CAPTURING SEMANTICS USING A LINK ANALYSIS BASED CONCEPT EXTRACTOR APPROACH

by

SWARNIM KULKARNI

B.E., DEVI AHILYA VISHWAVIDHAYALAYA, 2007

A THESIS

submitted in partial fulfillment of the
requirements for the degree

MASTER OF SCIENCE

Department of Computing and Information Sciences
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas
2009

Approved by:

Major Professor
Dr. Doina Caragea

Copyright

Abstract

The web contains a massive amount of information and is continuously growing every day. Extracting information that is relevant to a user is an uphill task. Search engines like Google ^{TM}, Yahoo! ^{TM} have made the task a lot easier and have indeed made people much more "smarter". However, most of the existing search engines still rely on the traditional *keyword-based* searching techniques i.e. returning documents that contain the keywords in the query. They do not take the associated semantics into consideration.

To incorporate *semantics* into search, one could proceed in at least two ways. Firstly, we could plunge into the world of "Semantic Web", where the information is represented in formal formats such as RDF, N3 etc which can effectively capture the associated semantics in the documents. Secondly, we could try to explore a new semantic world in the existing structure of World Wide Web (WWW). While the first approach can be very effective when semantic information is available in RDF/N3 formats, for many web pages such information is not readily available. This is why we consider the second approach in this work.

In this work, we attempt to capture the semantics associated with a query by first extracting the concepts relevant to the query. For this purpose, we propose a novel **L**ink **A**nalysis based **C**oncept **E**xtractor (LACE) that extract the concepts associated with the query by exploiting the meta data of a web page. Next, we propose a method to determine relationships between a query and its extracted concepts. Finally, we show how LACE can be used to compute a statistical measure of semantic similarity between concepts. At each step, we evaluate our approach by comparison with other existing techniques (on benchmark datasets, when available) and show that our results are competitive with existing state of the art results or even outperform them.

Table of Contents

List of Figures

List of Tables

Acknowledgments

I, hereby take this opportunity to thank all those people whose expertise and inspiration helped me immensely in the successful completion of my thesis.

Above all, I would like to thank my major advisor, Dr. Doina Caragea, Assistant Professor in *Computing and Information Sciences* at KSU, for her ingenuous help and support during the course of this project. My detailed discussions with her on a crude concept helped me explore the associated intricacies and also some of the possible flaws in the concept. She not only helped me technically, but she has also been a constant source of inspiration for me. As she says - *"Research is going from failure to failure without getting disappointed"* which is an adaptation of quote from Winston Churchill. These words have always helped me in overcoming my failures and constantly progressing towards the ultimate goal.

I would also like to thank my committee members, Dr. Gurdip Singh, Head of Department and Professor in *Computing and Information Sciences* at KSU, and Dr. Torben Amtoft, Associate Professor in *Computing and Information Sciences* at KSU, for helping me in my work with their expert ideas and inspiration.

Finally, I would like to thank my colleagues, Mandar Haridas, Snehal Monteiro, Vishal Bahirwani and Surbhi Mungre for their valuable suggestions and comments on my work.

This work has been funded by National Science Foundation NSF grant number 0711396.

Dedication

I dedicate this thesis to my father, Mr. Narendra Kulkarni and my mother, Mrs. Seema Kulkarni who have always taught me to take failures in life as a challenge and always keep moving ahead to achieve greater milestones. In their words *"The Greatest pleasure is in doing things that people say you cannot do!"*.

Chapter 1

Introduction

The advent of World Wide Web (WWW) has imposed a deep impact on our way of thinking and our perspective of "information". It was not long ago when people faced problems because there was not sufficient information available. However, with World Wide Web, we now face the problem of information explosion. It has become very important to devise techniques that can be used to extract information that is "relevant" to the user. Search engines such as Google^{TM}, Yahoo^{TM} have solved this problem to an extent and have provided indeed smarter solutions to the problem of information extraction. Even though successful, these search engines until date, are mostly based on plain keyword based searching and return documents that contain the word present in the query posted by the user. They do not take the semantics associated with the query into consideration. However, recently top search engines such as Google^{TM} have started working towards displaying results considering the semantics of the query. For example, if we search on Google for "$\sim$ mobile", we get "nokia.com" as the topmost hit.

This work aims to provide a "semantic" touch to modern information retrieval techniques. The power of semantics can be exploited in at least two ways. Firstly, we can capture semantics by letting our search engine work on documents formally represented in forms such as RDF, N3 etc which are the de facto standards for representation in the semantic world. The data in such formats can be queried efficiently using formal languages like SPARQL. Moreover, an inference engine can also generate useful "non-obvious" inferences

from the data.

Alternatively, we can explore opportunities to infer semantics from the current structure of the World Wide Web. Because of lack of semantic data available in appropriate formats, we decided to move with the second approach. We exploit the power of existing search engines and use the Web as the main knowledge base for all our approaches.

In order to capture the semantics associated with the query, we propose a Link Analysis based Concept Extractor (LACE) that extracts concepts related to a query posted by the user and generates a concept cloud for the query. After we extract a set of related concepts to the query, we determine their relationships to the original query. In order to determine these relationships, we propose to use a collection of three well known knowledge bases namely, WordNet, Wikipedia and Web Directories. We use Wordnet and Wikipedia to determine synonym relationships and Web Directories to determine other types of relationships between words, if found that they are not synonyms of each other. Finally, we also propose LACE to quantify the semantic relatedness between words. To do that, we compute the similarity between the concept clouds generated by LACE for the two words to determine a statistical measure of semantic similarity between them. For each of the above proposed methods, we perform a comprehensive evaluation by comparison with the existing techniques to prove that they outperform all of them and also handle some of their flaws.

The rest of the thesis is organized as follows. Chapter 2 gives an introduction of LACE and shows how it can be used to generate concepts related to the query. Chapter 3 deals with determining the relationship between the generated concepts and the posted query. Chapter 4 demonstrates the use of LACE to compute degree of semantic relatedness between two words. Conclusions of the work and some future directions for research are provided in Chapter 5.

Chapter 2

Extracting Semantically Related Concepts

The problem of identifying concepts semantically related to a set of keywords has received a lot of attention in the recent years. Solutions to this problem can be useful for many practical applications, from query reformulation in information retrieval to text categorization. In this chapter, we present a Link Analysis based Concept Extractor (LACE) that identifies concepts semantically related to a given keyword query. Our approach exploits the richness of the web as well as the power of top search engines, and combines them with the elegance of *PageRank* to obtain high quality results. In particular, the inclusion of *PageRank* ensures that the extracted concepts come from high quality pages. A comparison of our method to other "state of the art" concept detection methods shows that LACE produces results similar or better than those produced by other methods. At the same time, LACE addresses some of the limitations of the previous approaches.

2.1 Background and motivation

Keywords and concepts share a many-to-many relationship [Ozcan and Aslandogan, 2005]. That is, a particular keyword can belong to many concepts and similarly, a particular concept can be represented by several words. Usually, user queries are short and ambiguous, making the task of differentiating between the relevant and the irrelevant information very difficult.

Techniques for identifying concepts semantically related to a set of keywords can be very useful for this and other applications, as described below:

- Knowledge of semantically related concepts can result in better precision and recall, when answering a user query. This can be achieved by refocusing the search, so that the retrieved documents do not simply contain the user keywords, however the meaning intended by the user, hence, eliminating results that do not match the related concepts.

- Semantically related concepts can also be used for text clustering and categorization [Hotho et al., 2003; Xu and Gong, 2004]. Similar concepts are, in a broader sense, the categories into which the generated results can be grouped.

- Furthermore, we can address the problem of ambiguity of keywords by handling the ambiguous word(s) in the context of the neighbors [Canas et al., 2003; Lee et al., 2000].

- At last, concepts related to the query issued by the user can be used to identify and display only the most relevant advertisements.

2.2 Related work

The problem of identifying semantically related concepts from keywords has received considerable attention in the last few years, and as a result, has seen continuous improvements.

The initial work on predicting semantic relatedness and deriving similar concepts was done using existing lexical databases such as Roger's Thesaurus [Jarmasz and Szpakowicz, 2003] and WordNet [Banerjee and Pedersen, 2003; Finkelstein et al., 2002; Wu and Palmer, 1994]. However, Strube and Ponzetto [2005] have shown that Wikipedia can easily outperform WordNet in calculating the semantic similarity, when a variety of approaches to semantic relatedness, including paths in graph and the text content in the articles, are employed.

In fact, Wikipedia has been used as a source of concepts by many researchers. Syed et al. [2008] have used the articles from Wikipedia along with the category and link graphs to identify concepts which are common to a set of documents. The category graph in Wikipedia helps to predict the general concepts, whereas the article link graph can be used to predict concepts which are not in the category graph. Gabrilovich and Markovitch [2007] have proposed an approach called "Explicit Semantic Analysis (ESA)" in which concepts derived from Wikipedia are used to represent the meaning of any text and compute the semantic relatedness between parts of natural language text. In particular, machine learning techniques are used to "explicitly" represent the meaning of any text as a weighted vector. According to Gabrilovich and Markovitch [2007], the ESA results are better than the results of other existing state of art approaches. Milne [2007] has used an approach similar to the ESA approach. In Milne's approach, the semantic relatedness between terms is computed on the basis of links found between terms' corresponding articles on Wikipedia, however the underlying text within an article is not processed. Although successful, the above approaches make use of Wikipedia as knowledge base, and thus, face the following issues: (1) Not every word has a category graph associated with it; (2) Links from a relevant document may not always point to another relevant document.

The technique of link analysis has been widely used to tackle a variety of problems. Cai et al. [2004] have used it to extract the semantics of a web page, by first dividing a web page into blocks using vision-based segmentation, followed by the creation of a semantic graph, where each node represents exactly one semantic topic. Nakayama et al. [2008b] have applied a similar approach to the link structure of Wikipedia to extract the semantic relationships between terms. Moreover, link analysis has also been used for query reformulation and query expansion purposes.

Our approach is similar in spirit to the approach proposed by Leelapatra and Netisopakul [2008], who have used the HITS algorithm for the purpose of query expansion. One main difference is that we use the PageRank algorithm to extract concepts related to the given

query keywords, as the effectiveness of PageRank to compute semantic relatedness has already been demonstrated by its use on existing lexical databases [Mihalcea et al., 2004]. In case the relatedness results need to be further used for query expansion or reformulation, we also show how to minimize the amount of computation to be done after the query is issued by the user, so that the final results can be returned as quickly as possible. However, as described in the Chapter 1, the relatedness results can be used with other applications as well. In addition to the related concepts, we also return the PageRank scores of the corresponding web pages to ensure that concepts from high-quality web pages receive higher weighs compared to the weights of the concepts from lower quality web pages. To the best of our knowledge, a link analysis approach over the whole web graph structure, making use of PageRank and meta data of a web page, has not yet been used for the purpose of identification of related concepts to a given query.

2.3 Link Analysis based Concept Extractor (LACE)

The steps of the concept extraction process are shown in Figure 2.1. The process starts by taking a query (i.e., set of keywords) from a user via the *User Interface* as shown in Figure 2.7 and Figure 2.8. The *Query Reformulation* module constructs various combinations of keywords for the given query. To understand the importance of this step, consider the following scenario: two users, user A and user B, are interested in learning about the president of USA.

User A enters the keywords "President USA", whereas user B enters the keywords "USA President". Surprisingly, by analyzing the top 10 results returned by Google^{TM} for the two searches, we see that a simple word swap results in three unique relevant links in each search. Hence, by considering all possible combinations of keywords, we ensure that a larger area in the concept space, relevant to the given query, is covered. Hence, in the above example, the output of Query Reformulator is *President USA USA President*. Once the initial query is reformulated, in the next step of the process, the *Link Extractor* feeds all possible query

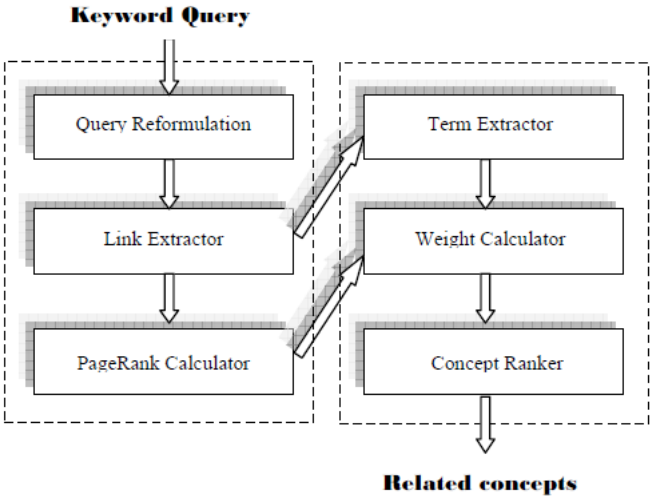


Figure 2.1: *Concept extraction pipeline*

```
Sending to yahoo..

Links Extracted :
http://en.wikipedia.org/wiki/List_of_Presidents_of_the_United_States
http://en.wikipedia.org/wiki/President_of_the_United_States
http://www.conservapedia.com/President_of_the_United_States_of_America
http://www.dkosopedia.com/wiki/Vice_President_of_the_United_States_of_America
http://www.presidentsusa.net/presvplist.html
http://www.serve.gov
http://www.usa-presidents.info
http://www.usa-presidents.info/grant.htm
http://www.usatoday.com/news/politics/election2008/president.htm
.....
```

Figure 2.2: *Output of Link extractor*

reformulations to a search engine and extracts the resulting links, thus creating a document corpus (which is stored locally). Yahoo!^{TM} Search engine was used at this step. The number of pages to be extracted for each possible query formulation can be set by the user. The output for the *link extractor* is as shown in Figure 2.2. The created document corpus is accessed by the *PageRank Calculator* module, which extracts the links associated with the documents in the corpus and first creates a graph from them. It then calculates the

```
Now computing PageRank scores....
Sorted PageRank values are:
http://en.wikipedia.org/wiki/List_of_Presidents_of_the_United_States
0.034501292670803776
http://en.wikipedia.org/wiki/President_of_the_United_States
0.034501292670803776
http://www.conservapedia.com/President_of_the_United_States_of_America
0.034501292670803776
http://www.dkosopedia.com/wiki/Vice_President_of_the_United_States_of_America
0.034501292670803776
http://www.presidentsusa.net/presvplist.html 0.034501292670803776
http://www.serve.gov 0.034501292670803776
```

Figure 2.3: *Output of PageRank calculator*

PageRank scores from this graph. The default PageRank values for each of the web pages is 1. The output for the *PageRank Calculator* for the query "President USA" is as shown in Figure 2.3. For the given query, in the created graph for a set of 30 web pages(top 30 extracted results), 15 were found to be interlinked. Next, the *Term Extractor* module extracts the terms from the top n pages (i.e., those with highest PageRank scores). The parameter n can also be set by the user. Its default value is 5. Rather than extracting all terms from the text of a web page, we exploit the meta information of a web page. More precisely, we extract the meta keywords along with the title of a web page, as preliminary experimentation with the system has shown that these terms capture the context of a web page in most cases. The output for *Term Extractor* module for the given example is as shown in Figure 2.4. The terms collected by the *Term Extractor* are next sent to the *Weight Calculator* module. This module uses the PageRank scores of the corresponding web pages to calculate the weights of the terms. Specifically, the weight of a term is calculated as follows: Let t be a term and w_t the weight corresponding to the term. Let P_i denote the PageRank score of the web page i containing the term t. Let k be the number of web pages containing the term t. Then: $w_t = \frac{\left(\sum_{i=1}^{k} P_i\right)}{N}$, where N denotes the total number of documents in the document corpus. Finally, the *Concept Ranker* module ranks the terms

```
Now working on :http://dubai.usconsulate.gov/obama_president2.html
Now working on :http://www.usa-presidents.info/grant.htm
I extracted : presidents, presidency, usa, america, biography, ulysses
simpson grant
Token : presidents
Token :  presidency
Token :  usa
Token :  america
Token :  biography
Token :  ulysses simpson grant
Now working on :http://www.usa-presidents.info
I extracted : presidents, presidency, usa, america, biography
Token : presidents
Token :  presidency
Token :  usa
Token :  america
Token :  biography
```

Figure 2.4: *Output of Term extractor*

in the descending order of their weights. The number of concepts to be displayed depends on the threshold value set by the user. The *Concept Ranker* module takes the concepts and their weights as input and ranks them in decreasing order of their weights. The output for the *Weight Calculator* and *Concept Ranker* module is as shown in Figure 2.5.

In order to speed up the retrieval of the results, we propose to crawl the pages offline and calculate the PageRank scores before taking the query from the user. Figure 2.2 specifies the architecture of such a model. It is assisted by a *Crawler* that can crawl a large number of web pages for a specific domain from a knowledge base (e.g., DMOZ) and then feeds the collected corpus to the *Graph Builder*. The *Graph Builder* creates a graph from the corpus by accounting for all incoming and outgoing links from each web page. The created graph is then fed to the *PageRank Calculator* module, which calculates the PageRank for each web page as follows: Let A be a web page in the collection. The PageRank of A is calculated recursively by the PageRank of the pages that point to A. That is,

$$PR(A) = \frac{(1-d)}{N} + d\sum_{i=1}^{n} \frac{PR(w_i)}{C(w_i)}$$

```
Wts calculated for 356 tokens

Sorted values are:
barack obama 0.005750215445133962
abraham lincoln 0.004541774389233647
barack 0.0033917313002068544
president 0.0033917313002068544
usa 0.0033917313002068544
44th president 0.0023000861780535 85
al gore 0.0023000861780535 85
america 0.0023000861780535 85
andrew jackson 0.0023000861780535 85
andrew johnson 0.0023000861780535 85
barak 0.0023000861780535 85
barck 0.0023000861780535 85
barek 0.0023000861780535 85
barrack 0.0023000861780535 85
```

Figure 2.5: *Output of Weight calculator and Concept ranker*

where $w_1, w_2, \cdots, w_n$ specify the pages that point to the page A, $PR(w_i)$ denotes the PageRank of the web page w_i and $C(w_i)$ denotes the number of outgoing links from the page. The PageRank algorithm is based on a random surfer model. This model assumes an imaginary surfer who randomly goes from one link to another on a webpage. The probability of the person continuing is given by the damping factor 'd'. Since the surfer jumps to another page at random after he stopped clicking links, the probability therefore is implemented as a constant (1-d) into the algorithm. In our system, the value of d is set to 0.85.

Our current implementation is based on the architecture shown in Figure 2.1. However, if the *PageRank Scorer* system shown in Figure 2.2 would be implemented, the *PageRank Calculator* module in Figure 2.1 would be replaced by the *PageRank Scorer* system in Figure 2.2. The benefit of the architecture in Figure 2.2 is that we get a better connectivity of the graph as we now consider a larger set of links to form the graph. Hence, the PageRank scores are more reliable.

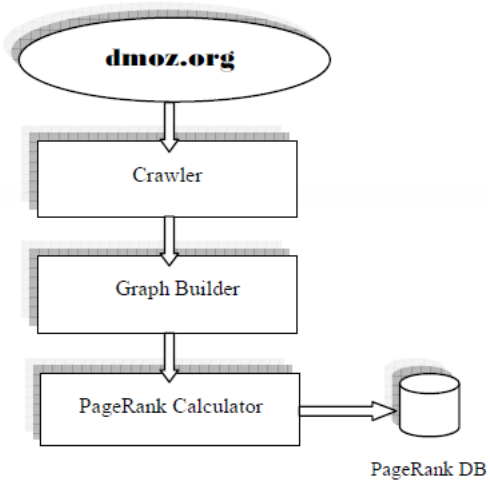


Figure 2.6: *The PageRank scorer system*

Figure 2.7: *LACE web interface. The user can post the query using the interface and extract associated concepts to the query*

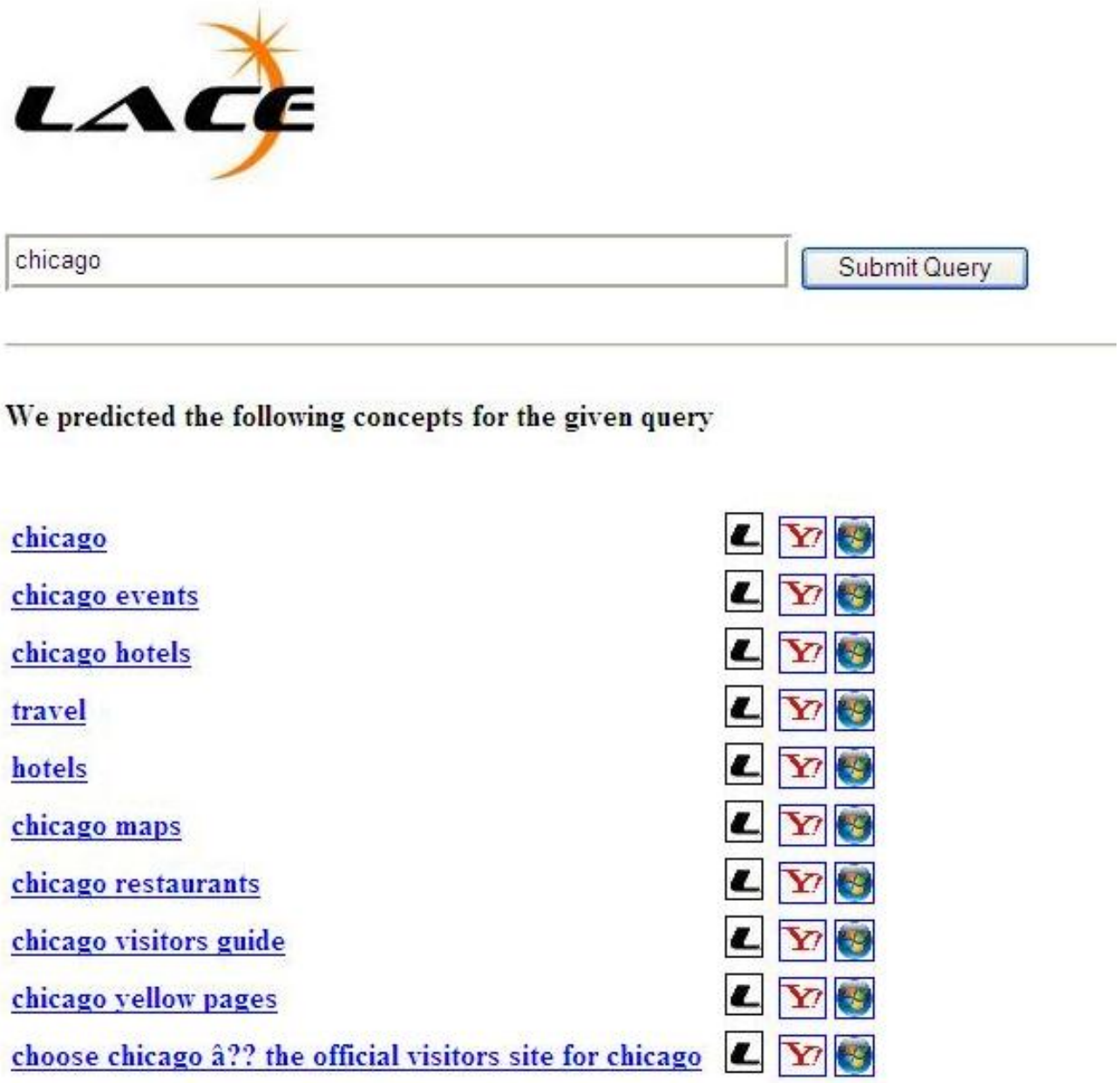


Figure 2.8: *Output of LACE for the query "Chicago". The semantically related concepts to the query are displayed to the user*

2.4 Experimental design and results

To evaluate our approach, we conducted six experiments, each experiment aiming at testing the system under a different scenario. In the first experiment, we used a single well-defined concept as the query. In the second experiment, we provided the system with similar keywords from a specific domain. In the third experiment, we tested the system by providing the name of a person as the query. In the fourth experiment, we provided a query with a spelling error in it. The goal of the fifth experiment was to test the ability of the system to perform word sense disambiguation. In the last experiment, we tested the ability of the system to find concepts related to a considerably long query. More details on the six experiments are provided below. The results for the experiments are presented in Table 1 to Table 6, respectively. For each of the results, we see that as we move down in the ranking list, the extracted concepts become less relevant to the original query. Further experimentation can be carried out in order to determine the best threshold value for the concepts. The results for the concept data collected is as of March 9th, 2009.

Query = "*Flu*"	
1	Influenza
2	Flu
3	Flu shot
4	Flu vaccine
5	Avian flu
6	Cold
7	Acetaminophen
8	Amantadine
9	Antigenic drift
10	Antigenic shift

Table 1: Single-word query

Query = "*Mars Venus Earth*"	
1	Mars
2	Earth
3	Moon
4	Sun
5	Jupiter
6	Mercury
7	Neptune
8	Pluto
9	Saturn
10	Solar System

Table 2: Set-based query

Experiment 1: Single-word specific query

In this basic experiment, we tested our system on a single-word query, specifically "Flu". Concepts related to "Flu" were extracted. The results of the experiment are shown in Table 1.

Experiment 2: Set-based query

In the second experiment, the system was tested on a query formed with similar keywords belonging to a single specific domain. The purpose of this experiment was to simulate the behavior of Google Sets[1]. For example, for the query "Mars Venus Earth", the common domain is "Heavenly bodies". We expect the system to give us the names of other heavenly bodies, as they belong to the same domain. The results are shown in Table 2 and they are comparable with the results obtained with Google Sets.

Experiment 3: Name query

In this experiment, we provided the system with the name of a person and expected to extract all possible concepts which are related to that person. For example, the input in this experiment was "Christopher Manning IR".

Query = "*Christopher Manning IR*"	
1	Chris manning
2	Christopher d. manning
3	Computer science
4	Data mining
5	Ergativity: argument structure and grammatical relations
6	Foundations of statistical natural language processing
7	Introduction to information retrieval
8	IRS
9	Retrieval
10	Search

Table 3: Name query

[1] http://labs.google.com/sets

Query = "*Contectiivtis*"	
1	Conjunctivitis
2	Pinkeye
3	Allergic conjunctivitis
4	Chlamydia
5	Eye infection
6	Infectious
7	Bacterial conjunctivitis
8	Eyelid
9	Infectious
10	Itching

Table 4: Misspelled-word query

He is the first author of an information retrieval textbook. We expect the related concepts to include his field of study, his publications topics or any other concepts related to him. The results of this query are displayed in Table 3.

Experiment 4: Misspelled-word query

In this experiment, we provided the system with a query consisting of a misspelled word and analyzed its behavior on such an input. As expected, the system was able to determine

Query = "*Leopard OS*"	
1	Apple
2	Leopard
3	Mac OS X
4	Mac OS History
5	Operating System
6	32-bit
7	64-bit
8	Software
9	.mac
10	10.5

Query = "*Leopard Animal*"	
1	Animals
2	Leopard
3	Snow Leopard
4	Mammals
5	Wildlife
6	Aardwolf
7	Acinonyx
8	Reptiles
9	Snakes
10	2005

Table 5: Ambiguous word query

Query = "*New York Statue of Liberty Manhattan*"	
1	Statue of liberty
2	Ellis island
3	Liberty
4	New york tour
5	New york city
6	Staten island
7	Statue of liberty ticket
8	Tourism
9	Travel
10	Tickets

Table 6: Long Query

the correct version of the query and to extract concepts related to the correct version. The results of such a query, specifically "Contectiivtis", which is a misspelling of the term "Conjunctivitis", are displayed in Table 4.

Experiment 5: Ambiguous word query

In the fifth experiment, we tested the power of our system to analyze the meaning of an ambiguous word on the basis of its neighbors. For example, in the given query, the term "Leopard" is an ambiguous term as it can refer to an operating system as well as an animal. As expected, our system is able to accurately extract concepts related to the term "Leopard" on the basis of its neighboring terms. The results are shown in Table 5.

Experiment 6: Long query

In this final experiment, the system was tested on a longer query and was expected to extract concepts which are related to all the terms in the query and not just a few of them. The results for this experiment are shown in Table 6.

2.5 Comparison with other approaches

In addition to the evaluation described in the previous section, we also compared our system, LACE, with two other similar existing "state of the art" systems, specifically ESA [Gabrilovich and Markovitch, 2007] and WikiRelate! [Strube and Ponzetto, 2005], along several dimensions. The comparison is shown in Table 7.

Criteria	**LACE**	**ESA**	**Wiki**
KB specific	No	Yes	Yes
Error correction	Yes	No	No
Length of query	Multiple	Multiple	Single
Handle name query	Many	Few	Few

Table 7: Comparison with other approaches

As we can see, LACE not only provides results which are comparable to the results provided by other similar concept detection systems, however it is also able to handle some of the limitations of these approaches. The biggest strength of LACE is that it is not dependent on a single knowledge base. Therefore, in theory, it should work on an "infinite" vocabulary. Moreover, as our results show, LACE is capable of handling errors in the user query and also gives accurate results for considerably long queries. Lastly, it also has the capability to generate results for many name queries, which other concept detection algorithms are not able to do as their vocabulary is limited (e.g., to names described in Wikipedia). LACE is able to extract concepts for many name queries, as there is always a higher probability to find information about any person on the web than in any specific vocabulary such as Wikipedia.

2.6 Discussion

The effectiveness of the proposed method has been proved by comparing it to other "state of art" concept detection algorithms. Our approach has the following advantages :

- It is not specific to a particular knowledge base such as Wikipedia or WordNet as we use the web as our underlying knowledge base.

- It is capable of handling the user spelling mistakes made when providing the input to the system.
- There is no limit on the length of the query to be fed in by the user.
- The inclusion of PageRank scores in calculation of weights of concepts makes our approach resistant against techniques such as "keyword stuffing", which are sometimes employed by web pages to achieve a higher rank in search results.
- It is a simple approach with high quality results which are comparable to the results of other "state of the art" approaches.

In our approach, we have used the meta information of a web page for extraction of concepts, as we have observed that the meta tags capture the context of a web page well in most cases. However, in few other cases, with techniques such as "keyword stuffing", misleading keywords can easily be inserted into the meta tags of a web page. We implicitly handle this issue in two ways. Firstly, our system is based on the results returned by existing search engines and modern search engines have the capability to combat such techniques. Secondly, our system is based on the PageRank approach. Hence, we do value concepts from high quality pages more than concepts from lower quality pages. Therefore, in every case, we ensure that only the best quality results are returned to the user.

Chapter 3

Predicting relationships between pair of words

In the previous chapter, we showed how search engines and PageRank can be used to determine semantically related concepts to the query. In this chapter, we propose R-LACE i.e. *Relationship predictor* LACE that finds the relationships between the generated concepts and the given query and represents them in the form of a graph. Moreover, it also stores the generated results formally, in the form of RDF triples, to enable better inferences from them as compared to a traditional search engine. We evaluate our system by comparing it with other similar 'state of art' relationship identification systems and prove that R-LACE produces results which are either similar or better than those generated by these systems.

3.1 Background and motivation

The Semantic Web is an extension of the present web, with the representation of the information in machine readable format. It can be seen as a platform for information and knowledge exchange for both, humans and machines [Minack et al., 2008]. It involves adding 'metadata' to the existing information and express it in more formal structures such as RDF and FOAF. RDF, schemas and inference languages represent all the data in the form of a huge database that can be queried efficiently using languages such as SPARQL. The World Wide Web is the present whereas the Semantic Web is the future. Both hold immense

potential in them and this is the reason why the problem of developing a bridge between the two has allured a lot of researchers.

3.2 Related work

Accurately determining relationships between words can be useful in a wide range of applications such as query expansion [Buckley et al., 1994; Mitra et al., 1998; Vlez et al., 1997] in which the synonymous words can be used to modify the query posted by the user or to suggest a newer query, based on its similarity with the previous queries. Moreover, they can also be used for community mining [Matsuo et al., 2006a; Mika, 2005] and for natural language processing tasks such as language modeling [Rosenfield, 1996] and word sense disambiguation [Resnik, 1999]. Many approaches have been used to identify relationship between words from a given text using manually compiled semantic database such as WordNet [Jiang and Conrath, 1998; Lin, 1998a,b]. However, because of limitations on the size of the vocabulary of WordNet, recently researchers have started exploiting power of Web as the Knowledge Base. Sun et al. [2007] used the web to acquire semantic relationships between words. Turney [2001] identified synonyms based on the number of hits returned by a Web Search Engine. Matsuo et al. [2006b] used a similar approach to determine the similarity between words and applies their approach for graph based clustering algorithms.

Web search engines were used by Sahami and Heilman [2006] to address the problem of determining similarity between texts that do not share any word in common. Bollegala et al. [2007a] exploit the page counts and text snippets generated by a Web Search engine to determine the semantic similarity between words. Their approach calculates the similarity by using automatically extracted lexico-syntactic patterns from text snippets. PageRank-like techniques have also been used to calculate the similarity measure. Chen et al. [2006] proposed to exploit the text snippets returned by Web Search Engine as an important measure in computing the semantic similarity between words. In order to determine semantic similarity between two words A and B, the text snippets for the two words are collected and

the occurrence for word A is counted in the snippet for word B and vice versa. However, for two words A and B to be similar, they need not always appear in the text snippets of each other.

Some work has also been done on the problem of bridging the gap between the Web and the Semantic Web. Angeletou et al. [2007] propose enrichment of folkosonomy tags by harvesting the Semantic Web for explicit relations. They use online ontologies to dynamically collect and combine bits of knowledge. To the best of our knowledge, an approach to bridge the gap between the Web and the Semantic web by making use of web search engines and relationship identification algorithms has not been proposed yet.

3.3 System architecture

A Pipeline approach has been used to identify concepts for a given query, extract their relationships to the original query and then represent it as an RDF Graph. The architecture for R-LACE is shown in Figure 3.1. The System consists of four modules: Concept Extractor, Relationship Identifier, Graph Builder and RDF Generator. The system takes the query from the user via the *User Interface*. The query is fed to the *Concept Extractor* module that extracts the concepts related to the given query. After all the concepts have been extracted for a given query, the query and its generated concepts are used by the *Relationship Identifier* module which determines a relationship between the query and its associated concepts.

The *Graph Builder* takes the relationships identified by the Relationship Identifier and generates a semantic graph from it that is presented to the user. The *RDF Generator* interprets the identified relationships in the form of a "triple" and creates an RDF document out of it, which can be queried by R-LACE at a later stage using languages such as SPARQL to generate "meaningful" inferences from it.

Now we describe each of the modules of the processing pipeline in detail.

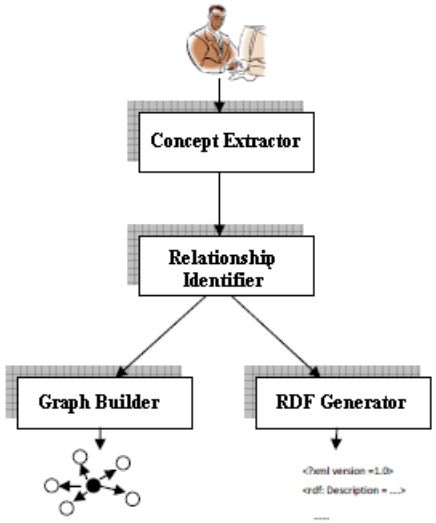


Figure 3.1: *R-LACE architecture*

3.3.1 Concept extractor

The Concept Extractor is used to extract concepts that are semantically related to the query. The details for the Concept Extractor are given in Chapter 2.

3.3.2 Relationship identifier

The architecture for the *Relationship Identifier* is shown in Figure 3.2. In our approach, we make use of three well known knowledge bases namely WordNet, Wikipedia and Yahoo!^{TM} Directories in order to identify the relationships between concepts.

The system gets the generated concepts from the *Concept Extractor* as an input. Each of the concept is paired with the original query to form several word pairs. The formed pair is sent to the WordNet database. We installed WordNet 3.0 locally and used the available API to access the database efficiently. If a relationship can be determined, the relationship is displayed to the user and the program terminates. However, if a relationship could not be determined but the concept was found on WordNet, we store all the available information from WordNet such as hypernyms, hyponyms and antonyms and move to a larger data source, Wikipedia to determine the relationship. If Wikipedia also fails to determine

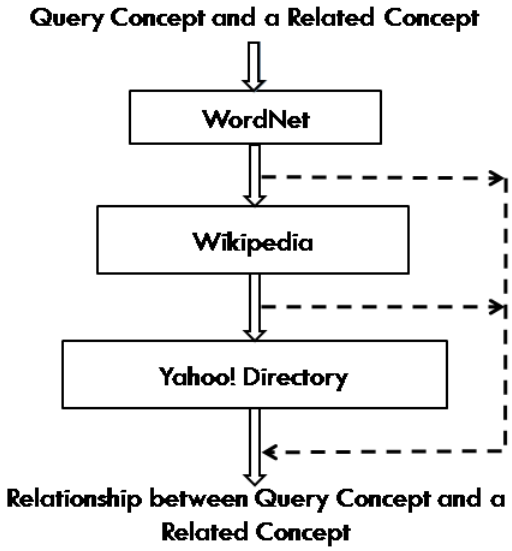


Figure 3.2: *Architecture for Relationship identifier*

a relationship, we use the category structure of *Yahoo!*^{TM} *Directory* for the relationship identification task. We now demonstrate using examples, how the unique capability of each data source can be useful in developing a system that has the power to function accurately and with high efficiency in almost all cases.

Determining relationship using WordNet

Consider the following example: The system gets as input the query "*Flu*". The *Concept Extractor* generated concepts for the query as: *Influenza, Flu Shot, Oseltamivir, Avian influenza, Hong Kong flu, Cold* and *Acetaminophen*. We start by determining whether the extracted concept is a synonym, hypernym or hyponym of the original query. We take the first concept "*Influenza*" and search for it on WordNet. Because it is present on WordNet, all possible synonyms, hypernyms and hyponyms for it are extracted and matched with the given query to identify a relationship. Since, *Flu* is present as synonym for *Influenza* on WordNet, it is tagged as a synonym. Hence, in this case, we were able to return the results efficiently without the need to look in larger data sources.

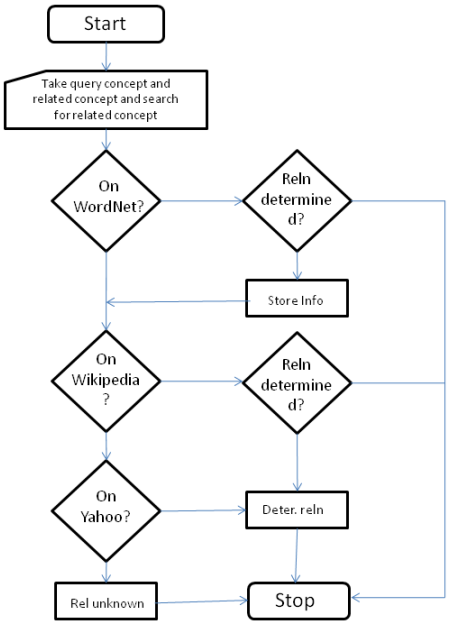


Figure 3.3: *Flowchart for Relationship identifier showing how relationships can be predicted between query concepts and the related concepts*

Determining relationship using Wikipedia

Consider the query: *Kansas State University*. The generated concepts are: *K-State, Kansas State, CIS, Ahearn field , Kansas, powercat, courses*. Using Wikipedia, we first only check whether the extracted concept and the query are synonyms of each other or not. If by searching for two concepts on Wikipedia, we retrieve the same page, we conclude that the two words are synonyms of each other. When the generated concept *K-State* and the query *Kansas State University* are searched on Wikipedia, they are redirected to the same page. Hence, we conclude that *K-State* is actually a synonym of *Kansas State University*.

In order to determine other relationships using Wikipedia, we extract the first sentence of the Wikipedia article as our preliminary experiments show that it usually well defines the title of the article in most cases. We then try to find the query word in the sentence so as to determine a relationship between the words. If we are able to do so, we return the relationship, otherwise we move to a larger data source, Yahoo! Directory.

In cases, when we are able to find the concept on WordNet but are unable to determine it's relationship to the query using it, we store the available information before moving to a higher data source because this information can be useful when we use the bigger data source. As an example, we wish to determine a relation between the query *Watery Eyes* and its related concept *Conjectivitis*. We could not determine a relation between them using WordNet but we found the term *Conjectivitis* and we stored its synonym *Pinkeye*. Now, when we moved to Wikipedia, we were able to determine the relation between *Watery Eyes* and *Conjectivitis* from the sentence *Watery eyes is a symptom of Pinkeye* because our system knows that *Pinkeye* is actually a synonym for *Conjectivitis*.

Determining relationship using Yahoo! Directory

Again, consider the query: *Kansas State University* and its associated concepts. We now show how a relationship can be determined using the category graph structure of Yahoo! Directories. We post *Kansas* as a query onto Yahoo!^{TM} Directories and extract the displayed categories from the results. We then search for the given words in the extracted categories and try to determine a relation between them. In our example, when the two concepts were posted onto Yahoo! Directories in the form of "*Kansas State University*""*Kansas*", we extract the relationship between them as *Kansas State University* –> *Manhattan* –> *Kansas*.

Hence, the triple layer in our model helps us to determine relationships in most of the cases. Specifically, we use WordNet to determine whether the concept and the query are synonyms, hypernyms and hyponyms of each other. We use Wikipedia mainly to determine if two words are synonyms or not, as Wikipedia is more comprehensive than WordNet. We still have WordNet in our model because if a word is found on WordNet, it is much easier and faster to determine its synonyms, hypernyms and hyponyms as compared to Wikipedia and Yahoo! Directories. Finally, we use Yahoo! Directories to determine a relationship between two concepts if it is found that they do not share a synonymy, hyponymy or hypernymy relationship.

3.3.3 Graph builder

Unlike traditional search engines that display results to the users in the form of relevant documents to the given query, we propose a more "semantic" way of representing the results. We present results to the users in the form of a Semantic Relationship Graph (SRG). The Semantic Relationship graph is a graph that displays the query and the associated concepts to the users in the form of a graph structure clearly displaying how each concept is related to the given query. An SRG is similar to a Semantic Network with the difference that in a Semantic Network, the relationship is plotted between every two nodes. However, in our case, we just present to the user how the query is related to its generated concepts. The generated SRG contains the query as the central node surrounded by its associated concepts and the edges representing the relationship between the nodes.

The *Graph Builder* takes the pairs of the concepts and the query and their relationships as input and builds an SRG from it that is presented to the user.

3.3.4 RDF generator

The final step of our pipeline is to generate an RDF from the identified relationships among concepts and store them, so that generated RDF can be queried by R-LACE. The relationships are represented in the form of triples <*Subject,Predicate,Object*>, where the *Subject* is the given query, *Object* is a related concept and the *Predicate* is the relationship between the *Subject* and the *Object*.

The RDF Generator takes the concepts and their relationships as input and creates an RDF document from them by using the determined relationships as name space values.

3.4 Experimental design and results

In order to evaluate our approach, we performed several set of experiments on individual modules of our system i.e. the Relation Identifier, the Graph Builder and the RDF Generator. We will present here the results from the Relationship Identifier, Graph Builder and

the RDF Generator. The results are shown in Table 3.1, Figure 3.4(a) and Figure 3.4(b) respectively.

3.4.1 Identifying relationships between query and the extracted concepts

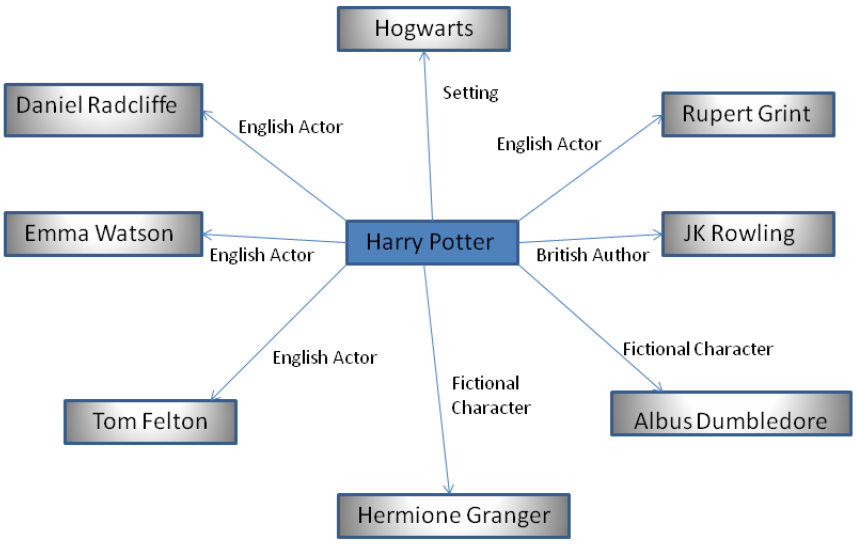
The generated concepts along with the associated query are provided as input to the *Relationship Identifier*. The results generated from the *Relationship Identifier* are displayed in Table 3.1. *Subject* refers to the query posted by the user and the generated concepts are referred as *Objects*. As demonstrated by the results, with our approach we are able to determine the relationship between the *Subject* and the *Object*. We are not only able to determine synonyms, hypernyms and hyponyms with a good accuracy, but were also able to identify other types of relationships.

3.4.2 Building an SRG

The output of the Graph Builder is shown in Figure 3.4. The Graph Builder takes the generated concepts and their relationships to the original query and displays them as a graph to the user. The results for the query *"Harry Potter"* are shown. The user even has the ability to explore the graph by clicking on a particular generated concept. The system then would display the graph to the user considering the clicked concept as a query and re-iterating it through the pipeline.

3.4.3 Representation in RDF format

The RDF generator also takes as input the identified relationships between the query and its extracted concepts and generates an RDF representation for it which is saved by the system. This generated RDF could be queried by an extension of LACE, R-LACE using formal query languages such as SPARQL and hence, can be used to perform "semantic" inferences from the generated result. For example, consider the sample RDF representation shown in Figure 3.4(b). For instance, it can answer the query - *Which English Actors have acted in Harry*



(a)

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
                xmlns:dc = http://purl.org/dc/elements/1.1
                smlns:exterms=http://www.example.org/terms/

        <rdf:Description rdf:about="Harry Potter">
        <exterms:EnglishActor>Daniel Radcliffe</exterms:EnglishActor>
        </rdf:Description>

        <rdf:Description rdf:about="Harry Potter">
        <exterms:EnglishActor>Rupert Grint</exterms:EnglishActor>
        </rdf:Description>

         <rdf:Description rdf:about="Harry Potter">
        <exterms:EnglishActor>Emma Watson</exterms:EnglishActor>
        </rdf:Description>

        <rdf:Description rdf:about="Harry Potter">
       <exterms:BritishAuthor>JK Rowling</exterms:BritishAuthor>
        </rdf:Description>

        .....
```

(b)

Figure 3.4: *(a) Output of Graph builder and (b) Generated RDF*

Subject	Predicate	Object	Subject	Predicate	Object
Sun Microsystems	programming languages	Java	Flu	synonym	Influenza
Sun Microsystems	microprocessors	Sparc	Flu	immunizations	Flu shot
Sun Microsystems	computer networking protocols	Jini	Flu	immunizations	Oseltamivir
Sun Microsystems	software	OpenOffice	Flu	super type	Avian Influenza
Sun Microsystems	computer hardware	Server	Flu	super type	Hong Kong flu
Sun Microsystems	unix	Solaris	Flu	diseases and conditions	Cold
Sun Microsystems	computer training	Training	Flu	drugs and medications	Acetaminophen
Subject	**Predicate**	**Object**	**Subject**	**Predicate**	**Object**
Kansas State University	synonym	K-state	Harry Potter	english actor	Daniel Radcliffe
Kansas State University	synonym	Kansas state	Harry Potter	english actor	Rupert Grint
Kansas State University	departments and programs	CIS	Harry Potter	fictional character	Albus dumbeldore
Kansas State University	athletics	Ahearn field house	Harry Potter	british author	JK Rowling
Kansas State University	manhattan	Kansas	Harry Potter	setting	Hogwarts
Kansas State University	athletics	Powercat	Harry Potter	fictional character	Hermione granger
Kansas State University	departments and programs	Courses	Harry Potter	english actor	Emma Watson

Table 3.1: *Results of Relationship identifier showing the relationships determined between the query concepts and the related concepts*

Potter? or a query such as *Who is Albus Dumbeldore?* using the generated RDF. The above queries can be answered by converting them into a "triplet" format that can be queried by an inference engine. For example, the query *Which English Actors have acted in Harry Potter?* can be converted into triplet format as <*X,EnglishActor,Harry Potter*>. Similarly, the query *Who is Albus Dumbeldore?* can be converted as <*X,Y,Albus Dumbeldore*>. The variables can be determined using the generated RDF and an inference engine. Details about the representation of RDF and querying the RDF to generate inferences from it will

Terms	**Reln by R-LACE**	**Reln by WikiC!**
Apple - Fruit	is	is
Cat - Mammal	is	is
Bird - Biped	no relation	is
Computer - Machine	is	is
Jimmy Snuka - Wrestler	is	is
Colorado - U.S.A	is in	is one of
Sharon Stone - Model	is	actor
Flintstones - Animated TV show	is	no relation
1980's music - Pat Benatar	Rock and Pop	no relation

Table 3.2: *R-LACE vs. WikiC!*

be published elsewhere.

3.5 Comparison to other approaches

We compared the performance of *Relationship Identifier* to other similar relationship determining algorithms [Nakayama et al., 2008a]. Since, the relation detection technique of R-LACE makes use of combination of three well known knowledge bases in a "smart" way, R-LACE outperforms other similar systems in identification of relationships between concepts. We compared our system to the one described in [Nakayama et al., 2008a], which is a Wikipedia based relationship extraction system, which we term as "WikiC!"'. We used several queries from [Nakayama et al., 2008a] and tested them on R-LACE including queries where WikiC! fails. The comparison is shown in Table 3.2.

We see that in most of the cases, R-LACE is able to identify the relationships similar to WikiC!. However, we could not find a relation in case of "Bird-Biped" because the category *Biped* was not present in Yahoo! Directory structure. R-LACE outperforms WikiC! when a relationship between terms is to be determined whose definition is not present on Wikipedia. For example, consider the two terms as *Animated TV Show* and *Flintstones*. We find a document for the latter but do not find one for the former on Wikipedia. Hence, in this

case WikiC! does not return a relationship. However, since *Animated TV Show* is present in Yahoo! Directories, R-LACE returns a relationship as *Flintstones is Animated TV Show*. As another example, consider the terms - *1980's music* and *Pat benatar*. Wikipedia has no article for *1980's music*, hence WikiC! does not return a relationship. However, in this case too, R-LACE returns the relationship as *rock and pop*.

Hence, we see that the relation detection algorithm of R-LACE outperforms other similar relation detection algorithms.

3.6 Discussion

In this chapter, we proposed a systematic approach to bridge the gap between the Web and Semantic Web. We make use of the power of existing search engines to extract the concepts and well known knowledge bases to determine relationships between the query and the generated concepts. We evaluate our results by comparing them with some of the already existing state of the art approaches that use Wikipedia as the knowledge base for the identification of concepts and identification of relationships between them. Our approach has the following advantages :

- The use of this approach in identification of relationships between concepts help in identifying not only the stronger relationships accurately, but also gives distant relationship between weakly related concepts.
- The displaying of results in a graphical format makes them easier to interpret for the user.
- Storing of the results in machine interpretable RDF format makes them easy to query for future similar queries and also to infer "semantic" from them.
- Our approach is a simple approach with high quality results which are comparable to the results of other state of art approaches.

However, our system also has some drawbacks. Most importantly, the identification of the relationship between the concept and the query is heavily dependent on the phrases used in the category graph structure. Hence, in few cases, the identified ′relationship′ phrase might loosely capture the "relationship" between two terms. As an example, when we try to determine a relationship between *Sparc* and *Sun Microsystems* (as shown in Table 3.1), R-LACE returns the relationship as *microprocessors*. This is because *microprocessors* is present as a category name in the category graph on Yahoo! Directories. However, the precise relationship is that *Sparc* is actually a "processor architecture" designed by *Sun Microsystems*. Hence, in this case, we were able to loosely capture the relationship, but we could not determine the exact relationship.

Chapter 4

Computation of the semantic relatedness between words

Determining the semantic relatedness between two words refers to computing a statistical measure of similarity between those words. Computation of similarity measures is useful in a wide range of applications such as natural language processing, query recommendation, relation extraction, spelling correction, document comparison and other information retrieval tasks. Although several methods that address this problem have been proposed in the past, effective computation of semantic relatedness still remains a challenging task. In this chapter, we propose a new technique for computing the relatedness between two words by using their concept clouds. In our approach, instead of computing the similarity between the two words directly, we propose to first compute the similarity measure between their generated concept clouds using web-based coefficients. Next, we map the obtained measure to determine the similarity between the original words. To generate the concept clouds, we make use of LACE. We perform an evaluation on the Miller-Charles benchmark dataset and obtain a high correlation coefficient of 0.882, which outperforms all other existing state of art methods, hence providing evidence for the high effectiveness of our method.

4.1 Background and motivation

Semantic metrics between words have been used by researchers to define *semantic relatedness, semantic similarity* and *semantic distance*, as described in [Gracia and Mena, 2008]. For completeness, we provide brief definitions of these concepts here. *Semantic relatedness* considers any type of relationship between two words (including hypernymy, hyponymy, synonymy and meronymy relationships, among others) and is usually a statistical similarity measure between the two words. *Semantic similarity* is a more specialized version of semantic relatedness that considers only synonymy and hypernymy relationships between words. *Semantic distance* is a distance-based measure of semantic relatedness. That is, the more related two words are, the smaller is the semantic distance between them.

Compared to machines, humans are able to accurately determine the similarity between two words based on their common sense knowledge. For example, in order to determine that the words <*apple*, *computer*> are more closely related than the words <*apple*, *car*>, humans would use their knowledge that *apple* is the name of a company that manufactures computer hardware and software, to determine that *apple* is semantically more related to *computer* than to *car*. Our goal is to provide machines with such power by using an automated *concept-based approach* to determine semantic relatedness.

In our approach we propose Q-LACE i.e *quantitative* LACE that computes the semantic relatedness between two words by computing the similarity between their concept clouds. To automatically generate concept clouds, we make use of LACE to generate a concept cloud for the query, by extracting its associated concepts from the web. Thus, for a given pair of words, we first extract the concepts associated with each word in the pair and generate their concepts clouds. We then compute a semantic relatedness between the two concept clouds and use it to determine the similarity between the initial word pair.

4.2 Related work

4.2.1 Determining semantic relatedness

The problem of determining the semantic relatedness between two words has been an area of interest to researchers from several areas for long time. Some very preliminary approaches [Rada et al., 1989] calculated the similarity between two words on the basis of the number of edges in the term hierarchy created by indexing of articles. Similar edge-counting based methods were also applied on existing knowledge repositories such as Roget's Thesaurus [Jarmasz and Szpakowicz, 2003] or WordNet [Hirst and St-Onge, 1998] to compute the semantic relatedness.

To improve the preliminary approaches to calculating the semantic relatedness between words, more sophisticated methods have been proposed. Instead on simply relying on the number of connecting edges, Leacock and Chodorow [1998] have proposed to take the depth of the term hierarchy into consideration. Others groups have proposed to use the description of words present in dictionaries [Lesk, 1986] and techniques such as LSA [Deerwester et al., 1990] to compute semantic relatedness. However, due to the very limited size of WordNet as a knowledge base and the absence of well known named entities (e.g *Harry Potter*) in WordNet, researchers have started to look for a more comprehensive knowledge base.

The advent of Wikipedia in 2001 has fulfilled the need for a more comprehensive knowledge base. Many techniques that used Wikipedia to compute semantic relatedness have been developed in the recent years. Among others, Strube and Ponzetto [2005] have used Wikipedia to determine semantic relatedness. Their results outperform those obtained using WordNet, hence showing the effectiveness of Wikipedia in determining the similarity between two words. Gabrilovich and Markovitch [2007] have developed a technique, called Explicit Semantic Analysis (ESA), to represent the meaning of words in a high dimensional space of concepts derived from Wikipedia. Experimental results show that ESA outperforms the method given by [Strube and Ponzetto, 2005]. Chernov et al. [2006] have suggested to make use of the links between categories present on Wikipedia to extract semantic informa-

tion. Milne and Witten [2008] have proposed the use of links between articles of Wikipedia rather than its categories to determine semantic relatedness between words. Zesch T. [2008] have proposed to use Wiktionary, a comprehensive wiki-based dictionary and thesaurus for computation of semantic relatedness. Although Wikipedia has proved to be a better knowledge base than WordNet, many terms (e.g., 1980 movies) are still unavailable on Wikipedia. This has motivated the use of the whole web as the knowledge base for calculating semantic relatedness.

Bollegala et al. [2007b] have proposed to use page counts and text snippets extracted from result pages of web searches to measure semantic relatedness between words. They achieve a high correlation measure of 0.83 on the Charles-Miller benchmark dataset. Sahami and Heilman [2006] have used a similar measure. Cilibrasi and Vitanyi [2007] have proposed to compute the semantic relatedness using the normalized google distance (NGD), in which they used Google^{TM} to determine how closely related two words are on the basis of their frequency of occurring together in web documents. Chen et al. [2006] have proposed to exploit the text snippets returned by a Web search engine as an important measure in computing the semantic similarity between two words.

The approach in [Salahli, 2009] is the closest to our approach, as it uses the related terms of two words to determine the semantic relatedness between the words. However, the major drawback of the approach proposed in [Salahli, 2009] is that the related terms are manually selected. As opposed to that, our approach automatically retrieves the most relevant terms to a given word query. Furthermore, Salahli [2009] compares the related terms to the original query. In our approach, we compute the semantic relatedness between two words using the semantic similarity between their generated concept clouds. To the best of our knowledge, such an approach has not been proposed yet.

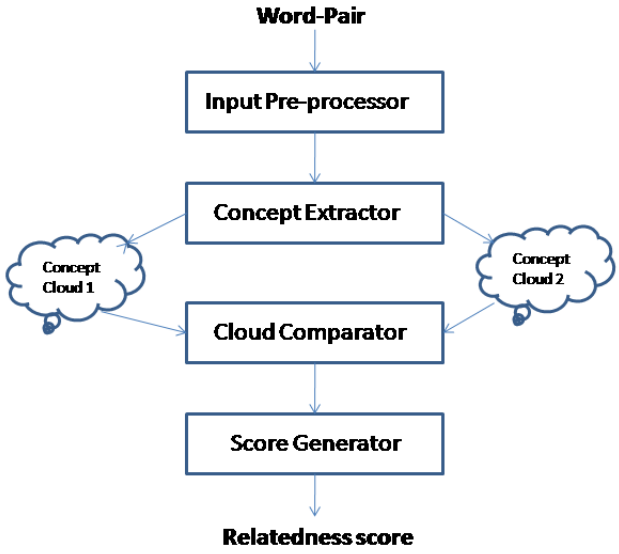


Figure 4.1: *System architecture*

4.3 System architecture

The complete system architecture for Q-LACE is shown in Figure 4.1. We use a two-phase approach to proceed with the computation of semantic relatedness between words. The first phase involves the use of LACE to extract concepts related to the given pair of words and to generate their concept clouds. In the second phase, we use web-based coefficients (Cosine, Jaccard, Dice, Overlap) to compute the semantic relatedness between the generated concept clouds, and use the resulting score to determine the relatedness between the original words. We will next describe LACE, followed by the description of the algorithm that determines the semantic relatedness between the generated concept clouds.

4.3.1 Input pre-processor

The *Input Pre-Processor* module takes the given word-pair as input and divides it into two separate words. Each word is then pre-processed, i.e., converted to lower case letters. Furthermore, any special characters such as "@", if present, are removed from the words. Finally, the two words are provided as input to the concept extractor (one at a time).

4.3.2 Cloud comparator

The function of the *Cloud Comparator* module is to perform a statistical comparison of the concept clouds corresponding to two words, using web based coefficients such as Dice and Jaccard coefficients. To achieve that, the *Cloud Comparator* computes a statistical similarity measure between all pairs of concepts $< A, B >$, where A belongs to the cloud of one term and B belongs to the cloud of the second term, and calculates the average of all similarity scores to determine the relatedness between the original words.

We will use the example in Figure 4.2 to illustrate the functionality of the *Cloud Comparator* module. We assume that we want to find the semantic relatedness between the words *automobile* and *car*. After executing the first two modules in our system, the concept

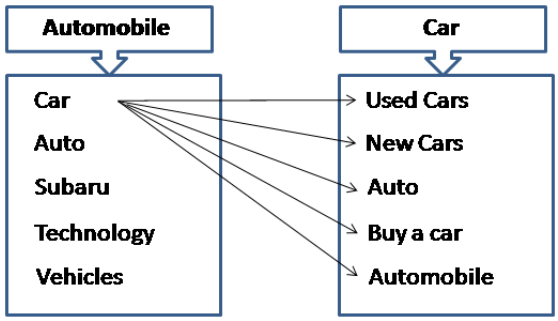


Figure 4.2: *An example illustrating the functionality of the Cloud comparator module.*

cloud for *automobile* is {*Car, Auto, Subaru, Technology, Vehicles*}, while the concept cloud for *car* is {*Used Cars, New Cars, Auto, Buy a Car, Automobile*}. The comparator takes each concept from the first cloud, e.g., *Car* and finds its similarity with concepts in the second cloud, using web-based coefficients. Preliminary experiments have shown that the Jaccard's coefficient produces the best results. Hence, we have used the Jaccard's coefficient to compute semantic relatedness between two concept clouds.

Traditionally, the Jaccard's coefficient is used to determine the similarity between two given sets, A and B, by taking the ratio between the size of the intersection of the two sets and the size of the union of the two sets. That is: $Jaccard(A, B) = \frac{|P \cap Q|}{|P \cup Q|}$.

However, Bollegala et al. [2007b] have modified the Jaccard's coefficient definition to make it possible to compute the similarity between two words, P and Q, using web search results. Thus:

$$Jaccard(P, Q) = \frac{H(P \cap Q)}{H(P) + H(Q) - H(P \cap Q)}.$$

where $H(P)$ and $H(Q)$ refer the number of pages retrieved when the query *"P"* and the query *"Q"* are posted to a search engine, respectively; and $H(P \cap Q)$ is the number of pages retrieved when the query *"P" "Q"* is posted to a search engine.

To compute the similarity between a term i from the first concept cloud A, denoted $con_A(i)$, and the second concept cloud B, we compute the Jaccard's coefficient between $con_A(i)$ and all concepts $con_B(j)$ in the concept cloud B and then take the average of all scores obtained. That is:

$$sim(con_A(i), cloud(B)) = \frac{\sum_{j=1}^{n} Jaccard(con_A(i), con_B(j))}{n},$$

where n is the total number of concepts in $cloud(B)$.

Consider the example in Figure 4.2. The similarity between the concept *car* and the cloud *car* is computed as:

$$sim(car, cloud(car)) = \frac{\sum_{j=1}^{5} Jaccard(car, con_{car}(j))}{5}.$$

We calculate this score for each concept in first cloud A and then pass on the array of scores to the *Score Generator* module.

4.3.3 Score generator

The *Score Generator* is the simplest module in our framework. It takes as input the array of scores received from the *Cloud Comparator* and computes the average of the scores to obtain a final score for the two initial words. That is,

$$score(A,B) = \frac{\sum_{i=1}^{m} sim(con_A(i), cloud(B))}{m}.$$

where m refers to the total number of concepts in $cloud(A)$. It can be seen that:

$$score(A,B) = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} Jaccard(con_A(i), con_B(j))}{m * n}.$$

The calculated score is reported to the user as the semantic relatedness score between the given words.

4.4 Experimental results and evaluation

4.4.1 Evaluation on Miller-Charles data

In the second phase of our evaluation, we are primarily interested in evaluating our system by comparison with other existing systems. We performed this evaluation of the Miller-Charles benchmark dataset, described below. Table 4.4.1 summarizes the results of the comparison. Some of the results shown in Table 4.4.1 have been obtained from [Bollegala et al., 2007b]. Please note that all scores, except for the Miller-Charles scores, have been scaled to [0,1] by dividing all scored by the maximum score (such that the best score becomes 1). The page count and the concept data collected is as of May 4th, 2009.

Miller-Charles dataset

The Miller-Charles dataset is a data set of 30 word pairs, which have been evaluated for semantic relatedness, on a scale of 0-4, by a group of 38 human subjects. However, most researchers have used only 28 word pairs, as two pairs are not available in WordNet. A scale of 0 implies no synonymy, while a scale of 4 implies perfect synonymy. The Miller-Charles dataset is considered as a reliable platform to evaluate measures of semantic relatedness.

Word Pair	Miller-Charles	Web Dice	Web Overlap	Sahami	CODC	Bollegala	Our approach
cord-smile	0.13	0.108	0.036	0.090	0	0	0.023
rooster-voyage	0.08	0.012	0.021	0.197	0	0.017	0.027
noon-string	0.08	0.133	0.060	0.082	0	0.018	0.034
glass-magician	0.11	0.124	0.408	0.143	0	0.180	0.027
monk-slave	0.55	0.191	0.067	0.095	0	0.375	0.029
coast-forest	0.42	0.870	0.310	0.248	0	0.405	0.078
monk-oracle	1.1	0.017	0.023	0.045	0	0.328	0.052
lad-wizard	0.42	0.077	0.070	0.149	0	0.220	0.012
forest-graveyard	0.84	0.072	0.246	0	0	0.547	0.062
food-rooster	0.89	0.013	0.425	0.075	0	0.060	0.121
coast-hill	0.87	0.965	0.279	0.293	0	0.874	0.010
car-journey	1.16	0.460	0.378	0.189	0.290	0.286	0.186
crane-implement	1.68	0.076	0.119	0.152	0	0.133	0.035
brother-lad	1.66	0.199	0.369	0.236	0.379	0.344	0.307
bird-crane	2.97	0.247	0.226	0.223	0	0.879	0.009
bird-cock	3.05	0.162	0.162	0.058	0.502	0.593	0.518
food-fruit	3.08	0.765	1	0.181	0.338	0.998	0.566
brother-monk	2.82	0.274	0.340	0.267	0.547	0.377	0.460
asylum-madhouse	3.61	0.025	0.102	0.212	0	0.773	0.849
furnace-stove	3.11	0.417	0.118	0.310	0.928	0.889	0.502
magician-wizard	3.5	0.309	0.383	0.233	0.671	1	0.493
journey-voyage	3.84	0.431	0.182	0.524	0.417	0.996	0.596
coast-shore	3.7	0.796	0.521	0.381	0.518	0.945	0.649
implement-tool	2.95	1	0.517	0.419	0.419	0.684	0.524
boy-lad	3.76	0.196	0.601	0.471	0	0.974	0.911
automobile-car	3.92	0.668	0.834	1	0.686	0.980	0.898
midday-noon	3.42	0.112	0.135	0.289	0.856	0.819	1.000
gem-jewel	3.84	0.309	0.094	0.211	1	0.686	0.884
Correlation	**1**	**0.267**	**0.382**	**0.579**	**0.693**	**0.834**	**0.882**

Table 4.1: *Semantic relatedness results obtained with our approach and several related methods, by comparison with the Miller-Charles scores. The results of the methods called Web Dice and Web Overlap [Bollegala et al., 2007b], Sahami [Sahami and Heilman, 2006], CODC [Chen et al., 2006] and Bollegala [Bollegala et al., 2007b] are obtained from [Bollegala et al., 2007b]. Pearson correlation coefficients between the scores obtained with each method and the Miller-Charles scores are also reported.*

Results

The results of our experiments on the Miller-Charles dataset are shown in Table 4.4.1 by comparison with the Miller-Charles scores and the results of several previous methods as

reported in [Bollegala et al., 2007b]. The correlations between the scores obtained with each method and the benchmark scores are also reported. As can be seen, our approach outperforms similar existing methods by achieving a high Pearson correlation coefficient of 0.882. The highest scoring pair was *midday-noon* while the lowest scoring pair was *bird-crane*. Moreover, another advantage of our approach is that it is not dependent on a single knowledge source such as WordNet or Wikipedia and hence, has the capability to determine semantic relatedness between almost any word-pair.

4.5 Discussion

In this chapter, we have proposed a method for computing the semantic relatedness between two given words. Our approach relies on LACE for finding related concepts based on web searches (i.e., concept clouds for the two words). We obtained a high correlation coefficient of 0.882 on benchmark dataset of Miller Charles dataset showing the high effectiveness of our approach.

The success of our approach can be explained as follows: *LACE* forms the basis for the proposed method. The *LACE* works by extracting concepts using the top links returned by a search engine. Hence, there is a high probability that these links are related to the most popular meaning of the posted query. For example, if the query "Apple" is posted to Yahoo!^{TM}, 28 links out of the top 30 links are related to the "company" Apple rather than the "fruit" apple. Therefore, the extracted concepts are related to the most popular meaning of the term. If we analyze the Miller-Charles dataset, we note that the word pairs that are related to the most popular meaning of the words get a higher rating by the human subjects. For example, between the word pairs *magician-wizard* and *glass-magician*, *magician-wizard* gets a higher rating by the human subjects (3.5) as compared to *glass-magician* (0.11). This is because *wizard* is a more popular meaning of the term *magician* as compared to *glass*. Our systems also assigns a higher score to *magician-wizard*, as the concepts extracted for *magician* are closer to concepts extracted for *wizard* than to concepts extracted for *glass*.

Chapter 5

Conclusion and Future Work

We live in an era where information holds a lot of importance. Thus, the task of retrieving "relevant" information from the huge pool of information has gained lot of attention. This has in fact fueled the success of web search engines such as Google, Yahoo, Altavista etc. They have helped the users in retrieving the most relevant information.

The information retrieval era began with introduction of algorithms such as Vector Space model that were primarily based on searching a set of documents on the basis of the keywords in the query. More sophisticated algorithms such as PageRank and HITS helped improve the quality of search results by taking into consideration the graph structure of the web. However, these algorithms still rely on the "keyword-based" search techniques and are able to return only those documents that contain the keywords given by the user in the query. However, the increasing needs of the user has made it necessary to also consider the associated semantics with the query. So, in order to provide the user with a better set of results, it has become mandatory to get a feel of what the user actually meant when he posted the query.

In this thesis, we deal with the this problem. We proposed a novel way of presenting the results to the user - the Concept Relationship graph (CRG). We proposed Link Analysis based Concept Extractor (LACE) that extracts associated concepts related to the query posted by the user. Hence, by using LACE, we confirm that we do not limit the search space to the words in the query but we also cover a larger concept space. We compared

our approach to some of the existing concept extraction algorithms and proved that it outperforms them by not only providing high quality results but also addressing some of the flaws in the existing approaches.

Moreover, after extraction of associated concepts related to the query, we also showed how each of these concepts are related to the original query. For this purpose, we proposed a novel "Relationship prediction" algorithm, R-LACE, that makes use of a collection of three well known knowledge bases, WordNet, Wikipedia and Yahoo! Web Directory to predict the relationship between given two words. We used WordNet and Wikipedia mainly to determine whether two words are synonyms of each other or not. The category graph structure of the web directories was used to determine the relationship between the two words if it is found that the two words are not synonyms of each other. Moreover, instead of just representing the results in a plain textual formats as a collection of links to the user, we proposed to present the results in the form of a Semantic Relationship Graph (SRG). Finally, we also showed how we can convert the generated relationships into formal languages such as RDF. The generated RDF can be given as an input to an inference engine to perform some semantic inferences. We compared the results of our approach with the results of some of the existing "state of art" approaches and proved the effectiveness of our approach.

Finally, we used LACE to determine a statistical measure of semantic relatedness between words. For a given pair of words, our task was to determine on a scale of 0-1, how similar the two words are. For this purpose, we generated the concept clouds of the two words using LACE and then calculated the semantic relatedness between the two words using the relatedness between the generated concept clouds. Web based coefficients such as Dice and Jaccard were used in order to determine the relatedness between words. The semantic relatedness of the concept clouds was used to implicitly determine the semantic relatedness between the original word pair. In order to evaluate our approach, we used the benchmark Miller-Charles dataset and obtained a high correlation coefficient of 0.882 which outperformed all other existing state of art approaches.

Hence, this work is mainly focused on opening new vistas to the world of information retrieval by changing the way web search engines work and present results to the user today and make the tough task of retrieving the relevant information more efficient.

Future Work

We shall now highlight a few directions in which the existing work can be extended:

Extraction of semantically related concepts:

As illustrated in Chapter 2, the proposed approach is based on the assumption that the meta data information present on a webpage captures the context of the webpage well, in most of the cases. However, additional natural language processing techniques can be incorporated to extract relevant keywords present in the text of the document. Moreover, additional experimentation can be done to optimize the existing parameterssuch as the optimal number of pages to be extracted for a given query and the optimal value of the threshold level. Finally, the approach can be employed in the areas of text categorization and text mining.

Predicting relationship between words

The proposed approach makes use of three knowledge bases namely, Wordnet, Wikipedia and Yahoo! Web directory in order to predict relationships between words. Using this approach, we are able to predict the relationships with a good accuracy. However, in order to further improve the accuracy, we plan to replace the three knowledge bases with a single knowldege base, Wikipedia and apply natural language processing techniques on the text present in the wikipedia document to gather inference rules. These inference rules can be used to determine the relationship between any two words.

Computation of semantic relatedness between words

With our approach to compute semantic relatedness using concept clouds, we obtained a high correlation coefficient of 0.882 on the benchmark Miller-Charles dataset. The Miller-Charles dataset is a collection of 28 word pairs rated by human subjects on a scale of 0-4. In order to further test the effectiveness of our approach, we plan to test it on a more comprehensive dataset. Moreover, we also plan to use this technique to perform ontology mapping. Also, we plan to extend this approach to calculate a similarity measure between two documents.

Bibliography

S Angeletou, M Sabou, L Specia, and E Motta. Bridging the gap between folksonomies and the semantic web. In *Workshop - European Semantic Web Conference*, 2007.

S. Banerjee and T. Pedersen. Extended gloss overlap as a measure of semantic relatedness. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-03)*, pages 805–810, 2003.

D. Bollegala, Y. Matsuo, and M. Ishizuka. Measuring semantic similarity between words using web search engines. In *Proceedings Int. WWW2007 Conf.*, 2007a.

D. Bollegala, Y. Matsuo, and M. Ishizuka. Measuring semantic similarity between words using web search engines. In *Proc. of WWW 2007*, 2007b.

C. Buckley, G. Salton, J. Allan, and A. Singhal. Automatic query expansion using smart: Trec 3. In *Proceedings of 3rd Text REtreival Conference*, pages 69–80, 1994.

D. Cai, X. He, J.R. Wen, and W.Y. Ma. Block level link analysis. In *Proceedings of the 27th International ACM SIGIR*, 2004.

A. J. Canas, A. Valerio, J. Lalinde-Pulido, M. Carvalho, and M. Arguedas. Using wordnet for word sense disambiguation to support concept map construction. In *Proceedings International Symposium on String Processing and Information Retrieval (SPIRE 2003)*, Manaus, Brazil, 2003.

H. Chen, M. Lin, and Y. Wei. Novel association measures using web search with double checking. In *Proceedings of the COLING/ACL 2006*, 2006.

S. Chernov, T. Iofciu, W. Nejdl, and X. Zhou. Extracting semantic relationships between

wikipedia categories. In *Proc. of SemWiki2006 Workshop, co-located with ESWC2006*, 2006.

R.L. Cilibrasi and P.M. Vitanyi. The google similarity distance. In *IEEE Transactions on Knowledge and Data Engineering*, pages 370–383, 2007.

S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and R. Hashman. Indexing by latent semantic indexing. In *Journal of the Amer. Soc. for Inf. Science*, 1990.

L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, S. Solan, G. Wolfman, and E. Ruppin. Placing search in context: The concept revisited. *ACM Transactions on Information Systems*, 20(1):116–131, 2002.

E. Gabrilovich and S. Markovitch. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-07)*, pages 1606–1611, 2007.

Gracia and Mena. Web-based measure of semantic relatedness. In *Proc. of the 9th Int. Conf. on Web Information Systems Engineering*, pages 136–150, 2008.

G. Hirst and D. St-Onge. Lexical chains as representations of context for the detection and correction of malapropisms. In *C. Fellbaum Ed.*, pages 305–332. MIT Press, 1998.

Andreas Hotho, Steffen Staab, and Gerd Stumme. Wordnet improves text document clustering. In *Proceedings of SIGIR 2003 Semantic Web Workshop*, 2003.

M. Jarmasz and S. Szpakowicz. Rogets thesaurus and semantic similarity. In *In Proceedings of RANLP-03*, pages 212–219, 2003.

J. Jiang and D. Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proceedings of the International Conference on Research in Computational Linguistics ROCLING X*, 1998.

C. Leacock and M. Chodorow. Combining local context and wordnet similarity for word sense identification. In *C. Fellbaum Ed.*, pages 265–283. MIT Press, 1998.

C. Lee, G. Lee, and S.J. Yun. Automatic wordnet mapping using word sense disambiguation. In *Proceedings of the 2000 Joint SIGDAT Conference*, 2000.

W. Leelapatra and P. Netisopakul. Improving query expansion using link analysis. In *Proceedings of the 5th International Conference on ECTI-CON*, 2008.

Michael Lesk. Automatic sense disambiguation using dictionaries. In *Proc. of the 5th Int. Conf. on Systems Documentation*, 1986.

D. Lin. Automatic retreival and clustering of similar words. In *Proceedings of the 17th COLING*, 1998a.

D. Lin. An information-theoretic denition of similarity. In *Proceedings of the 15th ICML-International Conference on Machine Learning*, pages 296–304, 1998b.

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 *Proceedings of 15th International World Wide Web Conference*, 2006a.

Y. Matsuo, T. Sakaki, K. Uchiyama, and M. Ishizuka. Graph-based word clustering using web search engine. In *Proceedings of EMNLP 2006*, 2006b.

R. Mihalcea, P. Tarau, and E. Figa. Pagerank on semantic networks with application to word sense disambiguation. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING 2004)*, Geneva, Switzerland, 2004.

P. Mika. Ontologies are us: A unified model of social networks and semantics. In *Proceedings of ISWC2005*, 2005.

D. Milne. Computing semantic relatedness using wikipedia link structure. In *Proceedings of the New Zealand Computer Science Research Student conference (NZCSRSC'07)*, Hamilton, New Zealand, 2007.

D. Milne and I.H. Witten. An effective, low-cost measure of semantic relatedness obtained from wikipedia links. In *Proc. of AAAI08 Workshop on Wikipedia and Artificial Intelligence*, Chicago,IL, 2008.

Enrico Minack, Wolf Siberski, and Gideon Zenz. Suits4rdf: Incremental query construction for the semantic web. In *International Semantic Web Conference - Posters and Demos*, volume 401 of *CEUR Workshop Proceedings*, 2008.

M. Mitra, A. Singhal, and C. Buckley. Improving automatic query expansion. In *Proceedings of 21st Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pages 206–214, 1998.

K Nakayama, T Hara, and S Nishio. Wikipedia link structure and text mining for semantic relation extraction. In *Proceedings of the Workshop on Semantic Search (SemSearch 2008) at the 5th European Semantic Web Conference (ESWC 2008) , June 2, 2008, Tenerife, Spain*, volume 334, 2008a.

K. Nakayama, T. Hara, and S. Nishio. Semsearch. In *Proceedings of the CEUR Workshop*, 2008b.

Rifat Ozcan and Y. Alp Aslandogan. Concept based information-access. In *Information Technology: Coding and Computing (ITCC)*, 2005.

R. Rada, H. Milli, E. Bicknell, and M. Blettner. Development and application of a metric to semantic nets. In *IEEE Transactions on Systems, Man and Cybernetics*, pages 17–30, 1989.

P. Resnik. Semantic similarity in a taxonomy: An information based measure and its application to problems of ambiguity in natural language. In *Journal of Aritificial Intelligence Research*, 1999.

R. Rosenfield. A maximum entropy approach to adaptive statistical modelling. In *Computer Speech and Language*, 1996.

M. Sahami and T. Heilman. A web-based kernel function for measuring the similarity of short text snippets. In *Proceedings of 15th International World Wide Web Conference*, 2006.

M. A. Salahli. An approach for measuring semantic relatedness via related terms. In *Mathematical and Comp. Applications*, volume 14, pages 55–63, 2009.

M. Strube and S.P Ponzetto. Exploiting semantic role labeling, wordnet and wikipedia for coreference resolution. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, NJ, USA, 2005.

Xia Sun, Qinghua Zheng, Haifeng Dang, Yunhua Hu, and Huixian Bai. An approach to acquire semantic relationships between words from web document. In *Lecture notes in Computer Science*, 2007.

Z. Syed, T. Finin, and A. Joshi. Wikipedia as an ontology for describing documents. In *Proceedings of 2nd Int. Conf. on Weblogs and Social Media*, 2008.

P. D. Turney. Minning the web for synonyms: Pmi-ir versus lsa on toefl. In *Proceedings of ECML-2001*, 2001.

B. Vlez, R. Wiess, M. Sheldon, and D. Giord. Fast and effective query refinement. In *Proceedings of 20th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pages 6–15, 1997.

Z. Wu and M. Palmer. Verb semantics and lexical selection. In *Proceedings of ACL-94*, 1994.

W Xu and Y Gong. Document clustering by concept factorization. In *Proceedings of the 27th annual international ACM SIGIR*, 2004.

Gurevych I. Zesch T., Mller Christof. Using wiktionary for computing semantic relatedness. In *Proceedings of AAAI 2008*, pages 861–868, 2008.