown below,

$$\Xi(R(a, b)) = \sum_{r_i \in R(a,b)} w_i \times f(r_i, a, b). \tag{7.2}$$

Here, w_i is the weight associated with the lexical pattern r_i and can be determined using training data. However, this formulation has two fundamental drawbacks. First, the number of weight parameters w_i is equal to the number of lexical patterns. Typically two words can co-occur in numerous patterns. Consequently, we end up with a large number of parameters in the model. Complex models with a large number of parameters are difficult to train because they tend to overfit the training data. Second, the linear combination given in Equation 7.2 assumes the lexical patterns to be mutually independent. However, in practice this is not true. For example, both patterns *X is a Y* and *Y such as X* indicate a hypernymic relation between *X* and *Y*.

To overcome the above mentioned limitations, we first cluster the lexical patterns to identify the semantically related patterns. Our clustering algorithm is detailed in section 2.3.8. Next, we define Ξ using the formed clusters as follows,

$$\Xi(R(a,b)) = \mathbf{x}_{ab}^T \Lambda \sigma. \tag{7.3}$$

Here, $\mathbf{x}_{ab}$ is a feature vector representing the words a and b. Each formed cluster contributes a feature in vector $\mathbf{x}_{ab}$ as described later in Section 7.2. The vector σ is a prototypical vector representing synonymous word pairs. We compute σ as the centroid of feature vectors representing synonymous word pairs. Λ is the inter-cluster correlation matrix. The (i,j)-th element of matrix Λ denotes the correlation between the two clusters c_i and c_j. Matrix Λ is expected to capture the dependence between semantic relations. Intuitively, if two clusters i and j are highly correlated, then the (i,j)-th element of Λ will be closer to 1. Equation 7.3 computes the similarity between a word pair (a,b) and synonymous word pairs. Intuitively, if the relations that exist between a and b are typical relations that hold between synonymous word pairs, then Equation 7.3 returns a high similarity score for a and b.

The proposed relational model of semantic similarity differs from feature models of similarity, such as the contrast model [153], in that it is defined over the set of semantic relations that exist between two words instead of the set of features for each word. Specifically, in contrast model the similarity $S(a,b)$ between two objects a and b is defined in terms of the features common to a and b, $A \cap B$, the features that are distinctive to a, $A - B$, and the features that are distinctive to b, $B - A$. The contrast model is formalized in the following equation,

$$S(a,b) = \theta f(A \cap B) - \alpha f(A - B) - \beta f(B - A). \tag{7.4}$$

Here, the function f measures the salience of a particular set of features, and non-negative parameters α, β, and θ determine the relative weights assigned to the different components. However, in the relational model of similarity we do not focus on features of individual words but on relations between two words.

Modeling similarity as a phenomenon of relations between objects rather than features of individual objects is central to computational models of analogy-making such as the structure mapping theory (SMT) [45]. SMT claims that an analogy is a mapping of knowledge from one domain (base) into another (target) which conveys that a system of relations known to hold in the base also holds in the target. The target objects do not have to resemble their corresponding base objects. During the mapping process, features of individual objects are dropped and only relations are mapped. The proposed relational model of similarity uses this relational view of similarity to compute semantic similarity between words.

7.1 Empirical Evaluation of the Model

To compute semantic similarity between two words using the relational model (Equation 7.3), we must first extract the numerous lexical patterns from contexts in which those two words appear. We use the pattern extraction procedure detailed in section 3.3.2 for this purpose. Evaluating a semantic similarity measure is difficult because the notion of semantic similarity is subjective. Miller-Charles dataset [103] has been frequently used to benchmark semantic similarity measures. Miller-Charles dataset contains 30 word-pairs rated by a group of 38 human subjects. The word-pairs are rated on a scale from 0 (no similarity) to 4 (perfect synonymy). Because of the omission of two word-pairs in earlier versions of WordNet, most researchers had used only 28 pairs for evaluations. The degree of correlation between the human ratings in the benchmark dataset and the similarity scores produced by an automatic semantic similarity measure, can be considered as a measurement of how well the semantic similarity measure captures the notion of semantic similarity held by humans. In addition to Miller-Charles dataset we also evaluate on the WordSimilarity-353 [46] dataset. In contrast to Miller-Charles dataset which has only 30 word pairs, WordSimilarity dataset contains 353 word pairs. Each pair has 13-16 human judgments, which were averaged for each pair to produce a single relatedness score. Following the previous work, we use both Miller-Charles dataset and WordSimilarity dataset

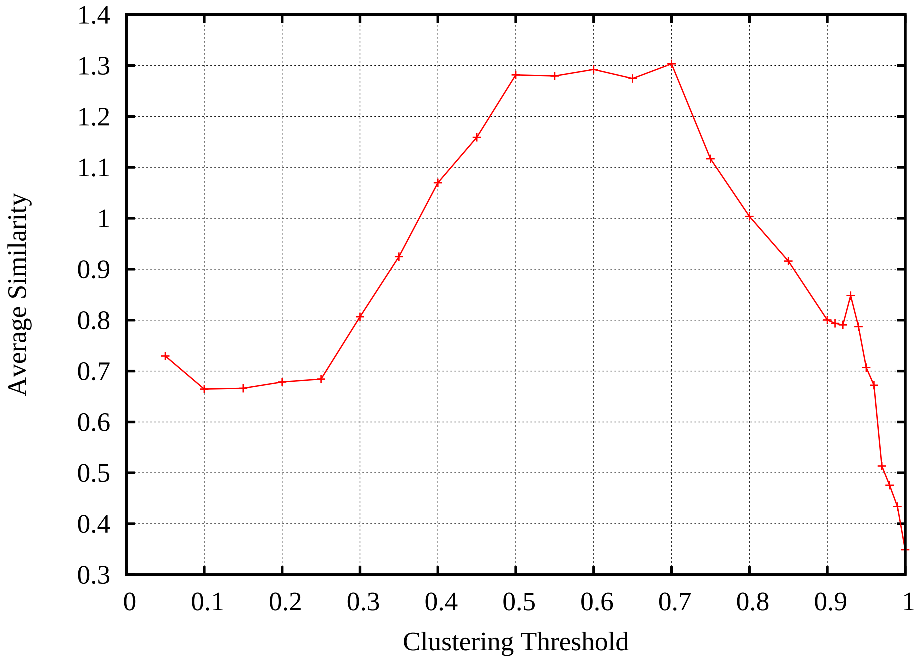


Figure 7.1: Average similarity vs. clustering threshold θ

to evaluate the proposed semantic similarity measure (Prop).

7.2 Computing Semantic Similarity

The degree of correlation between the human ratings in the benchmark dataset and the similarity scores produced by an automatic semantic similarity measure, can be considered as a measurement of how well the semantic similarity measure captures the notion of semantic similarity held by humans. In addition to Miller-Charles dataset I also evaluate on the WordSimilarity-353 [46] dataset. In contrast to Miller-Charles dataset which has only 30 word pairs, WordSimilarity-353 dataset contains 353 word pairs. Each pair has 13-16 human judgments, which were averaged for each pair to produce a single relatedness score. Following the previous work, I use both Miller-Charles dataset and WordSimilarity-353 dataset to evaluate the proposed semantic similarity measure. To perform a fair evaluation,

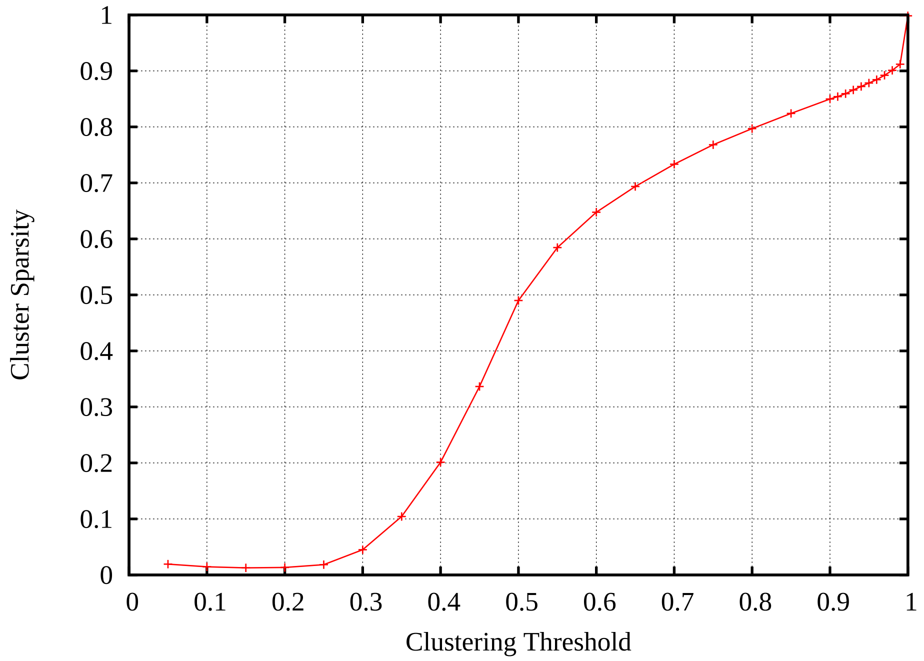


Figure 7.2: Sparsity vs. clustering threshold θ

we do not select any words that appear in the Miller-Charles dataset or the WordSimilarity-353 dataset, which are used later for evaluation purposes. As I will describe later, the clustering threshold θ is tuned using this set of 2000 word pairs selected from the WordNet.

I use YahooBOSS API[1] and download 1000 snippets for each of those word pairs. Experimentally, I set the values for the parameters in the pattern extraction algorithm (section 3.2.1): $L = 5$, $g = 2$, $G = 4$, and extract $5,238,637$ unique patterns. However, only $1,680,914$ of those patterns occur more than twice. Low frequency patterns often contain misspellings and are not suitable for training. Therefore, I select patterns that occur at least 10 times in the snippet collection. Moreover, I remove very long patterns (ca. over 20 chars). The final set contains $140,691$ unique lexical patterns. The remainder of the experiments described in this chapter use those patterns.

I use the clustering Algorithm 3 to cluster the extracted patterns. The only parameter in Algorithm 3, the clustering threshold θ, is set as follows. We vary the value of theta θ from

[1] http://developer.yahoo.com/search/boss/

0 to 1, and use Algorithm 3 to cluster the extracted set of patterns. We use the resultant set of clusters to represent a word pair by a feature vector. We compute a feature from each cluster as follows. We assign a weight w_{ij} to a pattern p_i that is in a cluster c_j as follows,

$$w_{ij} = \frac{\mu(p_i)}{\sum_{q \in c_j} \mu(q)}. \tag{7.5}$$

Here, $\mu(q)$ is the total frequency of a pattern, and it is given by,

$$\mu(p) = \sum_{(a,b) \in W} f(a,b,p). \tag{7.6}$$

Here, W is the set of word pairs. Because I perform a hard clustering on patterns, a pattern can belong to only one cluster (i.e. $w_{ij} = 0$ for $p_i \notin c_j$). Finally, we compute the value of the j-th feature in the feature vector for word pair (a,b) as follows,

$$\sum_{p_i \in c_j} w_{ij} f(a,b,p_i). \tag{7.7}$$

For each set of clusters, I compute the element Λ_{ij} of the corresponding inter-cluster correlation matrix Λ as the cosine similarity between the centroid vectors for clusters c_i and c_j. The prototype vector σ in Equation 7.3 is computed as the vector sum of individual feature vectors for the synonymous word pairs selected from the WordNet as described above. We then use Equation 7.3 to compute the average of similarity scores for synonymous word pairs we selected from WordNet.

I select the θ that maximizes the average similarity score between those synonymous word pairs. Formally, the optimal value of θ, $\hat{\theta}$ is given by the following Equation,

$$\hat{\theta} = \mathrm{argmax}_{\theta \in [0,1]} \left(\frac{1}{|W|} \sum\nolimits_{(a,b) \in W} \mathrm{sim}(a,b) \right). \tag{7.8}$$

Here, W is the set of synonymous word pairs (a,b), $|W|$ is the total number of synonymous word pairs (i.e. 2000 in our experiments), and $\mathrm{sim}(a,b)$ is given by Equation 7.3. Because the averages are taken over 2000 word pairs this procedure gives a reliable estimate for θ. Moreover, this method does not require negative training instances such as, non-synonymous word pairs, which are difficult to create manually. Average similarity scores for various θ values are shown in Figure 7.1. From Figure 7.1, we see that initially average similarity increases when θ is increased. This is because clustering of semantically related patterns reduces the sparseness in feature vectors. Average similarity is stable within a wide range of θ values between 0.5 and 0.7. However, increasing θ beyond 0.7 results in a rapid drop of average similarity. To explain this behavior consider Figure 7.2 where we plot the sparsity of the set of clusters (i.e. the ratio between singletons to total clusters) against threshold θ. As seen from Figure 7.2, high θ values result in a high percentage of singletons because only highly similar patterns will form clusters. Consequently, feature vectors for different word pairs do not have many features in common. The maximum average similarity score of 1.303 is obtained with $\theta = 0.7$, corresponding to $17,015$ total clusters out of which $12,476$ are singletons with exactly one pattern (sparsity = 0.733). For the remainder of the experiments in the paper we set θ to this optimal value and use the corresponding set of clusters to compute semantic similarity by Equation 7.3. Similarity scores computed using Equation 7.3 can be greater than 1 (see Figure 7.1) because of the terms corresponding to the non-diagonal elements in Λ. We do not normalize the similarity scores to $[0, 1]$ range in our experiments because the evaluation metrics we use are insensitive to linear transformations of similarity scores.

7.3 Experiments

7.3.1 Evaluation Measures

The evaluation of automatic semantic similarity measures is often performed by comparing the similarity scores produced by the semantic similarity measure under evaluation against a set of ratings assigned by human annotators for the same set of word pairs. Two datasets that have been used repeatedly in previous work on semantic similarity are the Miller-Charles dataset and the WordSimilarity-353 dataset.We have already seen the Miller-Charles dataset in Chapter 2. Two sets of scores assigned to the same set of word pairs can be efficiently summarized using correleation coefficients. Two correlations coefficients have been used in previous work: Pearson's correlation coefficient (aka Pearson's product moment coefficieint), and the Spearman rank correlation coefficient. I will next describe those correlation coefficients in detail.

The Pearson correlation coefficient is given by,

$$r = \frac{1}{(n-1)} \sum_{i=1}^{n} \left(\frac{X_i - \bar{X}}{s_x} \right) \left(\frac{Y_i - \bar{Y}}{s_y} \right). \tag{7.9}$$

Here, X and Y are the two variables between which we must compute the correlation, X_i and Y_i respectively, denote the values of the variables X and Y, n is the total number of data points, $\bar{X}$ and $\bar{Y}$ respectively, denote the sample means of the variables X and Y, s_x and s_y are respectively, the sample means of the variables X and Y.

Spearman rank correlation can be considered as a special case of Pearson correlation where we first convert the values of varibales X and Y into two set of rankings respectively denoted by x and y. If two or more data points have identical values, then the average of ranks of those data points is assigned. For example, consider the five data points, 2.3, 2.4, 2.8, 2.8, and 3.0. The ranks assigned for those five points are 1, 2, 3.5, 3.5, 5. Then the following equation can be used to compute the Spearman rank correlation coefficient (ρ) between the two variables,

$$\rho = \frac{n(\sum_{i=1}^{n} x_i y_i) - (\sum_{i=1}^{n} x_i)(\sum_{i=1}^{n} y_i)}{\sqrt{n(\sum_{i=1}^{n} x_i^2) - (\sum_{i=1}^{n} x_i)^2}\sqrt{n(\sum_{i=1}^{n} x_i^2) - (\sum_{i=1}^{n} x_i)^2}}. \tag{7.10}$$

Here, x_i, y_i denote the actual values of the ranks. It should be noted that if there are no equal ranks, then the following simple formula can be used to compute the Spearman rank correlation coefficient,

$$\rho = 1 - \frac{6\sum_{i=1}^{n} d_i^2}{n(n^2-1)}, \tag{7.11}$$

where, $d_i = x_i - y_i$ is the difference between corresponding ranks. Both Pearson and Spearman correlation coefficients are in the range $[-1, +1]$.

Computation of confidence intervals for correlation coefficients is often performed using Fisher's transformation, $F(r)$, which is given by,

$$F(r) = \frac{1}{2}\log\left(\frac{1+r}{1-r}\right). \tag{7.12}$$

The corresponding z score is given by,

$$z = \sqrt{n-3}F(r), \tag{7.13}$$

(It is typically assumed that there are at least more than three data points. If the number of data points is small then statistical significance tests are not reliable.) If we want to compute the 95% confidence intervals, then the corresponding lower bound is $z_L = z + 1.96$, and the upper bound is $z_U = z - 1.96$, where z is given by Formula 7.13. The upper and lower bounds on z score can the be converted back to confidence intervals on r by using the inverse of the Fisher transformation, given by

$$r = \frac{\exp(2F(r)) - 1}{\exp(2F(r)) + 1}. \tag{7.14}$$

We plug in the lower and upper bounds z_L, z_U in Formula 7.13 to compute the corresponding Fisher function values, and then substitiute those values in Formula 7.14 to compute the lower and upper bounds of correlation r. If the correlation between a similarity measure A and human ratings H falls outside the confidence interval computed for the correlation between a different similarity measure B and human ratings, then we can conclude that A and B are statistically significant (i.e. the probability of two methods being the same is less than 0.05).

7.3.2 Comaprisons with benchmark datasets

Table 2.4 compares the proposed relational model (RM) against Miller-Charles ratings (MC), and previously proposed web-based semantic similarity measures: Jaccard (Jacc), Dice, Overlap (Over), PMI [18], Normalized Google Distance (NGD) [29], Sahami and

Heilman (SH) [131], co-occurrence double checking model (CO) [26], and support vector machine-based (SVM) approach described in Chapter 2. The bottom row of Table 7.1 shows the Pearson correlation coefficient of similarity scores produced by each algorithm with MC. All similarity scores, except for the human-ratings in Miller-Charles dataset, are normalized to $[0, 1]$ range for the ease of comparison. It is noteworthy that the Pearson correlation coefficient is invariant under a linear transformation. All baselines as well as previous methods compared in Table 7.1 are reimplemented following the descriptions in the original papers. SH and NGD methods were not evaluated on Miller-Charles or Word-Similarity datasets by the authors of the original papers. Moreover, those methods are not publicly available to download. Therefore, a reimplementation was necessary to compare them on the same benchmark datasets as often used by the previous work on semantic similarity.

Table 7.1: Experimental results on Miller-Charles dataset.

word pair	MC	Jacc	Dice	Over	PMI	CO	SH	NGD	SVM	RM
automobile-car	1.00	0.65	0.66	0.83	0.43	0.69	1.00	0.15	0.98	0.92
journey-voyage	0.98	0.41	0.42	0.16	0.47	0.42	0.52	0.39	1.00	1.00
gem-jewel	0.98	0.29	0.30	0.07	0.69	1.00	0.21	0.42	0.69	0.82
boy-lad	0.96	0.18	0.19	0.59	0.63	0.00	0.47	0.12	0.97	0.96
coast-shore	0.94	0.78	0.79	0.51	0.56	0.52	0.38	0.52	0.95	0.97
asylum-madhouse	0.92	0.01	0.01	0.08	0.81	0.00	0.21	1.00	0.77	0.79
magician-wizard	0.89	0.29	0.30	0.37	0.86	0.67	0.23	0.44	1.00	1.00
midday-noon	0.87	0.10	0.10	0.12	0.59	0.86	0.29	0.74	0.82	0.99
furnace-stove	0.79	0.39	0.41	0.10	1.00	0.93	0.31	0.61	0.89	0.88
food-fruit	0.78	0.75	0.76	1.00	0.45	0.34	0.18	0.55	1.00	0.94
bird-cock	0.77	0.14	0.15	0.14	0.43	0.50	0.06	0.41	0.59	0.87
bird-crane	0.75	0.23	0.24	0.21	0.52	0.00	0.22	0.41	0.88	0.85
implement-tool	0.75	1.00	1.00	0.51	0.30	0.42	0.42	0.91	0.68	0.50
brother-monk	0.71	0.25	0.27	0.33	0.62	0.55	0.27	0.23	0.38	0.27

word pair	MC	Jacc	Dice	Over	PMI	CO	SH	NGD	SVM	RM
crane-implement	0.42	0.06	0.06	0.10	0.19	0.00	0.15	0.40	0.13	0.06
brother-lad	0.41	0.18	0.19	0.36	0.64	0.38	0.24	0.26	0.34	0.13
car-journey	0.28	0.44	0.45	0.36	0.20	0.29	0.19	0.00	0.29	0.17
monk-oracle	0.27	0.00	0.00	0.00	0.00	0.00	0.05	0.45	0.33	0.80
food-rooster	0.21	0.00	0.00	0.41	0.21	0.00	0.08	0.42	0.06	0.02
coast-hill	0.21	0.96	0.97	0.26	0.35	0.00	0.29	0.70	0.87	0.36
forest-graveyard	0.20	0.06	0.06	0.23	0.49	0.00	0.00	0.54	0.55	0.44
monk-slave	0.12	0.17	0.18	0.05	0.61	0.00	0.10	0.77	0.38	0.24
coast-forest	0.09	0.86	0.87	0.29	0.42	0.00	0.25	0.36	0.41	0.15
lad-wizard	0.09	0.06	0.07	0.05	0.43	0.00	0.15	0.66	0.22	0.23
cord-smile	0.01	0.09	0.10	0.02	0.21	0.00	0.09	0.13	0.00	0.01
glass-magician	0.01	0.11	0.11	0.40	0.60	0.00	0.14	0.21	0.18	0.05
rooster-voyage	0.00	0.00	0.00	0.00	0.23	0.00	0.20	0.21	0.02	0.05
noon-string	0.00	0.12	0.12	0.04	0.10	0.00	0.08	0.21	0.02	0.00
Spearman	1.00	0.39	0.39	0.40	0.52	−0.36	0.62	0.13	0.83	0.85
Lower	1.00	0.02	0.02	0.04	0.18	−0.65	0.33	−0.25	0.66	0.69
Upper	1.00	0.67	0.67	0.68	0.75	0.02	0.81	0.48	0.92	0.93
Pearson	1.00	0.26	0.27	0.38	0.55	0.69	0.58	0.21	0.83	0.87
Lower	1.00	−0.13	−0.12	0.01	0.22	0.43	0.26	−0.18	0.67	0.73
Upper	1.00	0.58	0.58	0.66	0.77	0.85	0.78	0.54	0.92	0.94

The highest correlation is reported by the proposed semantic similarity measure. The improvement of the proposed method is statistically significant (confidence interval $[0.73, 0.94]$) against all the similarity measures compared in Table 7.1, except against the SVM approach. From Table 7.1 we see that measures that use contextual information from snippets (e.g. SH, CO, SVM, and proposed) outperform the ones that use only co-occurrence statistics (e.g. Jaccard, overlap, Dice, PMI, and NGD) such as page-counts. This is because similarity measures that use contextual information are better equipped to compute the similarity between polysemous words. Although both SVM and proposed methods (RM) use lexical patterns, unlike the proposed method, the SVM method does not consider the relatedness between patterns. The superior performance of the proposed method can be attributable to its consideration of relatedness of patterns. It is interesting to see the significantly different performances reported by the Pearson and Spearman coefficients for the CO method. When the CO method is evaluated using Pearson correlation coefficient we see a high (0.69) correlation between human ratings and the similarity scores produced by the CO method. However, the evaluation results reported by the Spearman rank correlation coefficient shows a highly negative (-0.36) correlation between human ratings and the similarity scores produced by the CO method. This is because the CO method returns 0 for most word pairs in the Miller-Charles dataset. In fact, for 15 out of the 28 word pairs in the Miller-Charles dataset, the CO method returns a similarity score of 0. Consequently, the relationship between human scores and similarity scores produced by the CO method is not linear. As described earlier, Pearson correlation coefficient can be misleading in such situations. The main reason that the CO method returns zero similarity scores for most word pairs is that, although two words are semantically similar, search engines do not rank documents that are always semantically similar to a query. A search engine ranking algorithm is often more complex and uses various other factors such as static rank (i.e. Page rank, HITS, etc.), novelty of the content (on-line news sources), authority of the site (i.e. Wikipedia articles are often ranked at the top by Google), and past click-through data.

I evaluate the proposed method using the WordSimilarity-353 dataset. Experimental results are presented in Table 7.2. Although Pearson correlation coefficient is used by previous work on semantic similarity measure to compare human ratings in the Miller-Charles dataset against similarity scores produced by a system under evaluation, Spearman rank correlation coefficient is used for evaluations using WordSimilarity-353 dataset. Pearson correlation can be misleading when there does not exist a linear relationship between the two variables between which we compute correlation. The same data points can be arranged in both linear and non-linear fashion to produce the same Pearson correlation. In contrast, Spearman rank correlation coefficient first ranks the two sets of values to be compared, and then measure the correlation between the two sets of ranks ignoring the actual values. The WordSimilarity-353 dataset contains a large number of data points compared to the Miller-Charles dataset and we are only interested in the relative orderings of the word pairs. For those reasons, previous work evaluating semantic similarity measures using the WordSimilarity-353 dataset have used Spearman rank correlation coefficient instead of Pearson correlation coefficient. In Table 7.2, I have used both Pearson correlation coefficient and Spearman rank correlation coefficient to measure the correlation between human ratings and each set of similarity scores produced by the different algorithms.

Table 7.2: Experimental results on WordSimilarity-353 dataset.

word pair	WS	Jacc	Dice	Over	PMI	CO	SH	NGD	SVM	RM
tiger-tiger	1.00	1.00	1.00	0.48	0.96	1.00	1.00	1.00	1.00	0.57
fuck-sex	0.94	0.09	0.17	0.30	0.69	0.00	0.02	0.59	0.88	0.42
midday-noon	0.93	0.01	0.03	0.03	0.67	0.32	0.05	0.42	0.87	0.82
journey-voyage	0.93	0.02	0.05	0.03	0.55	0.00	0.02	0.41	0.86	0.83
dollar-buck	0.92	0.06	0.12	0.14	0.71	0.10	0.01	0.57	0.78	0.46
money-cash	0.91	0.10	0.19	0.21	0.53	0.20	0.03	0.52	0.88	0.89
coast-shore	0.91	0.06	0.12	0.12	0.66	0.23	0.05	0.55	0.87	0.79
money-cash	0.91	0.10	0.19	0.21	0.53	0.20	0.03	0.52	0.88	0.89
money-currency	0.90	0.04	0.07	0.15	0.48	0.24	0.03	0.37	0.98	0.84
football-soccer	0.90	0.24	0.38	0.28	0.68	0.27	0.04	0.73	1.00	0.84
magician-wizard	0.90	0.02	0.05	0.07	0.78	0.25	0.02	0.51	1.00	0.97
type-kind	0.89	0.09	0.17	0.14	0.45	0.10	0.01	0.46	0.92	0.89
gem-jewel	0.89	0.05	0.10	0.06	0.78	0.55	0.02	0.62	0.63	0.92
car-automobile	0.89	0.06	0.11	0.32	0.61	0.31	0.05	0.50	0.98	1.00
street-avenue	0.89	0.10	0.18	0.23	0.58	0.17	0.01	0.55	0.99	0.78
asylum-madhouse	0.88	0.00	0.00	0.02	0.67	0.00	0.03	0.31	0.95	0.64
boy-lad	0.88	0.01	0.03	0.14	0.63	0.26	0.04	0.39	0.72	0.78
environment-ecology	0.88	0.04	0.07	0.27	0.66	0.33	0.03	0.48	0.98	0.76
furnace-stove	0.88	0.04	0.07	0.05	0.87	0.40	0.05	0.61	0.98	1.00
seafood-lobster	0.87	0.05	0.10	0.10	0.83	0.65	0.04	0.61	0.98	0.85
mile-kilometer	0.86	0.01	0.01	0.03	0.52	0.00	0.04	0.28	0.99	0.73
Maradona-football	0.86	0.00	0.00	0.08	0.51	0.00	0.04	0.23	0.11	0.58

word pair	WS	Jacc	Dice	Over	PMI	CO	SH	NGD	SVM	RM
OPEC-oil	0.86	0.02	0.04	0.30	0.74	0.50	0.04	0.47	0.85	0.51
king-queen	0.85	0.10	0.19	0.15	0.62	0.23	0.02	0.59	0.96	0.69
murder-manslaughter	0.85	0.02	0.04	0.19	0.82	0.53	0.13	0.50	0.98	0.88
money-bank	0.85	0.12	0.22	0.18	0.51	0.19	0.04	0.54	0.74	0.48
computer-software	0.85	0.19	0.31	0.18	0.49	0.20	0.03	0.63	0.95	0.87
vodka-gin	0.84	0.07	0.13	0.06	0.93	0.60	0.03	0.71	1.00	0.92
Jerusalem-Israel	0.84	0.09	0.16	0.27	0.76	0.43	0.02	0.62	0.60	0.77
planet-star	0.84	0.07	0.13	0.15	0.52	0.19	0.02	0.47	0.93	0.95
calculation-computation	0.84	0.07	0.14	0.11	0.92	0.00	0.02	0.68	0.96	0.78
money-dollar	0.84	0.11	0.19	0.32	0.58	0.18	0.02	0.55	1.00	0.70
law-lawyer	0.83	0.12	0.21	0.36	0.69	0.52	0.08	0.61	0.94	0.00
championship-tournament	0.83	0.12	0.21	0.10	0.73	0.34	0.05	0.70	0.49	0.48
weather-forecast	0.83	0.17	0.28	0.35	0.66	0.54	0.06	0.65	1.00	0.84
seafood-food	0.83	0.04	0.08	0.37	0.62	0.33	0.03	0.47	1.00	0.90
network-hardware	0.83	0.10	0.18	0.16	0.49	0.18	0.03	0.50	0.65	0.63
nature-environment	0.83	0.11	0.19	0.10	0.53	0.19	0.03	0.58	1.00	0.78
FBI-investigation	0.83	0.07	0.13	0.10	0.74	0.41	0.05	0.60	0.30	0.58
man-woman	0.83	0.14	0.24	0.23	0.54	0.26	0.02	0.57	1.00	0.00
money-wealth	0.82	0.04	0.08	0.25	0.55	0.23	0.04	0.43	0.96	0.80
psychology-Freud	0.82	0.01	0.03	0.05	0.71	0.29	0.01	0.44	0.10	0.61
news-report	0.81	0.23	0.37	0.33	0.42	0.13	0.02	0.56	0.97	0.33
war-troops	0.81	0.06	0.11	0.27	0.65	0.23	0.03	0.52	0.82	0.60
vodka-brandy	0.81	0.05	0.09	0.04	0.88	0.36	0.03	0.65	0.93	0.91
Harvard-Yale	0.81	0.07	0.13	0.12	0.84	0.34	0.05	0.64	0.42	0.78
physics-proton	0.81	0.02	0.03	0.04	0.66	0.25	0.01	0.43	0.61	0.55
bank-money	0.81	0.15	0.25	0.21	0.53	0.19	0.04	0.58	0.91	0.70
planet-galaxy	0.81	0.04	0.08	0.10	0.65	0.19	0.01	0.50	0.96	0.56
stock-market	0.80	0.14	0.24	0.14	0.52	0.38	0.03	0.60	0.66	0.52
psychology-psychiatry	0.80	0.07	0.13	0.15	0.85	0.28	0.07	0.63	0.99	0.72
planet-moon	0.80	0.06	0.10	0.05	0.56	0.22	0.01	0.52	0.54	0.87

word pair	WS	Jacc	Dice	Over	PMI	CO	SH	NGD	SVM	RM
planet-constellation	0.80	0.01	0.02	0.11	0.65	0.23	0.01	0.38	0.95	0.60
credit-card	0.80	0.24	0.38	0.19	0.59	0.48	0.02	0.75	0.96	0.51
hotel-reservation	0.80	0.15	0.26	0.53	0.72	0.50	0.07	0.65	0.99	0.57
planet-sun	0.80	0.05	0.09	0.08	0.48	0.19	0.02	0.40	0.30	0.42
tiger-jaguar	0.80	0.04	0.07	0.04	0.63	0.00	0.01	0.51	0.97	0.73
tiger-feline	0.80	0.01	0.01	0.03	0.60	0.24	0.01	0.32	0.05	0.47
closet-clothes	0.80	0.06	0.11	0.11	0.74	0.00	0.04	0.57	0.38	0.49
soap-opera	0.79	0.04	0.08	0.08	0.63	0.27	0.00	0.50	0.25	0.67
planet-astronomer	0.79	0.01	0.03	0.27	0.78	0.46	0.03	0.46	0.32	0.40
planet-space	0.79	0.06	0.12	0.11	0.51	0.22	0.01	0.46	1.00	0.62
movie-theater	0.79	0.18	0.30	0.30	0.64	0.24	0.02	0.66	0.90	0.56
treatment-recovery	0.79	0.06	0.11	0.07	0.58	0.27	0.02	0.53	0.88	0.59
liquid-water	0.78	0.04	0.08	0.17	0.57	0.20	0.00	0.45	0.51	0.71
life-death	0.78	0.15	0.26	0.30	0.53	0.17	0.01	0.57	1.00	0.68
baby-mother	0.78	0.12	0.21	0.17	0.56	0.15	0.02	0.58	0.99	0.77
aluminum-metal	0.78	0.05	0.09	0.14	0.64	0.31	0.01	0.50	0.48	0.90
lobster-food	0.78	0.01	0.02	0.24	0.56	0.00	0.02	0.32	0.92	0.68
cell-phone	0.78	0.15	0.25	0.27	0.54	0.42	0.02	0.58	0.93	0.54
dollar-yen	0.77	0.10	0.18	0.19	0.74	0.43	0.04	0.64	0.99	0.71
wood-forest	0.77	0.00	0.00	0.00	0.00	0.24	0.01	1.00	0.07	0.81
money-deposit	0.77	0.03	0.06	0.24	0.54	0.20	0.03	0.39	0.36	0.24
television-film	0.77	0.53	0.69	0.42	0.67	0.16	0.02	0.86	1.00	0.74
psychology-mind	0.76	0.04	0.07	0.12	0.58	0.23	0.01	0.44	0.78	0.76
game-team	0.76	0.24	0.38	0.19	0.53	0.13	0.02	0.71	0.97	0.44
admission-ticket	0.76	0.06	0.11	0.09	0.66	0.00	0.03	0.55	0.41	0.50
Jerusalem-Palestinian	0.76	0.10	0.19	0.09	0.85	0.38	0.02	0.73	0.37	0.70
Arafat-terror	0.76	0.01	0.02	0.10	0.77	0.38	0.03	0.45	0.14	0.50
profit-loss	0.76	0.06	0.11	0.07	0.54	0.20	0.01	0.50	0.99	0.63
dividend-payment	0.76	0.02	0.04	0.10	0.62	0.00	0.02	0.41	0.18	0.74
computer-keyboard	0.76	0.05	0.09	0.26	0.58	0.21	0.02	0.46	0.55	0.75

word pair	WS	Jacc	Dice	Over	PMI	CO	SH	NGD	SVM	RM
boxing-round	0.76	0.04	0.09	0.10	0.58	0.23	0.01	0.47	0.33	0.21
rock-jazz	0.75	0.13	0.23	0.22	0.61	0.18	0.04	0.61	0.99	0.85
century-year	0.75	0.08	0.14	0.27	0.50	0.10	0.01	0.46	0.74	0.66
computer-internet	0.75	0.15	0.25	0.19	0.45	0.17	0.03	0.53	0.99	0.59
money-property	0.75	0.11	0.19	0.13	0.46	0.11	0.02	0.50	1.00	0.80
tennis-racket	0.75	0.01	0.02	0.18	0.69	0.56	0.02	0.41	0.30	0.75
announcement-news	0.75	0.06	0.12	0.99	0.57	0.13	0.02	0.47	1.00	0.55
day-dawn	0.75	0.02	0.04	0.34	0.50	0.15	0.01	0.33	0.36	0.30
canyon-landscape	0.75	0.02	0.04	0.03	0.59	0.00	0.01	0.43	0.22	0.72
food-fruit	0.75	0.07	0.12	0.29	0.59	0.24	0.02	0.50	0.99	0.84
telephone-communication	0.74	0.31	0.48	0.24	0.72	0.00	0.01	0.83	0.99	0.78
currency-market	0.74	0.07	0.13	0.20	0.57	0.21	0.04	0.50	0.50	0.45
psychology-cognition	0.74	0.03	0.06	0.18	0.87	0.48	0.04	0.57	0.97	0.63
seafood-sea	0.74	0.02	0.03	0.07	0.52	0.41	0.01	0.34	0.99	0.59
marathon-sprint	0.74	0.02	0.04	0.02	0.59	0.00	0.01	0.43	0.71	0.42
book-paper	0.74	0.16	0.27	0.21	0.53	0.18	0.01	0.59	0.87	0.77
book-library	0.74	0.14	0.24	0.19	0.51	0.19	0.01	0.56	0.66	0.67
Mexico-Brazil	0.74	0.24	0.38	0.28	0.66	0.00	0.05	0.73	0.89	0.77
psychology-depression	0.74	0.05	0.09	0.05	0.68	0.17	0.01	0.57	0.94	0.41
media-radio	0.74	0.13	0.22	0.16	0.43	0.15	0.02	0.49	1.00	0.69
jaguar-cat	0.74	0.01	0.01	0.02	0.37	0.17	0.00	0.17	0.39	0.70
fighting-defeating	0.73	0.14	0.24	1.00	1.00	0.00	0.02	0.72	1.00	0.53
movie-star	0.73	0.32	0.48	0.25	0.59	0.21	0.02	0.78	0.98	0.59
hundred-percent	0.73	0.04	0.08	0.07	0.59	0.16	0.01	0.47	0.18	0.22
dollar-profit	0.73	0.12	0.22	0.11	0.66	0.20	0.03	0.67	0.55	0.46
bird-crane	0.73	0.02	0.04	0.07	0.65	0.20	0.01	0.45	0.60	0.93
tiger-cat	0.73	0.03	0.05	0.07	0.50	0.23	0.00	0.36	0.43	0.69
physics-chemistry	0.73	0.20	0.33	0.16	0.85	0.26	0.08	0.80	1.00	0.84
country-citizen	0.72	0.08	0.14	0.34	0.61	0.08	0.01	0.52	0.72	0.56
money-possession	0.72	0.01	0.03	0.19	0.52	0.00	0.02	0.32	0.59	0.78

word pair	WS	Jacc	Dice	Over	PMI	CO	SH	NGD	SVM	RM
jaguar-car	0.72	0.05	0.09	0.29	0.60	0.00	0.03	0.47	0.55	0.51
cup-drink	0.72	0.06	0.11	0.06	0.53	0.17	0.01	0.50	0.40	0.57
psychology-health	0.72	0.02	0.05	0.25	0.51	0.12	0.02	0.35	0.55	0.85
museum-theater	0.71	0.12	0.21	0.11	0.63	0.16	0.02	0.65	0.90	0.85
summer-drought	0.71	0.01	0.02	0.15	0.55	0.00	0.01	0.31	0.24	0.58
phone-equipment	0.71	0.08	0.15	0.14	0.45	0.09	0.03	0.45	0.48	0.50
investor-earning	0.71	0.18	0.31	0.42	0.86	0.29	0.03	0.73	0.81	0.25
bird-cock	0.70	0.02	0.03	0.04	0.57	0.28	0.01	0.37	0.20	0.72
tiger-carnivore	0.70	0.00	0.00	0.05	0.64	0.30	0.02	0.30	1.00	0.34
company-stock	0.70	0.13	0.23	0.21	0.48	0.17	0.03	0.53	0.77	0.53
stroke-hospital	0.70	0.04	0.07	0.10	0.63	0.22	0.04	0.48	0.33	0.48
liability-insurance	0.70	0.05	0.09	0.15	0.60	0.35	0.03	0.48	0.41	0.66
game-victory	0.70	0.07	0.13	0.25	0.57	0.00	0.01	0.49	0.65	0.19
tiger-animal	0.69	0.05	0.09	0.07	0.58	0.22	0.01	0.48	0.59	0.69
psychology-anxiety	0.69	0.05	0.09	0.06	0.72	0.23	0.01	0.58	0.83	0.59
doctor-nurse	0.69	0.09	0.17	0.17	0.72	0.21	0.03	0.62	0.99	0.76
game-defeat	0.69	0.04	0.08	0.36	0.62	0.00	0.02	0.47	0.37	0.23
FBI-fingerprint	0.69	0.02	0.03	0.05	0.74	0.33	0.02	0.46	0.10	0.35
street-block	0.68	0.08	0.15	0.13	0.50	0.11	0.01	0.49	0.59	0.35
opera-performance	0.68	0.03	0.07	0.06	0.46	0.17	0.01	0.36	0.28	0.69
money-withdrawal	0.68	0.01	0.02	0.19	0.52	0.00	0.01	0.30	0.40	0.59
drink-eat	0.68	0.15	0.26	0.14	0.68	0.32	0.06	0.70	1.00	0.92
tiger-mammal	0.68	0.01	0.02	0.07	0.70	0.27	0.01	0.40	0.84	0.49
psychology-fear	0.68	0.03	0.06	0.05	0.59	0.00	0.00	0.46	0.04	0.22
drug-abuse	0.68	0.06	0.11	0.08	0.53	0.36	0.02	0.48	0.35	0.49
cup-tableware	0.68	0.00	0.01	0.09	0.59	0.00	0.03	0.31	0.96	0.83
student-professor	0.67	0.12	0.21	0.18	0.64	0.00	0.03	0.62	1.00	0.77
football-basketball	0.67	0.28	0.43	0.32	0.70	0.23	0.07	0.76	1.00	0.77
concert-virtuoso	0.67	0.01	0.01	0.10	0.67	0.00	0.05	0.36	0.14	0.25
computer-laboratory	0.67	0.03	0.06	0.16	0.52	0.10	0.02	0.39	0.27	0.54

word pair	WS	Jacc	Dice	Over	PMI	CO	SH	NGD	SVM	RM
television-radio	0.67	0.54	0.70	0.48	0.67	0.17	0.03	0.85	1.00	0.71
love-sex	0.67	0.10	0.18	0.15	0.47	0.18	0.01	0.49	0.98	0.79
problem-challenge	0.67	0.14	0.24	0.18	0.59	0.13	0.00	0.61	0.81	0.74
movie-critic	0.67	0.04	0.08	0.42	0.68	0.35	0.05	0.51	0.63	0.67
Arafat-peace	0.67	0.01	0.01	0.18	0.71	0.29	0.02	0.39	0.18	0.51
bed-closet	0.66	0.04	0.07	0.14	0.67	0.00	0.02	0.49	0.85	0.86
psychology-science	0.66	0.05	0.09	0.27	0.61	0.19	0.02	0.48	0.92	0.60
lawyer-evidence	0.66	0.08	0.14	0.10	0.67	0.00	0.04	0.59	0.50	0.23
fertility-egg	0.66	0.03	0.05	0.05	0.72	0.38	0.01	0.50	0.08	0.73
bishop-rabbi	0.66	0.01	0.03	0.03	0.70	0.00	0.01	0.44	0.58	0.70
precedent-law	0.66	0.01	0.02	0.06	0.44	0.00	0.05	0.25	0.29	0.76
minister-party	0.66	0.07	0.13	0.15	0.55	0.12	0.01	0.50	0.96	0.65
football-tennis	0.66	0.22	0.35	0.24	0.66	0.14	0.04	0.72	1.00	0.82
professor-doctor	0.65	0.05	0.10	0.06	0.58	0.10	0.02	0.52	0.55	0.91
psychology-clinic	0.65	0.08	0.15	0.08	0.75	0.23	0.01	0.66	0.62	0.20
cup-coffee	0.65	0.05	0.10	0.06	0.53	0.31	0.01	0.48	0.19	0.28
water-seepage	0.65	0.00	0.00	0.36	0.67	0.00	0.01	0.28	0.53	0.79
government-crisis	0.65	0.09	0.17	0.15	0.55	0.12	0.02	0.53	0.60	0.39
space-world	0.64	0.11	0.20	0.21	0.43	0.11	0.02	0.46	0.98	0.72
Japanese-American	0.64	0.07	0.12	0.12	0.47	0.14	0.01	0.43	0.28	0.64
dividend-calculation	0.64	0.04	0.08	0.04	0.80	0.25	0.01	0.62	0.17	0.48
victim-emergency	0.64	0.07	0.13	0.11	0.68	0.15	0.03	0.58	0.11	0.32
luxury-car	0.64	0.07	0.12	0.19	0.54	0.26	0.02	0.47	0.42	0.50
tool-implement	0.64	0.16	0.27	0.35	0.75	0.16	0.04	0.68	0.90	0.25
street-place	0.64	0.13	0.23	0.15	0.45	0.11	0.01	0.52	0.79	0.72
competition-price	0.64	0.06	0.10	0.16	0.47	0.13	0.01	0.40	0.49	0.79
psychology-doctor	0.63	0.03	0.05	0.05	0.56	0.00	0.01	0.42	0.01	0.27
gender-equality	0.63	0.04	0.07	0.11	0.69	0.50	0.02	0.51	0.38	0.53
listing-category	0.63	0.54	0.70	0.70	0.69	0.10	0.02	0.82	1.00	0.11
video-archive	0.63	0.21	0.34	0.32	0.46	0.11	0.05	0.58	0.93	0.58

word pair	WS	Jacc	Dice	Over	PMI	CO	SH	NGD	SVM	RM
oil-stock	0.63	0.09	0.17	0.11	0.54	0.12	0.01	0.55	0.48	0.55
governor-office	0.63	0.03	0.06	0.21	0.52	0.17	0.01	0.37	0.04	0.17
discovery-space	0.63	0.03	0.06	0.09	0.49	0.23	0.02	0.37	0.65	0.17
train-car	0.62	0.08	0.16	0.20	0.55	0.18	0.01	0.51	0.66	0.89
shower-thunderstorm	0.62	0.06	0.11	0.19	0.85	0.00	0.00	0.61	1.00	0.60
record-number	0.62	0.13	0.23	0.19	0.49	0.08	0.01	0.54	0.51	0.37
brother-monk	0.62	0.03	0.06	0.13	0.70	0.20	0.04	0.49	0.45	0.75
production-crew	0.62	0.04	0.07	0.06	0.50	0.19	0.01	0.40	0.20	0.82
nature-man	0.62	0.09	0.16	0.13	0.47	0.14	0.02	0.46	0.21	0.34
family-planning	0.62	0.07	0.13	0.14	0.45	0.15	0.01	0.42	0.39	0.50
disaster-area	0.62	0.03	0.06	0.20	0.51	0.12	0.01	0.37	0.35	0.56
skin-eye	0.61	0.10	0.19	0.10	0.57	0.15	0.06	0.61	0.76	0.64
food-preparation	0.61	0.04	0.07	0.14	0.49	0.17	0.01	0.38	0.31	0.59
preservation-world	0.61	0.01	0.01	0.19	0.42	0.10	0.01	0.20	0.07	0.21
movie-popcorn	0.61	0.01	0.03	0.20	0.58	0.17	0.01	0.36	0.59	0.63
lover-quarrel	0.61	0.01	0.01	0.07	0.76	0.00	0.03	0.41	0.00	0.19
game-series	0.61	0.18	0.30	0.17	0.52	0.13	0.02	0.64	0.84	0.50
bread-butter	0.61	0.12	0.22	0.12	0.83	0.29	0.02	0.72	1.00	0.51
dollar-loss	0.60	0.08	0.14	0.09	0.58	0.00	0.00	0.55	0.58	0.60
weapon-secret	0.60	0.07	0.13	0.20	0.71	0.29	0.01	0.57	0.52	0.41
precedent-antecedent	0.59	0.00	0.01	0.09	0.80	0.00	0.03	0.40	1.00	0.71
shower-flood	0.59	0.01	0.03	0.02	0.54	0.00	0.00	0.36	0.10	0.57
registration-arrangement	0.59	0.06	0.11	0.23	0.66	0.00	0.01	0.52	0.39	0.45
arrival-hotel	0.59	0.05	0.09	0.18	0.58	0.00	0.02	0.46	0.43	0.33
announcement-warning	0.59	0.06	0.11	0.08	0.61	0.11	0.01	0.52	0.96	0.80
game-round	0.59	0.11	0.19	0.18	0.52	0.15	0.01	0.53	0.60	0.65
baseball-season	0.59	0.12	0.21	0.14	0.60	0.20	0.01	0.62	0.65	0.32
drink-mouth	0.59	0.07	0.14	0.10	0.64	0.12	0.01	0.57	0.66	0.17
life-lesson	0.58	0.05	0.10	0.46	0.58	0.13	0.01	0.47	0.78	0.48
grocery-money	0.58	0.03	0.05	0.18	0.51	0.00	0.02	0.36	0.46	0.23

word pair	WS	Jacc	Dice	Over	PMI	CO	SH	NGD	SVM	RM
energy-crisis	0.58	0.07	0.13	0.09	0.52	0.16	0.01	0.49	0.46	0.40
king-rook	0.58	0.00	0.00	0.13	0.59	0.30	0.02	0.27	0.90	0.68
cucumber-potato	0.58	0.08	0.14	0.18	0.97	0.24	0.04	0.68	0.89	0.77
reason-criterion	0.58	0.08	0.14	0.86	0.86	0.00	0.01	0.63	1.00	0.40
equipment-maker	0.58	0.04	0.08	0.10	0.54	0.12	0.02	0.43	0.23	0.50
cup-liquid	0.58	0.02	0.04	0.05	0.51	0.00	0.01	0.36	0.09	0.31
deployment-withdrawal	0.58	0.02	0.03	0.02	0.65	0.00	0.01	0.46	0.98	0.72
tiger-zoo	0.58	0.03	0.06	0.03	0.58	0.30	0.01	0.47	0.15	0.59
precedent-example	0.58	0.01	0.02	0.05	0.45	0.00	0.01	0.26	0.78	0.61
journey-car	0.58	0.04	0.07	0.13	0.49	0.00	0.01	0.37	0.29	0.16
smart-stupid	0.57	0.03	0.06	0.06	0.53	0.00	0.01	0.41	0.99	0.73
plane-car	0.57	0.05	0.10	0.23	0.57	0.00	0.01	0.47	0.99	0.83
planet-people	0.56	0.04	0.08	0.21	0.44	0.11	0.01	0.35	0.65	0.71
lobster-wine	0.56	0.02	0.03	0.11	0.66	0.00	0.01	0.42	0.18	0.65
money-laundering	0.55	0.01	0.01	0.45	0.63	0.50	0.02	0.34	0.47	0.54
summer-nature	0.55	0.06	0.11	0.06	0.44	0.08	0.02	0.42	0.20	0.63
OPEC-country	0.55	0.00	0.01	0.18	0.52	0.14	0.01	0.24	0.33	0.14
Mars-scientist	0.55	0.03	0.06	0.06	0.62	0.22	0.02	0.46	0.16	0.43
decoration-valor	0.55	0.00	0.00	0.00	0.24	0.11	0.02	0.00	0.01	0.15
tiger-fauna	0.55	0.00	0.01	0.01	0.45	0.17	0.02	0.22	0.83	0.29
psychology-discipline	0.55	0.06	0.11	0.06	0.72	0.23	0.01	0.61	0.08	0.50
glass-metal	0.55	0.09	0.17	0.10	0.59	0.16	0.01	0.58	0.97	0.74
alcohol-chemistry	0.54	0.02	0.04	0.02	0.54	0.16	0.01	0.38	0.72	0.53
disability-death	0.54	0.03	0.07	0.11	0.57	0.00	0.01	0.43	0.96	0.86
change-attitude	0.53	0.04	0.08	0.26	0.56	0.18	0.00	0.43	0.45	0.39
arrangement-accommodation	0.53	0.06	0.11	0.08	0.70	0.00	0.03	0.57	0.30	0.41
territory-surface	0.52	0.02	0.05	0.03	0.52	0.00	0.00	0.40	0.02	0.39
size-prominence	0.52	0.00	0.00	0.11	0.44	0.00	0.01	0.15	0.99	0.63
exhibit-memorabilia	0.52	0.01	0.02	0.01	0.52	0.00	0.02	0.34	0.10	0.32
credit-information	0.52	0.07	0.13	0.21	0.38	0.10	0.01	0.33	0.59	0.41

word pair	WS	Jacc	Dice	Over	PMI	CO	SH	NGD	SVM	RM
territory-kilometer	0.52	0.00	0.01	0.01	0.44	0.00	0.02	0.19	0.02	0.25
man-governor	0.51	0.03	0.06	0.21	0.53	0.00	0.01	0.39	0.54	0.42
death-row	0.51	0.04	0.08	0.09	0.53	0.32	0.01	0.42	0.27	0.48
doctor-liability	0.51	0.02	0.05	0.04	0.53	0.09	0.02	0.39	0.02	0.34
impartiality-interest	0.50	0.01	0.02	0.98	0.84	0.00	0.02	0.46	1.00	0.43
energy-laboratory	0.50	0.05	0.09	0.13	0.58	0.16	0.01	0.46	0.35	0.36
secretary-senate	0.49	0.08	0.15	0.09	0.68	0.25	0.06	0.62	0.27	0.16
death-inmate	0.49	0.02	0.04	0.40	0.74	0.33	0.02	0.48	0.08	0.19
travel-activity	0.49	0.12	0.22	0.22	0.51	0.00	0.01	0.54	0.80	0.73
monk-oracle	0.49	0.00	0.01	0.01	0.44	0.00	0.02	0.19	0.60	0.63
doctor-personnel	0.49	0.03	0.05	0.03	0.49	0.00	0.01	0.38	1.00	0.33
cup-food	0.49	0.06	0.11	0.11	0.46	0.09	0.01	0.41	0.18	0.36
journal-association	0.49	0.10	0.17	0.09	0.52	0.18	0.04	0.57	0.42	0.19
street-children	0.48	0.12	0.22	0.11	0.49	0.11	0.01	0.57	0.63	0.47
car-flight	0.48	0.21	0.34	0.44	0.65	0.13	0.01	0.67	1.00	0.49
space-chemistry	0.48	0.02	0.03	0.09	0.48	0.00	0.01	0.31	0.12	0.58
situation-conclusion	0.47	0.09	0.16	0.15	0.67	0.00	0.01	0.59	0.65	0.48
tiger-organism	0.46	0.00	0.01	0.02	0.53	0.00	0.01	0.26	0.20	0.17
word-similarity	0.46	0.01	0.02	0.26	0.65	0.00	0.01	0.38	0.35	0.50
peace-plan	0.46	0.04	0.08	0.11	0.45	0.11	0.01	0.36	0.32	0.59
consumer-energy	0.46	0.13	0.23	0.14	0.58	0.14	0.01	0.63	0.14	0.35
ministry-culture	0.46	0.04	0.08	0.11	0.53	0.10	0.02	0.42	0.02	0.25
hospital-infrastructure	0.45	0.03	0.05	0.04	0.50	0.00	0.02	0.39	0.21	0.63
smart-student	0.45	0.05	0.09	0.05	0.47	0.12	0.01	0.42	0.22	0.50
investigation-effort	0.45	0.08	0.14	0.10	0.64	0.00	0.03	0.58	0.34	0.66
image-surface	0.44	0.05	0.09	0.16	0.50	0.00	0.01	0.41	0.11	0.58
life-term	0.44	0.51	0.67	0.78	0.65	0.31	0.16	0.79	1.00	0.60
start-match	0.43	0.07	0.14	0.16	0.46	0.10	0.01	0.43	0.24	0.22
computer-news	0.43	0.10	0.19	0.25	0.38	0.07	0.02	0.39	0.67	0.57
board-recommendation	0.43	0.05	0.10	0.44	0.61	0.00	0.01	0.49	0.69	0.42

word pair	WS	Jacc	Dice	Over	PMI	CO	SH	NGD	SVM	RM
lad-brother	0.43	0.01	0.02	0.07	0.62	0.00	0.02	0.37	0.45	0.23
food-rooster	0.43	0.00	0.00	0.12	0.48	0.00	0.02	0.20	0.06	0.47
observation-architecture	0.42	0.02	0.03	0.04	0.54	0.00	0.01	0.36	0.41	0.42
coast-hill	0.42	0.07	0.14	0.08	0.56	0.00	0.02	0.55	0.32	0.51
deployment-departure	0.41	0.01	0.01	0.01	0.50	0.00	0.02	0.30	0.13	0.58
benchmark-index	0.41	0.01	0.02	0.11	0.45	0.13	0.01	0.24	0.20	0.54
attempt-peace	0.41	0.06	0.11	0.07	0.58	0.11	0.01	0.52	0.32	0.19
consumer-confidence	0.40	0.04	0.07	0.06	0.53	0.28	0.01	0.42	0.22	0.50
start-year	0.39	0.24	0.38	0.21	0.47	0.13	0.01	0.66	0.90	0.24
focus-life	0.39	0.08	0.14	0.17	0.45	0.10	0.01	0.43	0.77	0.22
development-issue	0.38	0.25	0.40	0.25	0.59	0.10	0.01	0.72	0.95	0.47
day-summer	0.38	0.15	0.26	0.27	0.47	0.16	0.01	0.53	0.53	0.34
theater-history	0.38	0.07	0.13	0.19	0.51	0.00	0.02	0.46	0.65	0.75
situation-isolation	0.37	0.02	0.05	0.11	0.63	0.00	0.01	0.43	0.47	0.36
profit-warning	0.37	0.03	0.06	0.03	0.47	0.14	0.01	0.39	0.18	0.59
media-trading	0.37	0.10	0.19	0.39	0.54	0.00	0.01	0.52	0.86	0.48
chance-credibility	0.37	0.02	0.04	0.16	0.64	0.00	0.02	0.43	0.28	0.34
precedent-information	0.37	0.00	0.01	0.15	0.34	0.00	0.03	0.11	0.07	0.45
architecture-century	0.36	0.06	0.10	0.06	0.56	0.14	0.01	0.51	0.30	0.38
population-development	0.36	0.08	0.14	0.19	0.55	0.13	0.01	0.50	0.96	0.56
stock-live	0.36	0.05	0.10	0.09	0.37	0.09	0.01	0.32	0.26	0.54
peace-atmosphere	0.35	0.02	0.04	0.03	0.48	0.11	0.01	0.35	0.00	0.38
morality-marriage	0.35	0.02	0.04	0.11	0.71	0.00	0.01	0.47	0.72	0.69
minority-peace	0.35	0.03	0.07	0.09	0.62	0.00	0.01	0.46	0.12	0.41
cup-object	0.35	0.02	0.03	0.02	0.41	0.00	0.00	0.28	0.11	0.45
atmosphere-landscape	0.35	0.03	0.05	0.03	0.56	0.00	0.01	0.45	0.78	0.69
report-gain	0.35	0.06	0.12	0.25	0.51	0.16	0.01	0.44	0.23	0.35
music-project	0.35	0.09	0.17	0.14	0.41	0.08	0.01	0.43	0.49	0.63
seven-series	0.34	0.06	0.12	0.10	0.48	0.13	0.01	0.44	0.44	0.38
experience-music	0.33	0.08	0.14	0.11	0.38	0.12	0.01	0.37	0.58	0.50

word pair	WS	Jacc	Dice	Over	PMI	CO	SH	NGD	SVM	RM
school-center	0.33	0.19	0.31	0.19	0.47	0.14	0.02	0.60	0.89	0.63
five-month	0.32	0.16	0.28	0.17	0.53	0.10	0.01	0.62	0.85	0.44
announcement-production	0.32	0.19	0.32	0.33	0.73	0.00	0.02	0.70	0.84	0.29
morality-importance	0.32	0.01	0.03	0.08	0.67	0.21	0.01	0.42	0.00	0.40
money-operation	0.32	0.10	0.18	0.24	0.54	0.08	0.01	0.52	0.60	0.24
delay-news	0.32	0.04	0.08	0.71	0.52	0.15	0.02	0.40	1.00	0.51
governor-interview	0.31	0.04	0.07	0.08	0.57	0.12	0.02	0.45	0.08	0.21
practice-institution	0.30	0.09	0.16	0.17	0.67	0.10	0.02	0.59	0.69	0.70
century-nation	0.30	0.09	0.16	0.09	0.57	0.00	0.02	0.57	0.39	0.31
coast-forest	0.30	0.05	0.10	0.06	0.57	0.15	0.01	0.51	0.27	0.47
shore-woodland	0.29	0.03	0.05	0.06	0.71	0.00	0.03	0.50	0.14	0.60
drink-car	0.29	0.08	0.14	0.16	0.52	0.00	0.02	0.49	0.51	0.42
president-medal	0.28	0.02	0.03	0.11	0.54	0.00	0.02	0.35	0.02	0.20
prejudice-recognition	0.28	0.03	0.06	0.09	0.73	0.00	0.02	0.51	0.72	0.24
viewer-serial	0.28	0.03	0.05	0.03	0.58	0.11	0.02	0.46	0.07	0.14
peace-insurance	0.28	0.02	0.04	0.03	0.39	0.00	0.01	0.28	0.97	0.41
Mars-water	0.28	0.01	0.03	0.03	0.35	0.00	0.00	0.18	0.08	0.28
cup-artifact	0.28	0.00	0.01	0.07	0.55	0.00	0.02	0.26	0.82	0.28
media-gain	0.27	0.04	0.08	0.21	0.46	0.09	0.01	0.37	0.30	0.19
precedent-cognition	0.26	0.00	0.00	0.01	0.51	0.00	0.03	0.22	0.00	0.04
announcement-effort	0.26	0.11	0.19	0.14	0.68	0.00	0.02	0.63	0.04	0.33
line-insurance	0.25	0.06	0.11	0.09	0.42	0.12	0.02	0.38	0.03	0.29
crane-implement	0.25	0.01	0.02	0.02	0.53	0.00	0.04	0.31	0.01	0.10
drink-mother	0.25	0.07	0.13	0.07	0.56	0.08	0.02	0.54	0.60	0.27
opera-industry	0.25	0.02	0.03	0.04	0.37	0.00	0.01	0.22	0.10	0.36
volunteer-motto	0.24	0.01	0.01	0.02	0.49	0.00	0.01	0.27	0.01	0.25
listing-proximity	0.24	0.03	0.05	0.19	0.67	0.00	0.02	0.47	0.18	0.34
precedent-collection	0.23	0.01	0.02	0.04	0.40	0.00	0.02	0.20	0.05	0.22
Arafat-Jackson	0.23	0.00	0.00	0.02	0.46	0.00	0.02	0.16	0.06	0.31
cup-article	0.22	0.06	0.12	0.15	0.46	0.08	0.02	0.42	0.23	0.47

word pair	WS	Jacc	Dice	Over	PMI	CO	SH	NGD	SVM	RM
sign-recess	0.22	0.00	0.00	0.30	0.49	0.00	0.01	0.20	0.01	0.31
problem-airport	0.22	0.04	0.08	0.07	0.47	0.00	0.00	0.38	0.31	0.27
reason-hypertension	0.21	0.01	0.01	0.08	0.53	0.00	0.01	0.29	0.03	0.30
direction-combination	0.21	0.11	0.19	0.11	0.67	0.00	0.02	0.65	0.05	0.73
Wednesday-news	0.20	0.07	0.13	0.33	0.42	0.11	0.02	0.37	0.51	0.50
cup-entity	0.20	0.01	0.01	0.02	0.41	0.00	0.01	0.20	0.09	0.26
glass-magician	0.19	0.01	0.01	0.06	0.58	0.00	0.01	0.31	0.08	0.34
cemetery-woodland	0.19	0.01	0.02	0.01	0.63	0.23	0.01	0.41	0.10	0.64
possibility-girl	0.18	0.03	0.06	0.09	0.52	0.00	0.02	0.39	0.05	0.23
cup-substance	0.17	0.01	0.02	0.04	0.47	0.00	0.00	0.28	0.02	0.26
forest-graveyard	0.17	0.01	0.01	0.06	0.62	0.00	0.01	0.34	0.16	0.86
stock-egg	0.16	0.02	0.04	0.10	0.52	0.00	0.00	0.35	0.06	0.63
month-hotel	0.16	0.08	0.14	0.08	0.43	0.00	0.02	0.45	0.09	0.35
energy-secretary	0.16	0.05	0.09	0.09	0.53	0.18	0.01	0.44	0.17	0.21
precedent-group	0.16	0.01	0.02	0.12	0.40	0.11	0.02	0.20	0.07	0.22
production-hike	0.16	0.07	0.12	0.39	0.75	0.00	0.02	0.58	0.69	0.41
stock-phone	0.14	0.07	0.12	0.10	0.41	0.00	0.01	0.38	0.62	0.40
holy-sex	0.14	0.03	0.05	0.06	0.47	0.09	0.00	0.34	0.40	0.37
stock-CD	0.11	0.04	0.08	0.05	0.38	0.10	0.01	0.33	0.13	0.41
drink-ear	0.11	0.04	0.07	0.07	0.58	0.00	0.01	0.45	0.01	0.54
delay-racism	0.10	0.01	0.03	0.03	0.59	0.00	0.02	0.38	0.02	0.09
stock-life	0.07	0.07	0.13	0.12	0.41	0.08	0.01	0.38	0.50	0.44
stock-jaguar	0.07	0.03	0.05	0.10	0.53	0.00	0.01	0.37	0.08	0.31
monk-slave	0.07	0.02	0.04	0.02	0.69	0.00	0.03	0.48	0.05	0.37
lad-wizard	0.07	0.00	0.01	0.01	0.52	0.00	0.02	0.27	0.07	0.37
sugar-approach	0.07	0.02	0.05	0.04	0.50	0.00	0.00	0.37	0.10	0.22
rooster-voyage	0.04	0.00	0.00	0.01	0.43	0.00	0.01	0.15	0.04	0.30
noon-string	0.03	0.01	0.01	0.01	0.43	0.13	0.01	0.23	0.01	0.40
chord-smile	0.03	0.01	0.02	0.05	0.64	0.00	0.01	0.39	0.29	0.26
professor-cucumber	0.01	0.00	0.00	0.02	0.45	0.00	0.01	0.16	0.09	0.65

word pair	WS	Jacc	Dice	Over	PMI	CO	SH	NGD	SVM	RM
king-cabbage	0.00	0.00	0.01	0.06	0.50	0.00	0.01	0.23	0.51	0.45
Spearman	1.00	0.26	0.26	0.27	0.36	−0.33	0.36	0.40	0.53	0.52
Lower	1.00	0.16	0.16	0.17	0.26	−0.42	0.26	0.31	0.45	0.44
Upper	1.00	0.35	0.35	0.36	0.45	−0.23	0.45	0.48	0.60	0.59
Pearson	1.00	0.22	0.24	0.19	0.34	0.51	0.19	0.40	0.52	0.49
Lower	1.00	0.12	0.14	0.09	0.25	0.43	0.08	0.31	0.44	0.41
Upper	1.00	0.32	0.34	0.29	0.43	0.58	0.29	0.48	0.60	0.57

From Table 7.2 we see that similarity measures that only use page-counts such as, the Normalized Google Distance (NGD) and pointwise mutual information (PMI), have a similar level of performance compared to similarity measures that only use snippets, such as the Sahami and Heilman's (SH) approach. This is in contrast to the results reported for the Miller-Charles dataset (Table 2.4) where snippets-based approaches clearly outperformed page-counts-based approaches. Compared to the word pairs in the Miller-Charles dataset, there are numerous related word pairs and even some named-entities in the WordSimilarity-353 dataset. Compared to common nouns which have multiple senses on the web, popular named-entities have a limited and well-defined set of senses. Page-counts-based measures do not consider the local context in which two words co-occur and therefore produce incorrect similarity scores when one or both words being compared are polysemous. For example, the word pair (*brother*, *monk*) is assigned a high similarity score of 0.71 by human annotators in the Miller-Charles dataset. Here, the word *brother* is used in the WordNet sense, *a title given to a monk and used as form of address*. The high similarity score assigned by humans is thus justifiable. However, similarity measures that only use page-counts such as the Jaccard measure (0.25), and the Dice measure (0.27) report low similarity scores, whereas the co-occurrence double checking model which use snippets reports a high similarity score of 0.55.

Tables 7.3.2 and 7.3.2 summarize the previously proposed WordNet-based semantic similarity measures respectively on the Miller-Charles dataset and WordSimilarity-353

Table 7.3: Comparison with WordNet-based similarity measures on Miller-Charles dataset (Pearson correlation coefficient).

Method	Correlation
Edge-counting	0.664
Jiang & Conrath [69]	0.848
Lin [84]	0.822
Resnik [126]	0.745
Li et al. [160]	0.891
SVM	0.834
RM	0.867

Table 7.4: Comparison with WordNet-based methods on WordSimilarity-353 dataset (Spearman rank correlation coefficient).

Method	Correlation
WordNet Edges [68]	0.27
Hirst & St-Onge [61]	0.34
Jiang & Conrath [69]	0.34
WikiRelate! [142]	0.19-0.48
Leacock & Chodrow [80]	0.36
Lin [87]	0.36
Resnik [126]	0.37
SVM	0.53
RM	0.52

dataset. Despite the fact that the proposed method does not use manually compiled resources such as WordNet for computing similarity, its performance is comparable to similarity measures that use WordNet. The WordNet-based similarity measures utilizes numerous information that can be extracted from WordNet such as the length of the shortest path that connects the two words for which we are concerned of measuring similarity, the semantic relations that appear along the that path, the depth of the least common ancestor (LCA) that subsumes the two words, the number of nodes that falls under that LCA. Because WordNet is a fixed ontology (i.e. does not change over time as rapidly as the web), semantic similarity measures that use WordNet as the only source of information cannot compute the similarity between novel words such as named-entities (i.e. names of companies, names of people, etc.). Because WordNet does not encode any statistical information about the words it contains, algorithms that also require statistical information to compute semantic similarity [1, 160] have used fixed text corpora to obtain word statistical information such as word frequency counts. The combination of WordNet with external corpora has improved the accuracy of semantic similarity measurement. Moreover, the fixed hierarchy of WordNet does not encode information that is unique to a particular domain. For example, all trees are categorized directly under a generic node for trees. But this crude classification might not be suitable for measuring the similarity between names of trees in a botanical context. A higher correlation coefficient indicates a better agreement with human notion of semantic similarity. From Table 7.3.2 we can see that the proposed method outperforms a wide variety of semantic similarity measures developed using numerous resources including lexical resources such as WordNet and knowledge bases such as Wikipedia (i.e. WikiRelate!).

Despite the fact that the relational model of semantic similarity (RM) introduced in this Chapter, performs competitively with the support vector machine-based approach (SVM), which integrates both page-counts-based association measures as well as snippets-based lexical patterns through a machine learning algorithm, it is natural to extend the SVM approach to use clusters of patterns instead of individual lexical patterns. Moreover, the features used by WordNet-based approaches such as the depth of a node in the WordNet hierarchy and the length of the shortest path connecting two nodes, etc. can also be incorporated in the SVM approach. Specifically, we can define additional features for training using the information from the WordNet. Whether such a hybrid approach that combines WordNet-based features and Web-based features in a single model can further improve the performance of a semantic similarity measure remains unknown. However, it must be noted that extending web-based features using WordNet can be problematic in situations where the words that we must compute semantic similarity between, do not appear in the WordNet. As a workaround for this problem, one can first use web-based features to find a word $u\prime$ that is similar to the given word u, which does not appear in the WordNet, and then use $u\prime$ to generate WordNet-based features. However, whether such an approach can indeed improve the performance of a semantic similarity measure, remains to be experimented. In future research on semantic similarity, I intend to explore these possibilities.

Chapter 8

Conclusions and Future Work

8.1 Conclusions

In the thesis I studied two types of similarities: attributional similarity and relational similarity. In Chapter 2, I proposed a supervised approach to measure the attributional similarity between two given words using web search engines. I used page counts and lexical patterns extracted from snippets retrieved from a search engine. The proposed method uses synonymous word pairs selected from WordNet synsets as positive training instances and automatically generates negative training instances using a random shuffling technique. The proposed method outperformed numerous web-based attributional similarity measures on the Miller-Charles benchmark dataset, achieving a statistically significant Pearson correlation coefficient of 0.837.

In Chapter 3, I studied the problem of measuring the relational similarity between two given word pairs. I compared three different approaches in Chapter 3: a supervised approach using Support Vector Machines (Section 3.2), an unsupervised approach that measures Mahalanobis distance between two word pairs using a set of lexical pattern clusters (Section 3.3), and a supervised approach that learns the elements of the Mahalanobis matrix using training data. Experimental results show that clustering lexical patterns prior to computing relational similarity improves performance. This is a predicted behavior because by clustering lexical patterns we can group patterns that are semantically similar, thereby reducing the sparseness in feature vectors.

I studied the problem of entity resolution on the web as a specific application of web-based similarity measures. Resolving personal named-entities can be further subdivided into namesake disambiguation (Chapter 4), and name alias detection (Chapter 5). I proposed and evaluated a method to extract keywords that uniquely identify different people with a given ambiguous name. The keywords returned by the proposed method can accurately disambiguate namesakes in an information retrieval task. In particular, named entities and terms that are related to the ambiguous name are useful when disambiguating namesakes. I proposed term-entity model (TEM) to represent a person on the web, and proposed a fully automatic technique to create TEMs for a given name. As an extension to this work, I presented preliminary work that I have conducted in extracting attributes for people from web text. The name alias problem was studied in Chapter 5, where I proposed two different approaches to extract aliases of a name. In the first approach, I used lexical patterns to find candidate aliases of a given name. The second approach is based on anchor text co-occurrences. A hybrid approach that integrates both anchor texts and lexical patterns reported the best results.

In Chapter 7, I proposed a relational model of semantic similarity. This model is expected to find a link between attributional and relational similarity measures. Specifically, I computed the attributional similarity between two words using the set of semantic relations that hold between those two words. I proposed a pattern clustering method that requires only positive training data (i.e. synonymous word pairs extracted from WordNet). The relational model of attributional similarity was able to further improve the semantic similarity measurements discussed in Chapter 2. Next I provide potential future research directions in this field.

8.2 Open Issues

The notion of similarity as perceived by humans is not yet fully understood. Researchers in cognitive science and psycholinguistics have studied various aspects of human notion of similarity. In those studied several fundamental questions have been debated. Is similarity a notion that all humans possess by birth or is it learned over time with experience?, do humans use the same mechanism to perceive both attributional and relational similarity or different mechanism?, which part or parts in the brain is at work when humans compute similarity between objects?, are among the fundamental questions that have been raised repeatedly and debated over in previous work studying the human notion of similarity.

Chomsky [27] argues that similarity is an inherent ability. Not limiting to similarity, he argues that language is already *encoded* into human brain by birth and depending on the environment and experiences a particular individual might have, different parts get activated. He argues that an average school student might not be able to learn all constructions and meanings of words only by encountering them in a classroom or in his environment. This argument is compelling when one considers the large number of words a school student acquire. However, the exact genes (assuming that language skills are encoded in genes) that contribute to language abilities in a human is not yet discovered. The quest for language acquisition dates back to Plato. The famous Plato's problem (the term is attributed to Chomsky) asks "how do we gain the language abilities from a limited number of experiences'?'. According to Chomsky's universal grammar, any child, depending on where he or she is born on earth, can acquire the grammar of the native language only by a limit amount of passive inputs. On the other hand, empiricists argue that all meanings of words and grammar of a language can and is learned only by experiences. Landauer and Dumais [78] proposed the famous Latent Semantic Analysis (LSA) as a solution to Plato's problem. In LSA, first a matrix is representing a set of documents is created as follows. Each row in the matrix represents a document and each column represents a word. The (i, j) element of this matrix is the weight assigned to a word w_j inside a document d_i. The exact word weighting algorithm can be selected arbitrarily. A popular choice is the tfidf weighting scheme. Next, singular value decomposition is performed on this matrix to obtain the left and right orthonormal matrices and a diagonal matrix where the diagonal elements are eigenvalues. The left and right orthonormal matrices can then be used to either reduce the number of rows or columns in the original matrix. This operation, for example when performed to reduce the number of columns in the original matrix, results in clusters of words. Experimentally, it can be seen that words that are semantically similar groups to form new dimensions. Consequently, the sparseness of the original matrix reduces. LSA has been successfully applied for various task such as text summarization and word clustering. Laundauer and Dumais argue that co-occurrences of words in a language is sufficient to learn the meanings of words and no external knowledge or inherent language abilities is required. The observation that LSA clusters semantically similar words to produce new dimensions, supports this claim because when we encounter a new word we can *guess* its meaning using the other words that belong to the same cluster as the new word. Moreover, LSA does not assume any external knowledge sources such as dictionaries, and works only on the given set of documents. This can be seen as a school student guessing the meanings of new words using the contexts they appear in, without using any external aid, such as looking up in a dictionary or asking from a teacher. Although the empiricists view of language acquisition is very attractive from a data driven NLP point-of-view, computer algorithms that only use unsupervised data often require a large amount of data to achieve the performance levels that are achieved by supervised approaches with much less data. In certain applications, it is not possible to surpass the performance of supervised approaches only by using large unsupervised datasets. A student learning the meaning of new words without the aid of a dictionary or a teacher can be approximated as a process of unsupervised learning. Moreover, it is unlikely that we process terabytes of text data in our brains to learn the meaning of new words. It would require much large computation power and/or computation time for a human to even read such a large collection of textual data, let alone process it. Therefore, I believe that a certain amount of language skills are already encoded in the brain at birth. However, which components get triggered is determined by the experiences of a particular individual. However, the debate on language acquisition far from being settled.

Although certain animals such as gorillas and dolphins have been reported to possess a certain level of limited language abilities, the language skills of humans is far more advanced than any known animal. However, it is not yet clear whether it is the same for similarity measuring abilities. Animals can compare current experiences with previous encounters and act accordingly. For example, dogs are often trained to accomplish a certain task (e.g. searching for drugs in an air port) by giving a favorable feedback (i.e. giving some food) when the task is successfully accomplished. The relation between completing task and obtaining food is learned by the dog over time. Once this relationship is learned, the dog can be motivated to accomplish different tasks and with different feedbacks (e.g. touching the head instead of giving food). If we represent two tasks as (detect drugs, obtain food) and (wait for the master, get the head touched) then the two acts can be considered as a case of measuring relational similarity. The ability to measure similarity can help to generalize ones experiences and infer from prior experiences when a novel situation arises. Analogical reasoning is considered as an important process of human knowledge processing.

The distinction between attributional and relational similarity measuring mechanisms is an interesting one. We have already seen in Section 1.2 that the two types of similarities are indeed closely related. The relationship between attributional and relational similarity depends upon how we define attributes or relations. Gentre proposes *relational shift*, a hypothesis that states at first humans acquire the ability to measure attributional similarity and when he or she gain more experience about the environment that he or she lives in, relational similarity measurement abilities develop. Gentre carried out an interesting experiment where a group of children were asked to compare a sponge and clouds in the sky. When the question was posed to small children apparently they described that both sponges and clouds are soft, light in mass and white colored. Interestingly, when the same question was posed to older children (aging from 10 to 12 years), they explained that both sponges and clouds can store water and can later reproduce it. The analysis by small children is based on attributions of sponges and clouds, whereas the older children focused on relational properties. From a computational point of view measuring relational similarity does indeed appears to a more difficult task compared to measuring attributional similarity. One must first identify the relations between objects and subsequently measure the relations that exist between different pairs of objects to measure the relational similarity. Moreover, the task of recognizing relations must be carried out for each pair of objects. Contrastingly, when we define an object we often define its attributes (or properties). Therefore, when measuring attributional similarity we are already given the set of attributes that must be compared between the two objects. Moreover, the attributes of an object is determined irrespective of what other objects exists. Therefore, we do not need to compute attributes for each new combination (pairing) of two objects that we must compute attribute similarity for. This property of attributional similarity is desirable in kernalization of an learning algorithm that is designed for measuring attributional similarity. However, the "attributes first" view of objects itself is being challenged and attributes themselves are seen as a mere consequence of relations between objects in the relational models of similarity such as the structure mapping theory.

The question as to whether being able to measure similarity is an indication of intelligence is an interesting one. For example, SAT exam, which is used to select candidates for U.S. universities, contained word-analogy question until few year back. Word-analogy questions selected from SAT exams have been used to benchmark relational similarity algorithms. To solve a word-analogy question a candidate must not only understand the meanings of individual words, but must also be able to compare the relations that exist between the two words in each word pair. This requires higher processing skills not limiting to merely memorizing the meanings of words. Similar questions have been used in IQ tests. For example, guessing the next number or object in a sequence requires one to recognize some patterns (or relations) among preceding items in the sequence. Another popular type of questions used in IQ tests are detecting the object that does not relate to a set of objects. Measuring both attributional and relational similarity is a fundamental process when answering all these questions. An interesting association between "testing for intelligence" and "measuring similarity" is the Turing's test for intelligence. Turing argued that if a human cannot distinguish between the output (i.e. written text) produced by another human and a computer program, then the program can be considered as "intelligent". If this is true, then Turing's test is in fact a test for measuring the similarity between human output and the output produced by a computer program.

8.3 Future Work

In Section 1.5, I presented results from psychological experiments that indicate similarity to be symmetric. However, we ignored this aspect of similarity when I designed the proposed semantic similarity measure. In particular, I interchanged the variables X and Y in all extracted lexical patterns. This step ensures that the trained SVM model contains features corresponding to both a pattern $p_{x,y}$ and the X, Y interchanged pattern (reverse pattern), $p_{y,x}$. However, it must be noted that the weights assigned to those two patterns might not necessarily be equal. This is because not all synonyms are represented equally well in the web by a lexical pattern and its reverse pattern. For example, consider the synonyms *automobile* and *car*. This pair has the highest similarity score in Miller-Charles' benchmark dataset, indicating that humans agree that these two words are perfect synonyms. However, the page-count returned by Google for the pattern *"automobile is a car"* is $11,200$, whereas the same for the pattern *car is an automobile"* is $5,330$. We can see that the page-counts for the former pattern is more than double for the latter pattern. Therefore, even if we compare two perfect synonyms (according to human intuition of similarity), the proposed similarity measure is not symmetric. In addition to the asymmetry in lexical patterns, the pattern selection process also renders the propose similarity measure asymmetric. Although we (artificially) form symmetric patterns by interchanging X and Y variables in all extracted lexical patterns, the χ^2-based pattern selection process might not select the both versions of a lexical pattern. Consequently, the feature set itself contains asymmetry even before we conduct any training. One could convert an asymmetric similarity measure $sim(a,b)$ by considering the average similarity for the two orderings, $0.5(sim(a,b) + sim(b,a))$. However, whether such an artificial conversion into symmetry will reflect the human notion of similarity is an open question. Moreover, the consequences of asymmetry in subsequent applications such as, community clustering or word sense disambiguation as conducted in this thesis remains unknown. Further analysis is required to investigate the effect of asymmetry in web-based semantic similarity measures.

The distinction between attributional similarity, semantic similarity, relatedness, and synonymy is a subtle one. Synonymy is a strong form of attributional similarity. According to Plato, synonyms are words that can be replaced in any context without considering other words that appear with it. However, most linguists do not agree with this definition of synonymy because no word can be perfectly replaced in all contexts by its synonyms. Acronyms might be an exception to this rule because an acronym and its full form is read in the same way and can be replaced in all contexts. However, substitutability as a property of synonymy does not hold for other types of words. If two words are synonymous then it is safe to assume that they have a high degree of attributional similarity. Extracting related words have received much attention lately because of their use in information retrieval systems such as web search engines. Many commercial search engines provide related words as suggestions for user queries. Related words are often identified using previous queries to the system by other users. Alternatively, a search engine can expand a user query (in the form of a disjunctive combination of the original query and its related words), to improve recall in a search engine. However, considering the vast amount of information indexed by web search engines, the focus is more on precision than recall. Moreover, the quality of the automatically extracted related words makes it insufficient to automatically expand user queries in a simple manner such as a disjunctive query. Therefore, search engines employ the former approach, where related words are first suggested to the user. Related words show a certain level of attributional similarity. In fact, benchmark datasets of attributional similarity such as the WordSimilarity-353 dataset, contains related word pairs such as *Maradona* and *Football*. However, we would expect two related words to have a lesser value of attributional similarity than two synonyms. Semantic similarity is used as a term to refer synonymy as well as semantic relatedness in previous work. However, it does not include relational similarity. Research on relational similarity have explicitly used the term attributional similarity to distinguish the two types of similarities. However, the term semantic similarity is too vague and covers various types of similarities. The issue of relatedness as a factor of attributional similarity requires further studied. Recently, Agirre et al. [1] investigated the problem of measuring relatedness and similarity between words using a partition of WordSimilarity-353 dataset. They asked two annotators to classify the word pairs in the WordSimilarity-353 dataset into several categories: synonyms, antonyms, identical, hyperonym-hyponym, holonym-meronym, meronym-holonym, and none of the above. They observed a high inter-annotator agreement of 0.80, and a Kappa score of 0.77. They grouped the word pairs in the WordSimilarity-353 dataset into three categories based on the manual classifications by the annotators. The three groups are similar pairs (those classified as synonyms, antonyms, identical, or hyponym-hyperonym), related pairs (those classified as meronym-holonym, and pairs classified as none-of-the-above, with a human average similarity greater than 5), and unrelated pairs (those classified as none-of-the-above that had average similarity less than or equal to 5). They proposed two gold standards by taking the union of similar and unrelated pairs (similarity dataset), and the union of related and unrelated pairs (relatedness dataset). Interestingly, their experimental results show that considering a window of words (i.e. lexical patterns, order between words is preserved) from the context of a word is highly effective to measure similarity between words, whereas considering the context of a word as a bag-of-words (i.e. a vector of words representing the context of a word, order between words is lost) is more suitable to measure relatedness.

Classification of related words according to the *role* of a word is an interesting research direction of web-based related word extraction. Given a set $S(t)$ of related words extracted by some algorithm A for a target word t, the problem of related word classification can be defined as partitioning $S(t)$ into a group of overlapping clusters such that each cluster represents a set of words that are related to t because of a different role. For example, consider the set of words, $\{$*hound*, *cat*, *frankfurter*, *hot dog*, *k9*$\}$. Here, one can classify the words in this set into three groups as follows: (hound, k9) [synonyms of dog], (cat) [dogs and cats are pets], and (frankfurter, hot dog) [food items that are related to a different sense of the word *dog*]. This classification is by no means unique. For example, one might classify both cat and k9 as pets. Therefore, the definition of the problem permits overlaps between clusters (i.e. a soft clustering of related words, where some words can appear in more than one cluster). Classifying words according to WordNet relations such as synonymy, hypernymy, meronymy, etc. is not sufficient to explain the different clusters of words in our example. For example, the relation *both dogs and X can be kept as pets* is a relation that cannot be categorized into any of those fundamental relation types in the WordNet. Moreover, the number of such relations that are useful as partitioning schemes can be large and might not be able to enumerate. Supervised multi-class classification is not suitable for the current problem. In particular, if we are only interested in classifying synonyms or hyponyms then we can train a binary classifier using a set of synonymous (or hyponymous) word pairs, and then use the trained classifier to identify the words in $S(t)$ that belongs to the same category (i.e. synonyms or hyponyms). But this is not possible in the current situation because we do not know in advance what relations there can be in a given set of related words, and even if we knew all the possible relations that can exist in advance, it would not be possible to prepare training instances for all relation types. Specially, creating negative training instances for relational learning can be problematic. Alternatively, one can take an unsupervised approach where we first identify the different relations that are associated with the target word t and each word in the related words set $S(t)$, and subsequently cluster the related words that can be identified with a particular relation. If the related word classification problem can be solved with high accuracy, then it can be used to filter the noise (incorrectly extracted related words) in related words extraction algorithms. Considering the fact that the ability to classify objects is a fundamental skill that humans have, devising a computer algorithm to do the same (at least in the limited case discussed above) would surely illustrate the difficulty/ease of the computations required to achieve such a task.

Transitivity of similar relations does not hold in general. For example, if two words A and B are similar and B is also similar to another word C, we cannot expect A and C to be similar in general. For example, consider the two words *horse* and *car*. Considering the fact that cars and historically horses are used for transportation, we would expect cars and horses to be attributionally similar. Here, the value of the attribute *usage* is common (i.e. transportation) for both objects. Similarly, considering the fact that both horses and pigs are farm animals, one would expect a high similarity between horses and pigs. However, it does not follow from this analysis that cars and pigs are similar. In fact, they have very different attributes and are not considered as similar. On the other hand if we consider *cows*, which are also farm animals, we can observe a certain degree of similarity between cars and cows because cows are still used for transportation (e.g. bullock carts). To my knowledge, there is no computer algorithm that can determine which cases are transitive and which are not. Further studies into transitivity of attributional similarity will extend our understanding of similarity. Existing algorithms for measuring attributional or relational similarity only return a value indicating the degree of similarity between a pair of words or two pairs of words. They do not indicate why two words are similar or not similar. Extracting informative lexical patterns that explain why two words are assigned a particular similarity score will not only help to easily detect errors in automatic similarity measures, but will also strengthen our understanding on what factors contribute to similarity between words or semantic relations.

Identification of relations is an important first step in measuring both attributional and relational similarity. In this thesis, I used lexical patterns as an approximate representation of semantic relations. However, not all relations can be explicitly stated in the form of lexical patterns. For example, consider co-references that span across multiple sentences. Because lexical patterns are generated from entities that co-occur in the same sentence, we cannot express a relation that span across multiple sentences and have co-references in other sentences in a document by lexical patterns. An alternative approach would be to first perform within and cross-document co-reference resolution and then use the co-reference information to express relations that span multiple sentences or several documents. However, this approach can be problematic in the web because of the noise in web documents, which render accurate co-reference analysis difficult, and also the sheer scale of the web that makes such in depth analysis of all documents computationally prohibitively costly. As a practical (partial) solution to the cross-document co-references in the case of name-aliases, I introduced the use of anchor texts and link graph on the web in Chapter 5. Moreover, some relations occur as a consequence of an event. For example, consider the sentence, "*X published his research paper in Y*". Here, *X* can be the author of a research paper and *Y* can be a conference or a journal. Publishing is an activity performed by researchers and this prior knowledge enables a human to detect a authorship relation between *X* and *Y*. However, this authorship relation is only implicitly stated in the example sentence. The verb *publish* is a clue that indicate this relation. However, besides the simplicity, there is no other reason to limit any of the algorithms discussed in the thesis for lexical patterns. For example, we can replace (or complement) the lexical patterns-based representation of semantic relations with dependency or syntactic paths, network representations such as markov logic networks or predicate logic. However, most commercially available web search engines do not provide the functionality to search using richer representations beyond lexical patterns. Not even regular expressions are supported in most web search engines. Although there has been some pioneering work on building search engines for NLP applications [24], they are yet to achieve the scalability provided by commercial web search engines.

In Chapter 6, I introduced a two-step approach to extract various attributes for people from the web. However, the low performance reported by all participating systems in the WePS workshop suggests the difficulty of the task. The fundamental question *what can be an attribute of an object?*, itself requires some discussion. In the case of people, we assume attributes such as date of birth, affiliation, nationality, e-mail address, etc. as attributes. In general, we can think of an attribute as a feature of a class of objects, that distinguish a particular class from another. For example, in the case of people, the attribute *date of birth* is common for all people and distinguishes the class *people* from another class (e.g. *books*). If we take a bottom-up approach to attribute extraction, we can start with a collection of objects that we are told to be in the same class, and then identify features that are common to most (if not all) objects in the collection. For example, consider the situation where we are given a list of name cards. We know in advance all name-cards belong to the class *people*. Our goal is to extract attributes for this class. Next, we start by explicitly writing down what features exist in each name-card. For example, we might have features such as the *first name*, *last name*, *telephone number*, *fax number*, *job title*, *academic degree* or other professional qualifications, and *e-mail address*. If those features also appear in the other name-cards, then we can select them as attributes for the class. This bottom-up view of attribute extraction is particularly useful when we are not sure as to which features are salient in a given collection of objects. In contrast to attribute *extraction*, which I studied in Chapter 5, the above mentioned process can be considered as an attribute *mining* task. In our example of name-cards, we were only given a set of name-cards for people (i.e. all objects belongs to a single class). We can further extend this idea to cover a collection of objects that belongs to more than one class and perform the attribute mining task in a discriminative fashion. However, it must be noted that the above mentioned analysis only provides us with a set of attributes for a class. It does not extract the values of those attributes. Once we have identified which attributes to be extracted for a particular class, we need other techniques (a list-wise approach and a rule-based approach were introduced in Chapter 6) to extract the values of those attributes.

Measuring similarity between words using the information available on the web will remain a challenging and interesting area of research for the years to come. The information explosion in the world wide web and the advancement in web search engines have provided us with both vast amount of textual data as well as efficient processing tools. It is no longer possible to run the entire web corpus through our algorithms. However, the vast size and redundancy of information in the web obviate the need for us to process the entire web corpus to obtain statistically reliable counts or samples. The algorithms presented in this thesis use light-weight processing steps such as downloading only the top ranking snippets or using the page counts for a small number of queries. Therefore, the algorithms presented for measuring attributional and relational similarity easily scale to the web. Moreover, the ever increasing size of the web does not impede the performance of the algorithms. On the contrary, when the web grows it helps us to obtain even more reliable page counts.

Bibliography

[1] Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Pasca, and Aitor Soroa. A study on similarity and relatedness using distributional and wordnet-based approaches. In *Proc. of NAACL-HLT'09*, 2009.

[2] J. Artiles, J. Gonzalo, and F. Verdejo. A testbed for people searching strategies in the www. In *Proc. of SIGIR'05*, pages 569–570, 2005.

[3] Javier Artiles, Julio Gonzalno, and Satoshi Seline. Weps 2 evaluation campaign: overview of the web people search clustering task. In *proc. of the 2nd Web People Search Evaluation Workshop (WePS 2009) at 18th International World Wide Web Conference*, 2009.

[4] Javier Artiles, Julio Gonzalo, and Satoshi Sekine. The semeval-2007 weps evaluation: Establishing a benchmark for the web people search task. In *proc. of the SemEval at Annual Meeting of the Association for Computational Linguistics*, 2007.

[5] R.A. Baeza-Yates and B.A. Ribeiro-Neto. *Modern Information Retrieval.* ACM Press / Addison-Wesley, 1999.

[6] A. Bagga and B. Baldwin. Entity-based cross document coreferencing using the vector space model. In *Proc. of 36th COLING-ACL*, pages 79–85, 1998.

[7] S. Banerjee and T. Pedersen. Extended gloss overlaps as a measure of semantic relatedeness. In *Proc. of IJCAI'03*, pages 805–810, 2003.

[8] Z. Bar-Yossef and M. Gurevich. Random sampling from a search engine's index. In *Proceedings of 15th International World Wide Web Conference*, 2006.

[9] K. Barker and S. Szpakowicz. Semi-automatic recognition of noun modifier relationships. In *Proc. of COLING'98*, pages 96–102, 1998.

[10] Ron Bekkerman and Andrew McCallum. Disambiguating web appearances of people in a social network. In *Proceedings of the World Wide Web Conference (WWW)*, pages 463–470, 2005.

[11] Kedar Bellare, Partha Pratim Talukdar, Giridhar Kumaran, Fernando Pereira, Mark Liberman, Andrew McCallum, and Mark Dredze. Lightly-supervised attribute extraction for web search. In *proc. of NIPS 2007 Workshop on Machine Learning for Web Search*, 2007.

[12] M. Berland and E. Charniak. Finding parts in very large corpora. In *Proc. of ACL'99*, pages 57–64, 1999.

[13] R. Bhagat and D. Ravichandran. Large scale acquisition of paraphrases for learning surface patterns. In *Proc. of ACL'08: HLT*, pages 674–682, 2008.

[14] E. Bicici and D. Yuret. Clustering word pairs to answer analogy questions. In *Proc. of TAINN'06*, 2006.

[15] M. Bilenko and R. Mooney. Adaptive duplicate detection using learnable string similarity measures. In *Proc. of SIGKDD'03*, 2003.

[16] D. Bollegala, Y. Matsuo, and M. Ishizuka. Disambiguating personal names on the web using automatically extracted key phrases. In *Proc. of the 17th European Conference on Artificial Intelligence*, pages 553–557, 2006.

[17] D. Bollegala, Y. Matsuo, and M. Ishizuka. An integrated approach to measuring semantic similarity between words using information available on the web. In *Proc. of HTL-NAACL'07*, pages 340–347, 2007.

[18] D. Bollegala, Y. Matsuo, and M. Ishizuka. Measuring semantic similarity between words using web search engines. In *Proc. of WWW'07*, pages 757–766, 2007.

[19] D. Bollegala, Y. Matsuo, and M. Ishizuka. Www sits the sat: Measuring relational similarity on the web. In *Proc. of ECAI'08*, pages 333–337, 2008.

[20] Danushka Bollegala, Taiki Honma, Yutaka Matsuo, and Mitsuru Ishizuka. Automatically extracting personal name aliases from the web. In *proc. of the 6th International Conference on Natural Language Processing (GoTAL 08), Advances in Natural Language Processing Springer LNCS 5221*, pages 77–88, 2008.

[21] C. Buckley, G. Salton, J. Allan, and A. Singhal. Automatic query expansion using smart: Trec 3. In *Proc. of 3rd Text REtreival Conference*, pages 69–80, 1994.

[22] A. Budanitsky and G. Hirst. Evaluating wordnet-based measures of semantic distance. *Computational Linguistics*, 32(1):13–47, 2006.

[23] R. C. Bunescu and R.J. Mooney. Learning to extract relations from the web using minimal supervision. In *Proc. of ACL'07*, pages 576–583, 2007.

[24] Michael J. Cafarella and Oren Etzioni. A search engine for natural language applications. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 442–452, New York, NY, USA, 2005. ACM Press.

[25] S. Chakrabarti. *Mining the Web: Discovering Knowledge from Hypertext Data*. Morgan Kaufmann, 2003.

[26] H. Chen, M. Lin, and Y. Wei. Novel association measures using web search with double checking. In *Proc. of the COLING/ACL 2006*, pages 1009–1016, 2006.

[27] Noam Chmosky. *Modular Approaches to the Study of the Mind.* San Diego State University Press, 1984.

[28] K. Church and P. Hanks. Word association norms, mutual information and lexicography. *Computational Linguistics*, 16:22–29, 1991.

[29] R.L. Cilibrasi and P.M.B. Vitanyi. The google similarity distance. *IEEE Transactions on Knowledge and Data Engineering*, 19(3):370–383, 2007.

[30] P. Cimano, S. Handschuh, and S. Staab. Towards the self-annotating web. In *Proc. of 13th WWW*, 2004.

[31] P. Cimano, S. Handschuh, and S. Staab. Towards the self-annotating web. In *Proc. of WWW'04*, 2004.

[32] P. Cimiano and J. Wenderoth. Automatic acquisition of ranked qualia structures from the web. In *Proc. of ACL'07*, pages 888–895, 2007.

[33] A. Culotta and J. Sorensen. Dependency tree kernels for relation extraction. In *Proc. of ACL'04*, pages 423–429, 2004.

[34] J. Curran. Ensemble methods for automatic thesaurus extraction. In *Proc. of EMNLP*, 2002.

[35] Douglass R. Cutting, Jan O. Pedersen, David Karger, and John W. Tukey. Scatter/gather: A cluster-based approach to browsing large document collections. In *Proceedings SIGIR '92*, pages 318–329, 1992.

[36] D. Davidov and A. Rappoport. Classification of semantic relationships between nominals using pattern clusters. In *Proc. of the ACL'08*, 2008.

[37] D. Davidov and A. Rappoport. Unsupervised discovery of generic relationships using pattern clusters and its evaluation by automatically generated sat analogy questions. In *Proc. of ACL'08-HLT*, pages 692–700, 2008.

[38] J. V. Davis and I. S. Dhillon. Differential entropic clustering of multivariate gaussians. In *Proc. of NIPS'06*, pages 337–344, 2006.

[39] J. V. Davis and I. S. Dhillon. Structured metric learning for high dimensional problems. In *Proc. of KDD '08*, pages 195–203, 2008.

[40] J. V. Davis, B. Kulis, P. Jain, S. Sra, and I. S. Dhillon. Information-theoretic metric learning. In *Proc. of ICML'07*, pages 209–216, 2007.

[41] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. In *Proc. of Sixth Symposium on Operating System Design and Implementation (OSDI'04)*, 2004.

[42] T. Dunning. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, 19:61–74, 1993.

[43] O. Etzioni, M. Cafarella, D. Downey, A. Popescu, T. Shaked, S. Soderl, D. S. Weld, and E. Yates. Unsupervised named-entity extraction from the web: An experimental study. *Artificial Intelligence*, 165:91–134, 2005.

[44] J. Euzenat. Semantic precision and recall for ontology alignment evaluation. In *Proc. of International Joint Conferences on Artificial Intelligence (IJCAI'07)*, pages 348–353, 2007.

[45] B. Falkenhainer, K.D. Forbus, and D. Gentner. Structure mapping engine: Algorithm and examples. *Artificial Intelligence*, 41:1–63, 1989.

[46] L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman, and E. Ruppin. Placing search in context: The concept revisited. *ACM TOIS*, 20:116–131, 2002.

[47] M.B. Fleischman and E. Hovy. Multi-document person name resolution. In *Proceedings of 42nd Annual Meeting of the Association for Computational Linguistics (ACL), Reference Resolution Workshop*, 2004.

[48] K.T. Frantzi and S. Ananiadou. The c-value/nc-value domain independent method for multi-word term extraction. *Journal of Natural Language Processing*, 6(3):145–179, 1999.

[49] C. Galvez and F. Moya-Anegon. Approximate personal name-matching through finite-state graphs. *Journal of the American Society for Information Science and Technology*, 58:1–17, 2007.

[50] S. Gauch and J. B. Smith. Search improvement via automatic query reformulation. *ACM Trans. on Information Systems*, 9(3):249–280, 1991.

[51] Lise Getoor, Nir Friedman, Daphne Koller, Avi Pfeffer, and Benjamin Taskar. Probabilistic relational models. In L. Getoor and B. Taskar, editors, *An Introduction to Statistical Relational Learning*. MIT Press, 2007.

[52] S. Ghemawat, H. Gobioff, and S. Leung. The google file system. In *Proc. of ACM Symposium on Operating Systems Principles*, 2003.

[53] P.E. Gill, W. Murray, and M.H. Wright. *Practical optimization*. Academic Press, 1981.

[54] R. Guha and A. Garg. Disambiguating people in search. In *Stanford University*, 2004.

[55] R. V. Guha, R. McCool, and E. Miller. Semantic search. In *Proc. of WWW'03*, pages 700–709, 2003.

[56] W.V. Hage, H. Kolib, and G. Schreiber. A method for learning part-whole relations. In *Proc. of 5th International Semantic Web Conferences*, 2006.

[57] U. Hahn, N. Chater, and L. B. Richardson. Similarity as transformation. *Cognition*, 87:1–32, 2003.

[58] Hui Han, Hongyuan Zha, and C. Lee Giles. Name disambiguation in author citations using a k-way spectral clustering method. In *Proceedings of the International Conference on Digital Libraries*, 2005.

[59] Z. Harris. Distributional structure. *Word*, 10:146–162, 1954.

[60] M.A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proc. of 14th COLING*, pages 539–545, 1992.

[61] G. Hirst and D. St-Onge. Lexical chains as representations of context for the detection and correction of malapropisms, 1997.

[62] T. Hisamitsu and Y. Niwa. Topic-word selection based on combinatorial probability. In *Proc. of NLPRS'01*, pages 289–296, 2001.

[63] T. Hokama and H. Kitagawa. Extracting mnemonic names of people from the web. In *Proc. of 9th Intl. Conf. on Asian Digital Libraries (ICADL'06)*, pages 121–130, 2006.

[64] H. L. Hollingworth. Judgements of similarity and difference. *Psychological Review*, 20:271, 1913.

[65] J. Hosman and T. Kuennapas. On the relation between similarity and dissimilarity estimates. Technical report, University of Stockholm, 1973.

[66] J. Huang, S. Ertekin, and C.L. Giles. Efficient name disambiguation for large scale databases. In *Proc. of 10th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD)*, pages 536–544, 2006.

[67] J. Huang, S. Ertekin, and C.L. Giles. Fast author name disambiguation in citeseer. *ISI Technical Report*, 2006.

[68] M. Jarmasz. Roget's thesaurus as a lexical resource for natural language processing. Master's thesis, University of Ottawa, 1993.

[69] J.J. Jiang and D.W. Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proc. of ROCLING'98*, 1998.

[70] T. Joachims. Optimizing search engines using clickthrough data. In *Proc. of KDD'02*, 2002.

[71] Y. Kalfoglou and M. Schorlemmer. Ontology mapping: the state of the art. *The Knowledge Engineering Review*, 18:1–31, 2003.

[72] P. Kanani, A. McCallum, and C. Pal. Improving author coreference by resource-bound information gathering from the web. In *Proc. of International Joint Conferences on Artificial Intelligence (IJCAI'07)*, pages 429–434, 2007.

[73] Ravi Kannan, Santosh Vempala, and Adrian Vetta. On clusterings: Good, bad, and spectral. In *Proceedings of the 41st Annual Symposium on the Foundation of Computer Science*, pages 367–380, 2000.

[74] F. Keller and M. Lapata. Using the web to obtain frequencies for unseen bigrams. *Computational Linguistics*, 29(3):459–484, 2003.

[75] Adam Kilgarriff. Googleology is bad science. *Computational Linguistics*, 33:147–151, 2007.

[76] C. L. Krumhansl. Concerning the applicability of geometric models to similarity data: The interrelationship between similarity and spatial density. *Psychological Review*, 85:445–463, 1978.

[77] T. Kudo, K. Yamamoto, and Y. Matsumoto. Applying conditional random fields to japanese morphological analysis. In *Proc. of EMNLP'04*, 2004.

[78] Thomas K. Landauer and Susan T. Dumais. A solution to plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological Review*, 104(2):211–240, April 1997.

[79] M. Lapata and F. Keller. Web-based models ofr natural language processing. *ACM Transactions on Speech and Language Processing*, 2(1):1–31, 2005.

[80] C. Leacock and M. Chodorow. *Combining Local Context and WordNet Similarity for Word Sense Identification*. MIT, 1998.

[81] M. Li, X. Chen, X. Li, B. Ma, and P.M.B. Vitanyi. The similarity metric. *IEEE Transactions on Information Theory*, 50(12):3250–3264, 2004.

[82] M. Li and P. Vitanyi. *An Introduction to Kolmogorov Complexity and its Applications*. Springer-Verlag, 1997.

[83] Xin Li, Paul Morie, and Dan Roth. Semantic integration in text, from ambiguous names to identifiable entities. *AI Magazine, American Association for Artificial Intelligence*, Spring:45–58, 2005.

[84] D. Lin. Automatic retrieval and clustering of similar words. In *Proc. of the 17th COLING*, pages 768–774, 1998.

[85] D. Lin. Automatic retrieval and clustering of similar words. In *Proc. of COLING-ACL'98*, pages 768–774, 1998.

[86] D. Lin. Automatic retrieval and clustering of similar words. In *Proc. of COLING'98*, pages 768–774, Morristown, NJ, USA, 1998. Association for Computational Linguistics.

[87] D. Lin. An information-theoretic definition of similarity. In *Proc. of the 15th ICML*, pages 296–304, 1998.

[88] D. Lin and P. Pantel. Dirt: Discovery of inference rules from text. In *Proc. of ACM SIGKDD'01*, pages 323–328, 2001.

[89] F. Lorrain and H. C. White. Structural equivalence of individuals in social networks. *Journal of Mathematical Sociology*, 1:49–80, 1971.

[90] P. Mangalath, J. Quesada, and W. Kintsch. Analogy-making as prediction using relational information and lsa vectors. In *Proc. of Int'l Conf. on Research in Computational Linguistics*, 2004.

[91] Gideon S. Mann and David Yarowsky. Unsupervised personal name disambiguation. In *Proceedings of CoNLL-2003*, pages 33–40, 2003.

[92] C. Manning and H. Schutze. *Foundations of Statistical Natural Language Processing*. MIT Press, 1999.

[93] C. D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, Massachusetts, 2002.

[94] Z. Marx, D. Ido, B. Joachim, and S. Eli. Coupled clustering: A method for detecting structural correspondence. *Journal of Machine Learning Research*, 3:747–780, 2002.

[95] Y. Matsuo, M. Hamasaki, Y. Nakamura, T. Nishimura, K. Hashidaand H. Takeda, J. Mori, D. Bollegala, and M. Ishizuka. Spinning multiple social networks for semantic web. In *Proc. of the American Association for Artificial Intelligence*, 2006.

[96] Y. Matsuo, J. Mori, M. Hamasaki, K. Ishida, T. Nishimura, H. Takeda, K. Hasida, and M. Ishizuka. Polyphonet: An advanced social network extraction system. In *Proc. of 15th International World Wide Web Conference*, 2006.

[97] Y. Matsuo, T. Sakaki, K. Uchiyama, and M. Ishizuka. Graph-based word clustering using web search engine. In *Proc. of EMNLP 2006*, 2006.

[98] D. McCarthy, R. Koeling, J. Weeds, and J. Carroll. Finding predominant word senses in untagged text. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04)*, pages 279–286, 2004.

[99] D.L. Medin, R.L. Goldstone, and D. Genter. Similarity involving attributes and relations: Judgments of similarity and difference are not inverse. *Psychological Sciences*, 1(1):64–69, 1990.

[100] D.L. Medin, R.L. Goldstone, and D. Gentner. Respects for similarity. *Psychological Review*, 6(1):1–28, 1991.

[101] P. Mika. Flink: Semantic web technology for the extraction and analysis of social networks. *Web Semantics: Science, Services and Agents on the World Wide Web*, 3:211–223, 2005.

[102] P. Mika. Ontologies are us: A unified model of social networks and semantics. In *Proc. of ISWC2005*, 2005.

[103] G. Miller and W. Charles. Contextual correlates of semantic similarity. *Language and Cognitive Processes*, 6(1):1–28, 1998.

[104] G.A. Miller. Wordnet: a lexical database for english. *Commun. ACM*, 38(11):39–41, 1995.

[105] M. Mitra, A. Singhal, and C. Buckley. Improving automatic query expansion. In *Proc. of 21st Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pages 206–214, 1998.

[106] J. Mori, Y. Matsuo, and M. Ishizuka. Extracting keyphrases to represent relations in social networks from web. In *Proc. of 20th IJCAI*, 2007.

[107] P. Nakov and M. Hearst. Solving relational similarity problems using the web as a corpus. In *Proc. of ACL'08-HLT*, pages 452–460, 2008.

[108] V. Natase and S. Szpakowicz. Exploring noun-modifier semantic relations. In *Proc. of fifth int'l workshop on computational semantics (IWCS-5)*, pages 285–301, 2003.

[109] J. Nocedal and S. Wright. *Numerical optimization*. Springer, 2000.

[110] N.F. Noy and M.A. Musen. An algorithm for merging and aligning ontologies: Automation and tool support. In *Proc. of the AAAI workshop on ontology management*, 1999.

[111] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web, 1999.

[112] M. Pasca, D. Lin, J. Bigham, A. Lifchits, and A. Jain. Organizing and searching the world wide web of facts - step one: the one-million fact extraction challenge. In *Proc. of AAAI-2006*, 2006.

[113] M. Pasca, D. Lin, J. Bigham, A. Lifchits, and A. Jain. Organizing and searching the world wide web of facts - step one: the one-million fact extraction challenge. In *Proc. of AAAI'06*, pages 1400–1405, 2006.

[114] M. Pasca, D. Lin, J. Bigham, A. Lifchits, and A. Jain. Organizing and searching the world wide web of facts - step one: the one-million fact extraction challenge. In *proc. of the 21st National Conference on Artificial Intelligence (AAAI-06)*, pages 1400 – 1405, 2006.

[115] T. Pedersen and A. Kulkarni. Discovering identities in web contexts with unsupervised clustering. In *Proc. of the IJCAI'07 Workshop for Noisy Unstructured Text Data*, 2007.

[116] T. Pedersen and A. Kulkarni. Unsupervised discrimination of person names in web contexts. In *Proc. of Eighth International Conference on Intelligent Text Processing and Computational Linguistics*, pages 18–24, 2007.

[117] T. Pedersen, A. Kulkarni, R. Angheluta, Z. Kozareva, and T. Solorio. An unsupervised language independent method of name discrimination using second order co-occurance features. In *Proc. of the Seventh International Conference on Intelligent Text Processing andComputational Linguistics (CICLing)*, 2006.

[118] Ted Pedersen, Amruta Purandare, and Anagha Kulkarni. Name discrimination by clustering similar contexts. In *Proceedings of the Sixth International Conference on Intelligent Text Processing and Computational Linguistics*, 2005.

[119] J. Pei, J. Han, B. Mortazavi-Asi, J. Wang, H. Pinto, Q. Chen, U. Dayal, and M. Hsu. Mining sequential patterns by pattern-growth: the prefixspan approach. *IEEE Transactions on Knowledge and Data Engineering*, 16(11):1424–1440, 2004.

[120] Xuan-Hieu Phan, Le-Minh Nguyen, and Susumu Horiguchi. Personal name resolution crossover documents by a semantics-based approach. *IEICE Transactions on Information and Systems*, E89-D:825–836, 2005.

[121] J. Platt. Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. *Advances in Large Margin Classifiers*, pages 61–74, 2000.

[122] R. Rada, H. Mili, E. Bichnell, and M. Blettner. Development and application of a metric on semantic nets. *IEEE Transactions on Systems, Man and Cybernetics*, 9(1):17–30, 1989.

[123] L. Ratinov and D. Roth. Design challenges and misconceptions in named entity recognition. In *Proc. of the Annual Conference on Computational Natural Language Learning (CoNLL'09)*, Jun 2009.

[124] D. Ravichandran and E. Hovy. Learning surface text patterns for a question answering system. In *Proc. of ACL '02*, pages 41–47, 2001.

[125] Y. Ravin and Z. Kaiz. Is hilary rodham clinton the president? disambiguating names across documents. In *Proc. of the ACL '99 Workshop on Coreference and its Applications*, 1999.

[126] P. Resnik. Using information content to evaluate semantic similarity in a taxonomy. In *Proc. of IJCAI'95*, 1995.

[127] P. Resnik. Semantic similarity in a taxonomy: An information based measure and its application to problems of ambiguity in natural language. *Journal of Artificial Intelligence Research*, 11:95–130, 1999.

[128] P. Resnik and N. A. Smith. The web as a parallel corpus. *Computational Linguistics*, 29(3):349–380, 2003.

[129] R. Rosenfield. A maximum entropy approach to adaptive statistical modeling. *Computer Speech and Language*, 10:187–228, 1996.

[130] H. Rubenstein and J.B. Goodenough. Contextual correlates of synonymy. *Communications of the ACM*, 8:627–633, 1965.

[131] M. Sahami and T. Heilman. A web-based kernel function for measuring the similarity of short text snippets. In *Proc. of WWW'06*, 2006.

[132] G. Salton and C. Buckley. *Introduction to Modern Information Retreival*. McGraw-Hill Book Company, 1983.

[133] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24:513–523, 1988.

[134] V. Schickel-Zuber and B. Faltings. Oss: A semantic similarity function based on hierarchical ontologies. In *Proc. of IJCAI'07*, pages 551–556, 2007.

[135] Hinrich Schutze. Automatic word sense discrimination. *Computational Linguistics*, 24(1):97–123, 1998.

[136] Satoshi Sekine and Javier Artiles. Weps 2 evaluation campaign: overview of the web people search attribute extraction task. In *proc. of the 2nd Web People Search Evaluation Workshop (WePS 2009) at 18th International World Wide Web Conference*, 2009.

[137] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[138] F. Smadja. Retrieving collocations from text: Xtract. *Computational Linguistics*, 19(1):143–177, 1993.

[139] R. Snow, D. Jurafsky, and A.Y. Ng. Learning syntactic patterns for automatic hypernym discovery. In *Proc. of Advances in Neural Information Processing Systems (NIPS) 17*, pages 1297–1304, 2005.

[140] R. Snow, D. Jurafsky, and Y. Ng. Learning syntactic patterns for automatic hypernym discovery. In *Proc. of NIPS'05*, 2005.

[141] Y. Song, J. Huang, I.G. Councill, J. Li, and C.L. Giles. Generative models for name disambiguation. In *Proc. of International World Wide Web Conference (WWW)*, pages 1163–1164, 2007.

[142] M. Strube and S. P. Ponzetto. Wikirelate! computing semantic relatedness using wikipedia. In *Proc. of AAAI' 06*, 2006.

[143] J. B. Tenenbaum. Bayesian modeling of human concept learning. In *NIPS'99*, 1999.

[144] P. Turney. Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews. In *Proc. of ACL'02*, pages 417–424, 2002.

[145] P. Turney. A uniform approach to analogies, synonyms, antonyms, and associations. In *Proc. of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 905–912, 2008.

[146] P. D. Turney. Mining the web for synonyms: Pmi-ir versus lsa on toefl. In *Proc. of ECML-2001*, pages 491–502, 2001.

[147] P. D. Turney. Minning the web for synonyms: Pmi-ir versus lsa on toefl. In *Proc. of ECML-2001*, pages 491–502, 2001.

[148] P.D. Turney. Measuring semantic similarity by latent relational analysis. In *Proc. of IJCAI'05*, pages 1136–1141, 2005.

[149] P.D. Turney. Expressing implicit semantic relations without supervision. In *Proc. of Coling/ACL'06*, pages 313–320, 2006.

[150] P.D. Turney. Similarity of semantic relations. *Computational Linguistics*, 32(3):379–416, 2006.

[151] P.D. Turney and M.L. Littman. Corpus-based learning of analogies and semantic relations. *Machine Learning*, 60:251–278, 2005.

[152] P.D. Turney, M.L. Littman, J. Bigham, and V. Shnayder. Combining independent modules to solve multiple-choice synonym and analogy problems. In *Proc. of RANLP'03*, pages 482–486, 2003.

[153] A. Tversky. Features of similarity. *Psychological Review*, 84:327–652, 1977.

[154] A. Tversky. Features of similarity. *Psychological Review*, 84(4):327–352, 1997.

[155] V. Vapnik. *Statistical Learning Theory*. Wiley, Chichester, GB, 1998.

[156] T. Veale. The analogical thesaurus. In *Proc. of 15th Innovative Applications of Artificial Intelligence Conference (IAAI'03)*, pages 137–142, 2003.

[157] T. Veale. Wordnet sits the sat: A knowledge-based approach to lexical analogy. In *Proc. of ECAI'04*, pages 606–612, 2004.

[158] T. Veale and M. T. Keane. The competence of structure mapping on hard analogies. In *Proc. of IJCAI'03*, 2003.

[159] B. Vlez, R. Wiess, M.A. Sheldon, and D.K. Gifford. Fast and effective query refinement. In *Proc. of 20th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pages 6–15, 1997.

[160] D. McLean Y. Li, Zuhair A. Bandar. An approach for measuring semantic similarity between words using multiple information sources. *IEEE Transactions on Knowledge and Data Engineering*, 15(4):871–882, 2003.

[161] D. Zelenko, C. Aone, and A. Richardella. Kernel methods for relation extraction. *Journal of Machine Learning Research*, 3:1083–1106, 2003.