October 2011

Improving Search Engine Results by Query Extension and Categorization

Guo Mei
University of Western Ontario

Supervisor
Dr. Roberto Solis-Oba
The University of Western Ontario

Graduate Program in Computer Science

A thesis submitted in partial fulfillment of the requirements for the degree in Doctor of Philosophy

IMPROVING SEARCH ENGINE RESULTS BY QUERY EXTENSION AND CATEGORIZATION

(Thesis format: Monograph)

by

Guo Mei

Graduate Program in Department of Computer Science

A thesis submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

The School of Graduate and Postdoctoral Studies

The University of Western Ontario

London, Ontario, Canada

THE UNIVERSITY OF WESTERN ONTARIO

School of Graduate and Postdoctoral Studies

CERTIFICATE OF EXAMINATION

Supervisor:

.....
Dr. Roberto Solis-Oba

Supervisory Committee:

.....
Dr. Hanan Lutfiyya

.....
Dr. Sheng Yu

Examiners:

.....
Dr. Bin Ma

.....
Dr. Olga Veksler

.....
Dr. Sheng Yu

.....
Dr. Xingfu Zou

The thesis by

Guo Mei

entitled:

Improving Search Engine Results by Query Extension and Categorization

is accepted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

.....
Date

.....
Chair of the Thesis Examination Board

Acknowledgments

Special acknowledgment to my advisor Dr. Roberto Solis-Oba, who first sparked my interest in Computer Science. I still remember the very first day I felt the beauty of algorithms when I was in his lecture. It has been an honor to be his M.Sc. and Ph.D. student. His insight, sense of humor and understanding has been a great help in overcoming many of the difficulties throughout my master's and doctoral program.

My family deserves a warm and special acknowledgment for the love and care. Without their support, I could not have done it. This degree is for you, Mom and Dad.

I thank all the faculty and staff in the Department of Computer Science of the University of Western Ontario; I have learned a lot from being in a lecture, being a TA or attending seminars.

Abstract

Since its emergence, the Internet has changed the way in which information is distributed and it has strongly influenced how people communicate. Nowadays, Web search engines are widely used to locate information on the Web, and online social networks have become pervasive platforms of communication.

Retrieving relevant Web pages in response to a query is not an easy task for Web search engines due to the enormous corpus of data that the Web stores and the inherent ambiguity of search queries. We present two approaches to improve the effectiveness of Web search engines. The first approach allows us to retrieve more Web pages relevant to a user's query by extending the query to include synonyms and other variations. The second, gives us the ability to retrieve Web pages that more precisely reflect the user's intentions by filtering out those pages which are not related to the user-specified interests.

Discovering communities in online social networks (OSNs) has attracted much attention in recent years. We introduce the concept of *subject-driven communities* and propose to discover such communities by modeling a community using a posting/commenting *interaction graph* which is relevant to a given subject of interest, and then applying link analysis on the interaction graph to locate the core members of a community.

Keywords: Web Search, Query Extension, Query Categorization, Online Social Network, Community Mining, Text Categorization, Link Analysis

Contents

List of Figures

List of Tables

List of Appendices

Chapter 1

Introduction

The origins of the Internet can be traced back to the 1960s [29], and its commercialization in the 1990s resulted in its unprecedented popularity and incorporation into virtually every aspect of modern human life. Today, the Internet, so-called *the network of networks*, carries a vast range of information and provides a multitude of services. It has become one of the largest (if not the largest) information depositories and communication platforms in the world. For example, in December 2010, the Internet became the largest source of news for Americans [13], surpassing traditional media like TV, radio and newspaper.

Since its release in 1991 [73], the size of the World Wide Web (WWW, or the Web)[1] has been expanding exponentially each year. An estimation of the number of the Web pages cached by Web search engines is about 17 billion [55] as of July 12th, 2011, and the number of Web pages that can be accessed by search engines only represents a small fraction of the whole Web, according to [49]. How to locate information about a particular topic within the enormous number of existing Web pages poses great challenges to information retrieval (IR). Web directories and search engines emerged to cope with this problem.

[1]WWW is a system designed for accessing Web documents via the Internet; it defines a set of protocols through which these documents are interconnected. Each Web document has a Uniform Resource Locator (URL) that specifies the location of the document. People can jump from document to document by following the URLs. The URLs in a Web document are call *hyperlinks*, and documents with hyperlinks are said to be written with *hypertext*, the counterpart of plain text in which there are no hyperlinks.

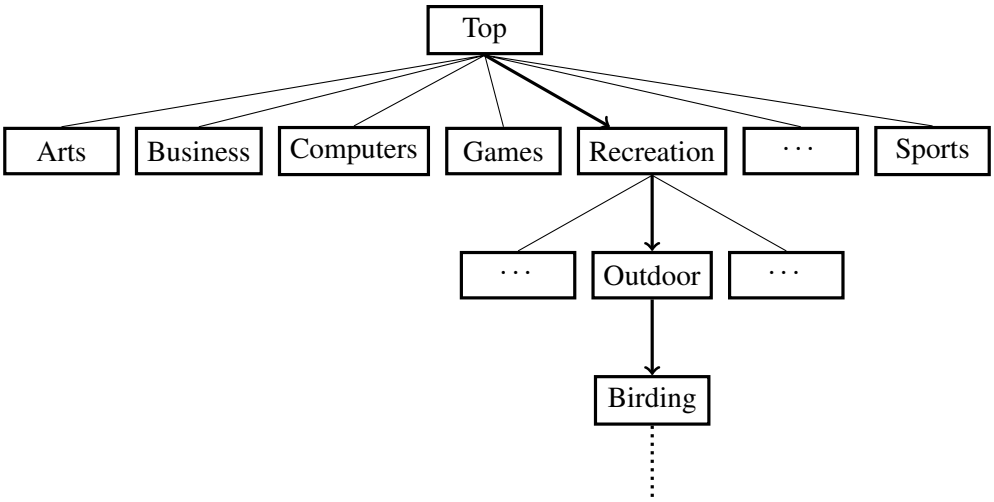


Figure 1.1: The tree-like structure of a Web directory and the path from the top level to a subcategory.

A Web directory is a directory of websites[2], which organizes the websites by categories, typically in a hierarchical way, which has a tree-like structure. Given a topic, a user can browse the directory starting at the top level, finding the appropriate category at each level all the way down to that subcategory which is most relevant to the topic, and browse websites listed in this subcategory to find the desired information. Figure 1.1 illustrates the tree-like structure of a Web directory, and the path from the top level to the "Birding" category.

Web search engines work in a different way from Web directories. A typical Web search engine consists of three components: A crawler (also named a spider or a robot), which collects Web pages from the WWW by following hyperlinks connecting Web pages; an indexer, which is essentially a database, optimized for searching, that stores locally the Web pages collected by the crawler[3]; and a query server, which returns appropriate Web pages matching a user's query[4], usually in most-relevant-first order.

It is proven by the success of Google [35], the largest Web search engine to date, that mod-

[2]A website is a collection of related web pages, which is usually managed by one entity, for example an organization, a company or an individual.

[3]The storing of Web pages is also called caching; the process of caching Web pages and optimizing the database for searching is called indexing.

[4]A query is a list of words describing the information the user is interested in.

ern search engines have been much more popular than Web directories. Web search engines have several key advantages over Web directories: 1) Web search engines store many more Web pages than Web directories do, because search engines can crawl Web pages automatically; 2) Web search engines respond to a user's query almost immediately, while it takes time to browse Web directories; 3) The invention of link analysis algorithms like the PageRank algorithm [11], boosted the effectiveness of search engines.

As we shall show in Section 1.1, Web search engines merely focus on the query words instead of the users' intention and that sometimes make search engines return results that are not interesting to the user. Even though automatic Web page categorization does not work well on the whole corpus of the WWW, its underneath techniques, especially data mining techniques, can be employed to refine a Web search engines' results and improve its effectiveness. In this thesis, we propose *Query Extension* to make the results returned by a search engine not limited by the query words, and *Query Categorization* which adds text classification technologies into Web search to present only the results more relevant to the users.

With the widespread availability and easy access of the Internet, communication services such as electronic mail (email) services, bulletin board systems (BBS), instant messaging services, blog[5] services and online social network (OSN) services, have enabled entirely new forms of social interaction. Here we refer to the OSN service as a service, platform, or website, utilizing the Internet and the WWW, that focuses on building and reflecting the social relations among people who share interests and activities. In the past few years OSNs have gained significant popularity; for example, Facebook [24], an online social network website, was the most popular website in the US in 2010 [20]. The scale (in terms of both geographical span and size) and the complexity of OSNs have attracted a large amount of attention from researchers, including computer scientists and sociologists, who try to understand the social relations among members of an OSN, and the underneath structure of these relations. This research area is usually called *online social network analysis*, and it includes exploring the

[5]Initially blog was the short term of "weblog", but then it became more widely accepted than the original term [7].

structure of OSNs, studying the relationships among users and their exchange of messages, predicting the friendship between members, tracing the evolution of OSNs and mining communities in OSNs.

Because of the popularity and size of OSNs, it is impossible for a member to interact with every other member in an OSN. Usually a member only interacts with a few groups of members; these groups can be considered as communities. Mining communities in OSNs often reveals interesting relationships among the members, such as common hobbies, social functions, occupations, etc, and gives insight into the structure and organization of the different sectors of an online society. Since the friendship network is a natural and intuitive representation of an OSN, many research works focus on identifying communities in such a network. However, the friendship network is based on only one kind of social relations and so it cannot represent all interactions among OSN members. We introduce a new approach to identify communities active on a given subject that uses a more general model than the friendship network.

At first sight, the challenges of information retrieval in the WWW and online social networks do not seem related, but the approaches used to deal with them are interconnected, and usually involve technologies from semantic analysis, data mining and network analysis. For example, the World Wide Web and OSNs can be modeled by graphs and analyzed using tools from graph theory; lexical analysis and text classification can be applied on Web pages or posts/comments in an OSN to understand their contents. In this thesis, we combine these technologies to address two very important problems: How to improve the effectiveness of search engines and how to identify communities in OSNs.

This thesis is organized in the following way. In the remaining of this chapter, we briefly introduce the basic notions of information retrieval, Web search, text and hypertext classification, and social network analysis. In Chapter 2, we introduce Query Extension and Query Categorization designed to improve the effectiveness (in terms of completeness and accuracy) of search engines. We also describe our implementation of these approaches, and our experimental results. In Chapter 3, we introduce the interaction graph model to identify subject-driven

communities in OSNs and present our experimental results. Finally in Chapter 4 we give a summary of our work.

1.1 Web Directories and Web Search Engines

Looking back at the history of the WWW, we notice that the development of information retrieval techniques for the Internet was a competition between Web directories and Web search engines. However, Web directories and search engines were not developed independently; they evolved together by trying to avoid the shortcomings of each other.

After the World Wide Web went public in 1991, researchers started to look for methods to efficiently access information in the WWW. Both, Web directories and search engines, were implemented when W3 Catalog, World Wide Web Wanderer, Aliweb and JumpStation were released. W3 Catalog, which crawled specialized catalogs (maintained by hand) and built up a searchable list, can be considered as a mixture of a Web directory and a search engine. World Wide Web Wanderer was the first Web crawler, whose main purpose was to measure the size of the WWW. Aliweb could return search results to the user upon receiving queries, but it did not crawl the WWW. The Web pages it cached were submitted by other people. JumpStation was the first Web search engine which implemented the three essential components of a Web search engine (a crawler, an indexer, and a query server). In the early days, Web Search engines indexed only the URL and the header[6] of a page. The first search engine which indexed the full text of Web pages was WebCrawler, released in 1994, and it opened the door for many other applications; to name a few, within one year of its debut came Lycos [53], Infoseek, AltaVista [2], and Excite [23].

Web search engines crawl the WWW and index the Web pages they encounter. Typically, a search engine extracts each word from a Web page and maintains a list of words for all pages it caches. When a user sends a query, the search engine looks for the words in the query, and

[6] A portion of a Web document defined in the Hyper Text Markup Language (HTML) standard that usually contains descriptive information about the Web page, for example, the title, version, encoding, etc.

returns the URLs of Web pages matching the query. The order of the returned URLs is usually most-relevant-first: Given a query, both types of Web search engines, those which index URLs and headers and those which index the full text, rank Web pages based on the occurrence of the query words; we call these search engines *text-indexing* search engines. Dividing a search engine in three parts, a crawler which collects Web pages, an indexer which processes the pages and a query server which takes care of the interaction with users, makes Web search engines convenient to use: The response is fast, and the order of the returned URLs according to the occurrences of the query seems intuitive. Web search engines soon became the main source for people to access information in the WWW. In 1996, Excite became the exclusive search service provider for big companies like Apple, Microsoft and Netscape.

The convenience of Web search engines attracted not only general users, but also *spammers*, who are, in the context of Web search, individuals or companies who put deceiving text on their Web pages to abuse a search engine's ranking system. For example, one can put an excessive repetition of popular query words like "car", "NBA" or "rose cup" on a pornography page; when a search engine indexes this page, it sees a number of these words and thus assumes that this page is highly relevant to these queries. When a user sends one of these query words to this search engine, the spam Web page is returned, even though it is not related to the query at all.

Web directories came shortly after the first search engines were released. Yahoo! Directory [81] was born in 1994 and ODP (named Gnuhoo at that time) [64] in 1998. Both Yahoo! Directory and ODP are manually compiled hierarchical directories. Within three years Yahoo! became the largest search and navigational service provider in the United States [77]. One of the reasons behind the early success of Yahoo! is that Web search engines suffered from spamming, to which Yahoo! Directory was immune, and the size of the WWW was not unmanageable at that time for the use of manual methods[7].

With the exponential expansion and continuous evolution of the WWW, manually main-

[7] Yahoo! Directory had 730,000 entries in 1997.

taining categories showed several drawbacks: 1) too much skilled manpower was required; 2) difficult to keep a directory up to date; 3) highly subjective decisions could not guarantee the precision of the directory [4]. Researchers have proposed different approaches to categorize Web pages automatically [4, 5, 15, 16, 28]. However, the scalability of these algorithms is still an open question and because of that the experiments based on these algorithms are limited to merely small collections of Web pages [80]. To the best of our knowledge, none of the existing commercial Web directories is built up using automatic categorization techniques.

To cope with spamming, new approaches of searching with hyperlinks emerged in 1998, when the HITS algorithm [43] and the PageRank algorithm [11] were published. The latter algorithm is now the essential technique on which Google, the most popular search engine to date, is based. The success of Google forced Yahoo! to change its search service from Yahoo! Directory to a Web search engine, namely Yahoo! Search [83]. As of May 2011, Google, Yahoo! Search and Bing [6] held more than 95% of the shares in the search engine market, according to a report [34] from the Search Engine Watch [70].

Since the HITS algorithm and PageRank algorithms gave information retrieval a new spin that dramatically increased their effectiveness, in the next two sections, we give a brief introduction to these algorithms.

1.1.1 HITS Algorithm

Based on the observation that the number of Web pages returned by text-indexing search engines was too large for a human user to digest, Kleinberg proposed the HITS (Hyperlink-Induced Topic Search) algorithm [43] to locate the most authoritative pages relevant to a given query by analyzing the hyperlinks among Web pages. Kleinberg claimed that there are two types of pages: *Authorities* and *hubs*. Authorities are Web pages which are highly relevant to a query, and hubs are Web pages which link to many authorities, as illustrated in Figure 1.2. A directed graph is used to model the WWW. In the graph, a node represents a Web page, and a link from node h to node a denotes the hyperlink in page h pointing to page a. In broad terms

the HITS algorithm works as follows:

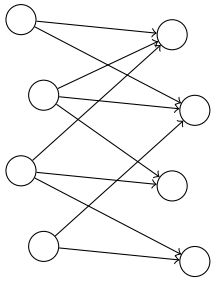


Figure 1.2: Authorities and hubs.

1. Starting from a user-supplied query, HITS assembles a *root set* of pages: A relatively small set of pages returned by any search engine on the query. It then expands this set to a larger *base set* by adding any pages that point to, or are pointed to by hyperlinks contained in any page in the root set. We depict the relationship between the root set and the base set in Figure 1.3.

2. Associate with each page p a *hub weight* $h(p)$ and an *authority weight* $a(p)$, all initialized to 1. Let $p \to q$ denote "p has a hyperlink pointing to page q". HITS iteratively updates the hub and authority weights as follows:

$$a(p) = \sum_{q \to p} h(q) \tag{1.1}$$

and

$$h(p) = \sum_{p \to q} a(p). \tag{1.2}$$

Thus, a single iteration replaces $a(p)$ by the sum of the hub weights of pages pointing to p, and replaces $h(p)$ by the sum of the authority weights of pages pointed to by p.

3. Repeat Step 2 until convergence is achieved.

It can be proven that this iterative process converges to a stable set of authority and hub weights after a polynomial number of rounds of Steps (1.1) and (1.2) [43]. HITS algorithm returns h and a as the final hub weight and authority weight of each page.

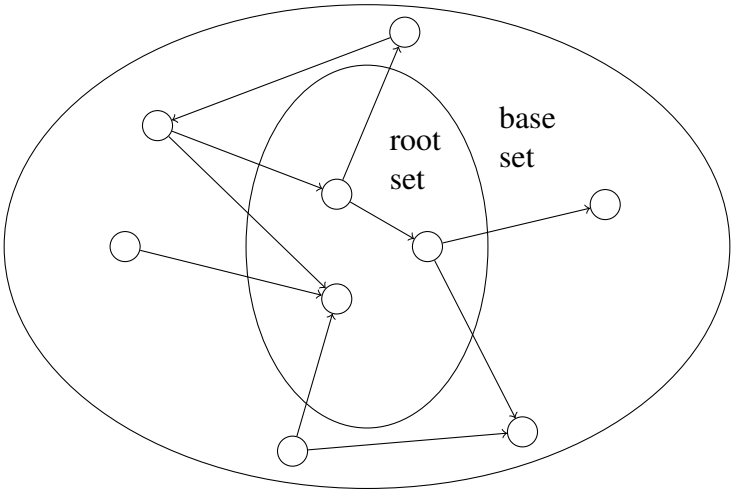


Figure 1.3: Expanding the root set to a base set.

Hubs and authorities exhibit what could be called a *mutually reinforcing relationship*: A good hub is a page that points to many good authorities and a good authority is a page that is pointed by many good hubs. Thus, HITS can be applied to various domains where the mutually reinforcing relationship holds. For instance, an approach to categorize Web pages using HITS algorithm was proposed in [14]. On the other hand, the HITS algorithm tends to assign high scores to the nodes which reside in well connected subgraphs; these clusters in the graph represent "communities" in the domain the graph models. For example, the HITS algorithm can be applied to find important individuals in an online social network [41].

Although it set a new direction to locate information in the WWW by analyzing hyperlinks, the HITS algorithm has some drawbacks which do not always allow it to properly deal with the complexity of the WWW and the enormous diversity of existing Web pages. For example, some of the links in a page are intended to navigate a website, and some others are advertisements; HITS cannot distinguish these special purpose hyperlinks, which might pollute the computation of hub and authority weights. Another drawback is the *topic drift* problem. It

is noticed in [43] that the HITS algorithm returns authoritative pages dealing with "broader" topics in the case of very specified query. It is also observed that for some queries, if the base set contains a cluster of tightly connected Web pages, the output of HITS tends to drift towards that cluster [5]. A study [10] has concluded that none of the common link-based analysis algorithms are immune to the topic drift problem. It is worth to mention that numerous methods to minimize the topic drift problem have been proposed, which usually involve content analysis of Web pages [5, 14, 15, 16].

1.1.2 PageRank Algorithm

Sergey Brin and Lawrence Page published their PageRank algorithm in 1998 [11]. *PageRank*, just like the *authority weight* in HITS, is a measure of the authority of a page based on the hyperlinks between Web pages. Note that there is no notion of *hub weight* in the PageRank algorithm. The authority of a Web page p depends on the number of incoming hyperlinks and on the authority of the pages q which cite p through a link. The *PageRank* of a Web page p is formally defined by

$$PR(p) = d \sum_{q \to p} \frac{PR(q)}{h(q)} + (1 - d), \tag{1.3}$$

where $d \in (0, 1)$ is a *damping factor*[8], $h(q)$ is the number of hyperlinks in document q, $q \to p$ denotes "q has a hyperlink to page p". It can be shown that Equation (1.3) has a unique (up to normalization) solution which can be computed efficiently [11].

The mutually reinforcing relationship of authorities and hubs of HITS is also valid when computing PageRank: Good Web pages are known by other good pages. Not like the text-indexing, in which the rank of a Web page is decided by the occurrence of the query, the PageRank of a page is decided by surrounding pages, therefore it is more objective and accurate. The PageRank algorithm directly inspired the foundation of Google. Google collects and indexes Web pages as text-indexing search engines do, but it also computes the PageRank of

[8] The probability that a user follows links on a Web page to browse other pages; usually set to 0.85.

each page with respect to the whole Web. When a user sends a query to Google, Google returns the pages containing the query, in the order of highest-PageRank-first, i.e., most-authoritative-page-first; this is what makes Google stand out among all Web search engines and this is the reason why Google has been the largest search service provider since 2000 [29].

From the point of view that both of them use hyperlinks to calculate the importance of Web pages, HITS and PageRank look very similar. But they differ from each other in some essential ways. For instance, HITS is topic-related while PageRank is topic-free. In the HITS algorithm the authority and hub weights $a(p)$ and $h(p)$ have to be computed for each query, but the PageRank $PR(p)$ needs to be computed only once, as long as the WWW remains the same.

1.2 Text Classification

Data mining approaches have been applied to text classification for about 20 years. In these approaches, a general inductive process (also called the *learner*) automatically builds a classifier for a given category c_i by extracting the characteristics of a set of documents that have been manually labeled by an expert as either belonging to category c_i or not belonging to c_i. This set of documents is called a *training set* and the process of building the classifier is called *training* or *learning*. From training, the inductive process determines the characteristics that a new unseen document must have in order to be classified under c_i. In data mining terminology, the classification problem is an activity of *supervised learning*, since the learning process is "supervised" by the knowledge of the categories contained in the training samples provided by a human expert.

The general procedure of text classification includes the following steps:

1. Gathering the training set. A set of samples are gathered and labeled.

2. Choosing a representation for the training set. In text classification, the vector space model is widely used; this model will be introduced in the next section.

3. Determining the learning method. Well known methods include, to name a few, Naïve Bayes, Support Vector Machine (SVM), Decision Tree, K-Nearest Neighborhood (KNN), and Neural Networks.

4. Training the classifier and building the model using the learning method on the training set.

5. Classifying unseen data by the model built in the previous step.

In this section, we describe two kinds of classifiers, Naïve Bayes and Support Vector Machine(SVM), which are among the most widely employed data mining classifiers. Before we start, we first introduce a text model on which those approaches can be applied.

1.2.1 Vector Space Model

To make text suitable for recognition by a classifier, a *modeling* or *indexing* procedure which maps a text document d_j into a compact representation of its content is required. In information retrieval, the *vector space model* is widely used for this purpose [27]. Documents are split into words by using simple syntactic rules, and words are usually transformed to some canonical form (e.g., "reading" is transformed to "read"). Each canonical word, which is essentially a feature of a document, represents a dimension in a Euclidean space, and documents are vectors in this space: $\vec{d_j} = \langle w_{1j} \cdots w_{|\tau|j} \rangle$, where w_{ij} is the weight of the ith feature in the jth document and, τ is the set of features. Different approaches have been explored to define the feature set τ and the weights of features w_{ij}. For example, in the simplest approaches, called the *set of words model* and the *bag of words model*, features are words and the weight of a feature is either a binary number 0 or 1 (representing the presence or absence of the word), or the number of times a word occurs in the document d_j, respectively [67, 68].

The intuitions that 1) the more often a feature occurs in a document, the more it is representative of the document, and 2) the more documents a feature occurs in, the less discriminating it is, lead to one of the most frequently used weighting scheme, the `tfidf`, or Term Frequency

Inverse Document Frequency, defined as:

$$\texttt{tfidf}(t_k, d_j) = \#(t_k, d_j) \cdot \log \frac{|T_r|}{\#_{T_r}(t_k)} \tag{1.4}$$

where T_r is the set of documents, $\#(t_k, d_j)$ denotes the term frequency, i.e. the number of times a term t_k occurs in d_j , and $\#_{T_r}(t_k)$ denotes the document frequency, i.e., the number of documents in T_r in which a term t_k occurs. The `tfidf` is usually normalized in order for the weights to fall in the interval [0, 1].

The usefulness of `tfidf` can be shown by an example. Given the query " the black pearl", a search engine will find millions of pages containing all three words in its database; but which is more relevant to the query, a page in which "the" occurs 100 times while "black" and "pearl" occur only once, or a page in which each of "the" , "pearl" and "black" occurs 20 times? Intuitively we know that the correct answer is the latter one, because "the" is so common in English and "black" and "pearl" are more meaningful and occur more frequently in the second page; however, simply counting the number of occurrences of the query words suggests the opposite. In fact, the `tfidf` of "the" is low because $\#_{T_r}(t_k)$ in Equation (1.4) is usually large. If the `tfidf`s of the three words, instead of the number of their occurrences, are added up to calculate the ranks of pages, a more desirable rank will be achieved. Thus, `tfidf` can be used to measure the relevance of a document to a query, and it is commonly employed by text-indexing search engines.

Because Equation (1.4) assigns low `tfidf`s to those words which are commonly used in all documents like "the", "are" or "in", it is reasonable not to include the words with low `tfidf` in the feature vectors. This process is known as *feature selection*.

Considering the facts that 1) the size of the vocabulary in a collection of documents is usually huge and thus the dimension of the feature vector is too large to be processed efficiently by classifiers, 2) many words are *stopwords*, or noise words, e.g., prepositions, conjunctions, which do not add any statistic significance but have semantic meaning as per the nature of the

language, feature selection is usually adopted to remove those non-informative features and reduce the dimension of the feature vector [79]. We have shown that `tfidf` can be applied for the purpose of feature selection; actually it is one of the most popular feature selection measures. In text classification, other widely adopted feature selection measures include *information gain* and *mutual information*: Information gain measures how much information is obtained for category prediction by knowing the presence or absence of a term in a document, while mutual information measures the mutual dependence of a term t and a category c.

Given a set of training documents, for each unique word the `tfidf`, information gain or mutual information is computed, and the terms whose `tfidf`/information gain/mutual information are less than some predetermined threshold are removed from the feature space. This procedure of feature selection can discard up to 90% of the training set vocabulary [78].

1.2.2 Naïve Bayes Classifier

According to Bayes Theorem, the probability that a document represented by a vector of terms $\vec{d_j} = \langle w_{1j}, \cdots, w_{|\tau|j} \rangle$ belongs to category c_i is given by

$$Pr(c_i|\vec{d_j}) = \frac{Pr(c_i)Pr(\vec{d_j}|c_i)}{Pr(\vec{d_j})}. \tag{1.5}$$

The estimation of $Pr(\vec{d_j}|c_i)$ is not easy since it is impossible to compute the probability of $\vec{d_j}$ given c_i by considering only the training documents in c_i, as $\vec{d_j}$ is not in c_i. A common strategy used to estimate this value is to assume that any two components (w_{kj}) of the vectors d_j are statistically independent of each other; thus, the distribution of vectors $\vec{d_j}$ given c_i can be decomposed as follows:

$$Pr(\vec{d_j}|c_i) = \prod_{k=1}^{|\tau|} Pr(w_{kj}|c_i) \tag{1.6}$$

and $Pr(w_{kj}|c_i)$ can be estimated as follows, by considering the documents in the training set:

$$Pr(w_{kj}|c_i) = \frac{\text{\# of terms } w_{kj} \text{ in } c_i}{\text{total \# of all terms in } c_i} \tag{1.7}$$

Probabilistic classifiers which are based on the aforementioned "naïve" assumption that features of a document are independent are called Naïve Bayes classifiers. In the training phase, we compute the probability of each distinct term w_k (the kth feature in the feature set τ) in c_i using Formula 1.7, and compute $Pr(c_i)$ as the ratio of the number of documents in c_i over the total number of documents in the training set. Then in the testing/classifying phase, given a unseen document $\vec{d} = \langle w_1, \cdots, w_{|\tau|} \rangle$, we can compute $Pr(\vec{d}|c_i)$ for each category c_i using Formula 1.6, and then output that category c which makes $Pr(c)Pr(\vec{d}|c)$ maximum, i.e.,

$$c = argmax_{c_i}\ Pr(c_i)Pr(\vec{d}|c_i) = argmax_{c_i}\ Pr(c_i) \prod_{k=1}^{|\tau|} Pr(w_k|c_i). \tag{1.8}$$

Note that we do not need to worry about $Pr(\vec{d_j})$ in Equation (1.5) because 1) it is constant, given a fixed set of documents, and 2) we care only about which c_i makes Equation (1.5) maximum, and not what the value of Equation (1.5) is.

The assumption that the features are independent from each other does not seem to hold for text documents in the real world, but despite this Naïve Bayes classifiers perform surprisingly well in practice, and still remain one of the most important classifiers.

1.2.3 Support Vector Machine

Given a set of vectors in a Euclidean space representing a collection of documents, each of which belongs to one of two classes, a hyperplane[9] which separates the vectors belonging to each class in such a way that the separation, or *margin*, between the two classes is maximum, is called the *maximum-margin hyperplane*. The vectors closest to the hyperplane are called *support vectors*. The term *Support Vector Machine* refers to those classifiers that build the hyperplane model to separate classes and to classify unseen data [19].

Formally, given a training set $\mathcal{D} = \{(\vec{x_i}, y_i)\}_{i=1}^{|\tau|}$ with respect to a category c where $\vec{x_i}$ is a vector representing a document, $y_i = 1$ if $\vec{x_i} \in c$ and $y_i = -1$ if $\vec{x_i} \notin c$, any hyperplane can be

[9] A hyperplane of an n-dimensional space is a subset of the space with dimension $n - 1$; it separates the space into two half spaces.

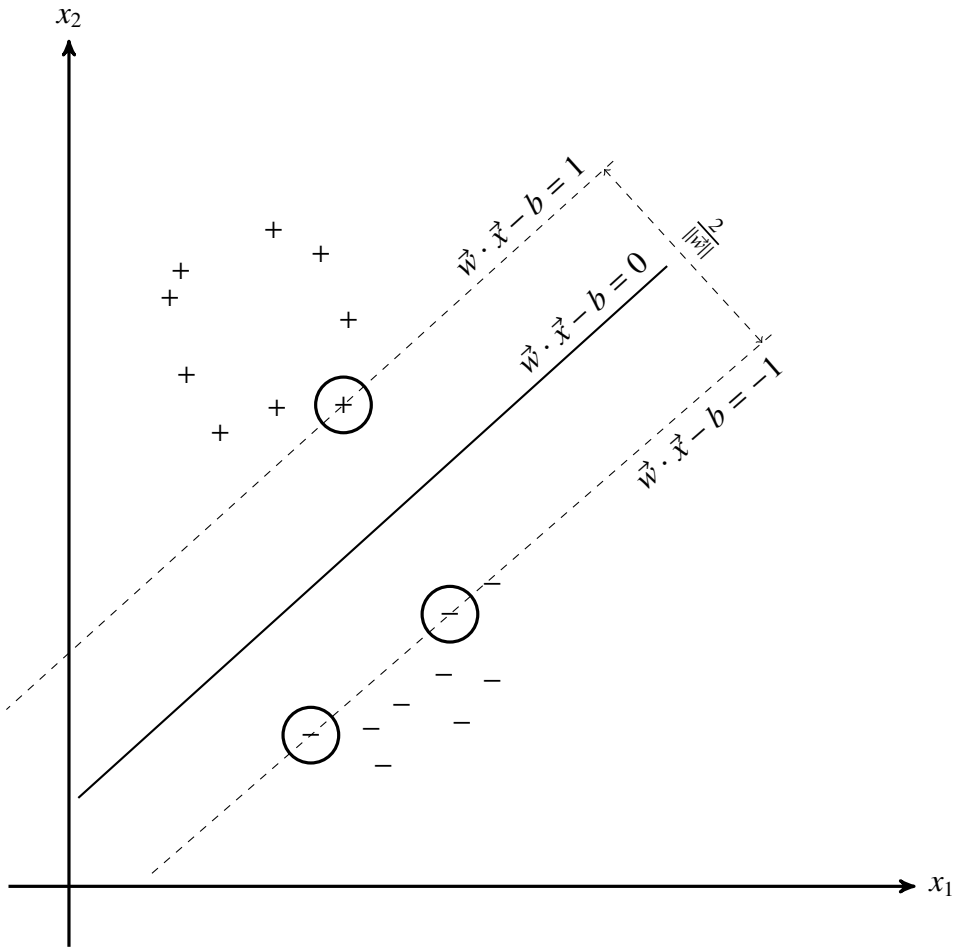


Figure 1.4: Hyperplane and the support vectors.

defined as the set or vectors $\vec{x}$ that satisfies $\vec{w} \cdot \vec{x} - b = 0$, where $\vec{w}$ is called the normal vector of the hyperplane, and the parameter $\frac{b}{\|\vec{w}\|}$ determines the offset of the hyperplane from the origin along the normal vector $\vec{w}$. We want to choose $\vec{w}$ and b to maximize the margin, or distance between the hyperplane and the support vectors. The distance between the two hyperplanes passing through the support vectors is $\frac{2}{\|\vec{w}\|}$, so we want to minimize $\|\vec{w}\|$, subject to

$$\vec{w} \cdot \vec{x_i} - b \geq 1; \; if \; y_i = 1 \tag{1.9}$$

and

$$\vec{w} \cdot \vec{x_i} - b \leq -1; \; if \; y_i = -1. \tag{1.10}$$

The last two constraints can be combined into:

$$y_i(\vec{w} \cdot \vec{x_i} - b) \geq 1 \tag{1.11}$$

for any $i = 1, \cdots, |\tau|$.

This is a quadratic programming problem, and can be solved by using mathematical techniques. Figure 1.4 illustrates a hyperplane and its support vectors: Two classes of vectors are marked by "+" and "−". The maximum-margin hyperplane satisfies $\vec{w} \cdot \vec{x} - b = 0$ and maximizes the margin $\frac{2}{\|\vec{w}\|}$; support vectors are marked with circles and satisfies $y_i(\vec{w} \cdot \vec{x_i}) - b = 1$.

Theoretical and empirical evidence show that one remarkable property of SVMs is that their ability to learn is independent of the dimensionality of the feature space [42]. SVMs choose a hyperplane based on the margin which separates the vectors, not the number of features. This property makes SVMs suitable for text classification, because the dimension of feature vectors is usually high, and so feature selection is not necessary.

1.2.4 Training Set

In data mining and text classification, it is well known that the distribution of the labeled samples in the training set has a huge impact on the performance of the classifiers. For example, when the learning process is based on an imbalanced data set[10], the characteristics of the majority class can overwhelm those of the minority class, and the probability of misclassifying an unseen document, which actually belongs to the minority class, as belonging to the majority class is high [1, 40, 54, 87]. One of the common approaches to deal with the imbalanced training set problem is *re-sampling*: Either *down-sampling*, which extracts a smaller set of majority samples while preserving all the minority samples, or *up-sampling*, which increase the number of minority samples by replicating them. Although down-sampling may lead to loss of informative samples and up-sampling leads to higher computational cost and may amplify the

[10] In the training set, if there are significant fewer training samples from one class than from the other, the training set is considered as imbalanced; these classes are named the minority class and the majority class, respectively.

labeling errors in the minority class, theoretical and empirical studies show that both methods produce acceptable results, and thus are widely employed in real world application [40, 45, 76].

1.2.5 Evaluation of Classifiers

To estimate the accuracy of a classifier with respect to a training set, *cross-validation* is one of the most widely used techniques. In cross-validation, all labeled samples are divided evenly into k subsets. Each subset is used as a test set for a classifier trained on the remaining $k-1$ subsets. The empirical accuracy is given by the average of the accuracies of these k classifiers. In the extreme case each subset consists of only one single example. This is called *leave one out cross-validation*.

Label of the sample	true	false
Classified as positive	TP	FP
Classified as negative	TN	FN

Table 1.1: Contingency table.

Given a sample s from the test set which is labeled as either *true* or *false*, indicating that either s belongs to a category c or not, after feeding it into the classifier the output is either *positive* or *negative*, suggesting that s is classified as belonging to c or not. The statistical results of a cross-validation can be show in a contingency table as Table 1.1, in which TP, or true positive, is the number of documents labeled as true and classified as positive; FP (false positive), TN (true negative) and FN (false negative) are defined similarly. Two fundamental measures for evaluating classifiers, *recall* and *precision*, are defined as:

$$recall = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{TN}} \tag{1.12}$$

and

$$precision = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{1.13}$$

In text classification, recall denotes how many *true* documents are successfully classified, and precision shows how many documents classified as *positive* are correctly classified. High recall and high precision imply that the classifier performs well, but they may be misleading if examined alone. For example, a trivial classifier that says "positive" to any document will have a perfect recall, but the precision will be low; a very strict classifier that rejects most documents may have a high precision but low recall.

The concepts of recall and precision and their variations are widely applied to data mining and information retrieval. For instance, in Web search, recall refers to the ratio of the actually retrieved Web pages to the number of all pages which should be retrieved, and precision refers to the ratio of number of Web pages relevant to the query to all pages retrieved. When evaluating our approaches of Query Extension and Query Categorization, we use similar measures, which will be introduced in Chapter 2.

1.2.6 Hypertext classification

The text classification techniques we introduced in the previous section are general approaches that can be used for all text documents. However, new challenges arose with the emergence of the WWW. Most Web pages are written in hypertext, which may bring in additional informative features, but can also introduce noise. For example, formating or font HTML tags like <h1>, <b>, and <i>[11] suggest the importance of the text, while the pages connected to a Web page by hyperlinks may give hints on how to classify this page, but navigation links and advertisements carry redundant, misleading information. Different approaches have been proposed to extract the unique characteristics of hypertext to utilize the informative features and avoid the deceptive ones for categorization.

The most intuitive approaches are either to treat hypertext as plain text without distinguishing any HTML specifications, or to convert hypertext to plain text by discarding all HTML

[11] In HTML standard, tags are used to define HTML elements, which are basic blocks of HTML documents. For example, <h1> defines the first level headings, <b> or <i> indicate that an element should be displayed in **bold** or *italic* font respectively.

tags. A simple improvement to the latter approach is to emphasize the elements which are important, for example by replicating the title, the headings, and the bold or italic elements. The performance of these approaches merely depends on the accuracy of the chosen classification technique and does not take advantage of the rich information of hyperlinks.

Some researchers suggest that instead of using only the content of a Web page for classification, the surrounding linked pages should be also considered. For instance, approaches bringing the content of all linked pages, or selective information from linked pages like emphasized elements and, anchor text[12] is explored in [3, 14, 28].

On the other hand, inspired by the HITS algorithm, approaches to classify Web pages by link analysis instead of text classification are proposed in [15, 16]. As mentioned in Section 1.1.1, the phenomenon of topic drift makes these approaches more suitable for *clustering*[13] than for categorization.

Hypertext classification is still one of the most challenging areas in information retrieval. The two most well known Web directories, Open Directory Project and Yahoo! Directory are compiled by humans, and none of the existing commercial Web directories is built up using automatic hypertext classification techniques.

1.3 Social Network Analysis

For better understanding the influence of social relations, social scientists have employed network theory to model societies; this approach is called *Social Network Analysis* (SNA). According to [9], SNA "provides an answer to a question that has preoccupied social philosophy since the time of Plato, namely, the problem of social order: How autonomous individuals can combine to create enduring, functioning societies".

In social networks, one of the most important assumptions is the notion that individuals are

[12] The HTML element between <a> and .

[13] Unlike "categorization", which is related to a explicit subject, or category, "clustering" tends to discover well connected components regardless of a subject or category.

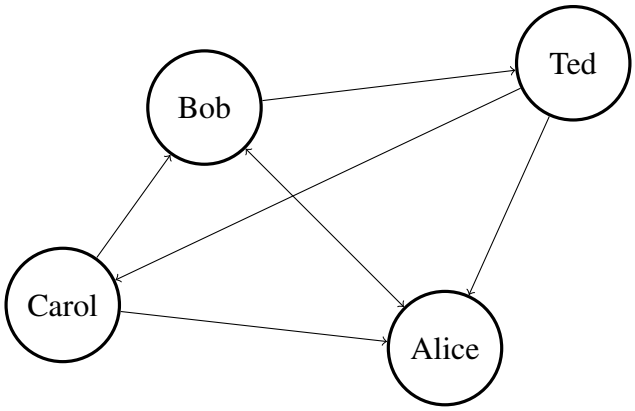


Figure 1.5: A social network of "like".

embedded in the webs of social relations and interactions. In network models, the individuals (or actors) are denoted by nodes, and the relations (or ties) between them are represented by edges. SNAs focus on the ties among actors instead of the attributes of each actor. Figure 1.5 models a simple "like" relationship, in which each node denotes a person, and a directed link $a \rightarrow b$ means that person a likes person b.

To collect the ties among actors, the *full network method* inquires every actor about their ties to all other actors. In some cases this method is impractical or too expensive; the *snowball method*, which is similar to the expansion from the root set to the base set in the HITS algorithm (see Figure 1.3), adds new actors by following known actors' ties.

One of the important properties of an actor is its power of influence in the network. Some researchers claim that, the *centrality* of the actor, or its position in the network, which can be measured as the distance to other actors, does not imply *power*. For example suppose that actors a and b have the same number of ties, actors connected to a reside in a well connected subgraph, but actors connected to b are isolated. Actor a is more central than b because the distances from a to other actors in the network are short, but actor b is more powerful because b has more influence on its neighbors while a's neighbors have many connections so they do not relay too much on a [8].

In this chapter we briefly reviewed the history of Web search engines and Web directories, and the underneath information retrieval techniques, including some Web search algorithms and text classification methods. We also introduced the basic concepts in social network analysis. In the next chapter, we present our approach to improve the effectiveness of Web search engines through the use of Query Extension and Query Categorization.

Chapter 2

Improvements on Existing Search Engines

2.1 Introduction

As mentioned in Chapter 1, Web search engines prevail these days as the dominant tool to find information in the WWW due to their fast response time and acceptable accuracy. However, sometimes it may still be hard for the average user to find the exact information they want by querying a search engine. We observed two facts that might lead search engines to produce undesirable results:

1. The result set returned by a search engine is based on the *words* of a user's query, not the *intention* of the user. For instance, if we search for "automaker" using a search engine like Google, surprisingly the search engine will not return in the result set any websites from the leading companies in the automotive industry like GM or Toyota. The reason why some of the most relevant Web pages are ignored by the search engine is that they do not contain the words "automaker" on their websites.

2. The number of URLs in the result set returned by a search engine is sometimes huge and may contain too many URLs not related to the user's interests. For example, different people sending the query "raptor" to a search engine looking for Web pages related to 1) a basketball club based in Toronto, Canada, 2) some breeds of birds, 3) some kinds of

dinosaurs, or 4) a type of fighter aircraft, will get the exact same set of URLs. Users may try different combinations of keywords to try to retrieve results which are more related to their needs, but the way to combine the keywords differs from one search engine to another.

Briefly speaking, looking for exact matches of query words in Web pages limits the ability of Web search engines to return pages which are relevant to a query but do not contain the query words; furthermore, the unclassified result sets returned by search engines make it difficult to locate the pages that are most relevant for a particular user. To address the aforementioned problems, we propose approaches to expand a query so that more Web pages will be introduced into the result set, and to refine the result set by selecting the Web pages according to the user's intention so that mainly relevant pages are presented to the user; we call these approaches *Query Extension* (QE) and *Query Categorization* (QC), respectively.

In Section 1.2.5 we introduced *recall* and *precision* for the evaluation of text classifiers. In this thesis, we use the related terms *completeness* and *accuracy* which will be formally defined in Section 2.5.6. For the time being, we loosely treat completeness as recall, and accuracy as precision, since these terms refer to how many Web pages relevant to the query are retrieved, and how many Web pages retrieved are relevant to the query, respectively.

2.2 Related Work

In Chapter 1 we have shown that text-indexing Web search engines are vulnerable to spamming, and link-based analysis algorithms like HITS suffer from topic drift. Moreover, although the PageRank algorithm ranks Web pages by analyzing the hyperlinks contained in them, the set of pages that are included in the result set and returned to the user is still decided by the appearance of the query words. We observe that Google suffers from the completeness problem, as the previous "automaker" example shows.

Different approaches have been proposed to cope with the limitations of text-indexing Web

search engines, which we can summarize in three categories: 1) Expanding query words by employing a thesaurus [36, 58] or by extracting a conceptual frame for the query [30]; 2) retrieving Web pages by using fuzzy logic or soft computing techniques [63]; 3) using natural language, instead of "keywords", to specify a query and then reformulating it into a new query more suitable for text-indexing search engines by performing inferences based on "common-sense reasoning" [51]. Although the second and the third approaches could be employed to help naïve users who have little knowledge of Web search engines and do not know how to combine keywords, they will not help more knowledgeable users who are familiar with and utilize Web search engines frequently. In addition, these two approaches are complex and their implementations are complicated. Thus, we believe that they are not the answer to the incompleteness problem. The first category of approaches promises to increase the number of URLs in the result set, however it also introduces more irrelevant Web pages; without appropriate post-processing, it will be more difficult for users to locate the information that they need. Our approach extends a query in a similar but simpler way compared to the first category, and avoids the undesirable results by our post-processing technique of *Query Categorization*.

Looking back at text-indexing search engines and link-based search algorithms discussed earlier, we observe that none of them takes into account the user's intention (other than the original query). For example, with a simple query like "raptor", there is no way to tell whether the user is interested in the basketball club *Toronto Raptors*, or the jet fighter *F-22*, so search engines can only return a mixture of every kind of Web pages related to the query word "raptor". The order in which the Web pages are presented to the user represents the importance granted by the search engine to each page within the whole collection, but this importance does not necessarily correspond to what the user wants.

Researchers have proposed various approaches using Web Page Categorization technologies to increase the accuracy of search engines. As shown in [21, 22], an interface showing category information in search results is very effective. The works in [18, 44] show approaches to classify search results into categories; these approaches re-arrange the result set in a more

conceptual way, but without the user involved and, furthermore, they do not present the results in a way friendly to the user's intention. In [46, 50, 61], user's interests are discovered from the user's searching history and/or browsing history; this approach has to monitor the user's behavior for a long period of time, and it would fail in the situation when several users share a computer. In [26, 85], search results are divided into concept clusters to help the users find the information they need; however, unlike a category, a concept cluster is difficult to define and thus might cause confusion. The research in [32] shows an approach for modifying a query according to the category specified by the user; the modified query is then sent to a search engine. Modifying a query to include category information involves deep knowledge of how certain search engine handles keywords in a query, so the same approach would fail when applied to a different search engine.

In this paper, we propose *Query Extension* to improve the completeness of a search engine, and *Query Categorization* to increase the accuracy of the search results. In the next two sections we give a detailed discussion of *Query Extension* and *Query Categorization*. Our test system and the analysis of the testing results are presented in Section 2.5.

2.3 Query Extension

There is no restriction on which kind of content is published on the WWW, and there are no guidelines for content providers specifying how a Web page should be presented. To be more specific, content providers could, for example, use various words to refer to the same term because of the existence of synonyms. For instance, one could use any of the words *vehicles*, *automobiles*, *cars*, or even *autos* on Web pages dealing with cars without leading to any misunderstandings. However, such freedom can get text-indexing Web search engines in trouble.

When a user issues a query to a search engine, the search engine returns mainly Web pages which contain the specified query words; those pages containing synonyms or other variations

instead of the query words themselves are usually not included in the result set. This may not be desirable since the user is looking for information about the query, but not necessarily about the query words themselves, i.e., the result set is incomplete.

To overcome this incompleteness problem and to improve the coverage of the result set, we propose here to extend a query so it includes not only the original query words but also some variations; synonyms are reasonable extensions for the original query. We take this idea a step further: When we send, for example, the query "vehicle" to a search engine, why are those Web pages containing the plural "vehicles" sometimes omitted? We should consider the singular and plural forms of nouns as query extensions, as well as the different tenses of verbs, and the comparative and superlative degrees of an adjective, i.e., the inflected forms of words.

To yield query extensions, we propose to construct a dictionary of synonyms and inflected forms, that we call the *inflection dictionary*, for English words. There are good synonym dictionaries like WordNet [25], and the well-known Merriam Webster Thesaurus [59], both of which are manually compiled. Some approaches for the automatic construction of a thesaurus are discussed and experimentally compared in [69] with WordNet, and the conclusion is that the quality of the results obtained with automatic methods is not as good as that of WordNet. Therefore, it is reasonable for us to use WordNet and Merriam Webster Thesaurus when building the synonym part of the inflection dictionary. The Merriam Webster Dictionary provides inflected forms for irregular adjectives, nouns and verbs, while regular inflected forms could be derived by using grammatical rules, for example, add "ed" to verbs for the past tense, add "s"to nouns for the plural form. Therefore, given an English word, we can generate a list of synonyms by consulting WordNet and Merriam Webster Thesaurus, and a list of inflected forms can be selected from the Merriam Webster Dictionary and by using grammatical rules.

We implemented a Perl script for the construction of the dictionary, and we examined the dictionary entries by hand; the implementation details will be given in Section 2.5.2. Each entry in the inflection dictionary has the form $e \rightarrow I$, where $I = \{s_1, s_2, \ldots, s_m\}$ is a list of the synonyms and inflected forms, or the *inflection set*, for e. Upon input of a query q =

$\{w_1, w_2, \ldots, w_n\}$, we augment each word w_i with its inflection set I_i, so the query q is extended to a product set $Q = I_1 \times I_2 \times \ldots \times I_n$, and each new query q'_k in Q has the form $\{I_{1l}, I_{2j}, \ldots, I_{nt}\}$, where I_{iy} is the yth entry in the inflection set I_i. An example of the this procedure is shown in Section 2.5.2.

2.4 Query Categorization

Regardless of how Web pages are collected, it is the goal of a Web search engine to organize them in a well-ordered way when returning the result set to the user. This ensures that the URLs of those pages which are, both of high quality and relevant to the query appear before others.

Using the above "automobile" example, one important observation is that the different meanings of the same query can be categorized. For example, in the Open Directory Project, websites about prices of used cars are in the category "Shopping: Vehicles", and sites about the history of automobiles can be found under "Recreation: Autos: Enthusiasts". Inspired by this observation, finding a way to categorize a query and then return the Web pages in the same category as the query would be a good attempt to improve the search results. This approach could be separated in two steps: The first is to identify the category of the query, and the second is to decide how closely related a given Web page from the result set is to this category.

2.4.1 Word Classification

To identify the category of the query we have to consider the following facts. Most English words have multiple meanings (polysemy) depending on the context and, therefore, could be classified into several categories. For example, while the term "*tree*" is used as a type of data structure in Computer Science, we also know what *tree* means as a life form. Consequently, when a user sends a query to a search engine, it is the user and no one else who knows what s/he really means. Therefore, to identify the category of a query, it is reasonable to let the user

decide to which category the query belongs. We decided to present the user with some possible categories (the procedure used to select the categories is described in Section 2.5.4) and the user can either pick one of them, or if the proposed categories are not satisfying enough, the user could choose any other category from a directory.

It is worth mentioning that our strategy to pick a category for the query complies with the typical approaches used in automatic categorization of words, the basic steps of which include dictionary lookup and ambiguity removal [37, 66].

2.4.2 Web Page Classification

Given a category specified by the user, we need to decide which Web pages from the result set belong to this category, so that only relevant pages are presented to the user. We have introduced several text classification techniques using the *vector space model* in Section 1.2. In this thesis, we use these techniques to categorize Web pages. We give implementation details in Section 2.5.4. Briefly speaking, a training set which consists of high quality Web pages classified into categories is collect beforehand; classifiers (for words and Web pages) are trained using the training set (`tfidf` is used as the weight of each word in feature vectors). Then the following Query Categorization procedure is used:

- Categorize the query by using word classification, as described in Section 2.4.1.
- Use a search engine to collect Web pages according to the specified query.
- Classify the Web pages in the result set and discard those pages which do not belong to the query's category.

2.5 Implementation and Testing

To test the approaches proposed in this research, we implemented a system consisting of the following modules.

- A User Interface which parses the user query and presents the final results.
- A query extension module which extends the user's query.
- A retrieval module which collects pages from a search engine.
- A categorization module which performs word and Web page classification.
- A ranking module.

Our system was implemented on a machine running Windows XP, and it was written in C++ under the Microsoft Visual Studio developing environment. We also implemented several auxiliary Perl scripts, for example to construct the inflection dictionary. We give below a description of these modules.

2.5.1 User Interface

For simplicity, a text terminal is used as the interface to read the input and present results to the user. To compare the results obtained with our approach with the results from a commercial Web search engine, we present both lists of results in a Web browser, side by side. The interaction between the user and the system is listed below.

1. The user enters a query.
2. A list of categories is presented.
3. The user picks one category.
4. After processing the query, a Web browser is opened and the results are presented.

A more detailed description of each step is given below.

2.5.2 Query Extension Module

To expand a query, we implemented a Query Extension Module in the test system which consults the inflection dictionary to get all synonyms and inflected forms of each query word. Each inflection will replace the original word in the query to form new queries (as explained in Section 2.4.1), which are passed to the retrieval module to obtain the expanded search result. Note that since for multiple-word queries, the number of new queries could be impractical for the retrieval module if every word is extended. Hence, we allow the option of enabling or disabling query extension: If a query word is surrounded by square brackets "[" and "]", the word will be extended; other words will remain the same in the extended query.

When implementing the test system, we did not build a dictionary that contains every English word. Instead, for the purpose of testing our algorithm we only added into the inflection dictionary the words which are used as queries in our tests. We installed a WordNet database in our system and implemented a Perl script which 1) retrieves the synonyms of a given word from the WordNet API, and 2) submits the word to Merriam Webster Thesaurus website and parses the response to extract inflected forms. We noticed that the "synonyms" retrieved from WordNet are sometimes not really synonyms of the given word. For example, "estimate", "gauge", "guess" and "judge" are returned as synonyms of the verb "approximate"; although "estimate" and "guess"can be considered as synonyms of "approximate", we do not think "gauge" and "judge" are good ones. Therefore, we manually examined all synonyms of a given word to exclude the misleading ones, and added into the inflection dictionary the regular inflected forms including plural form of a noun, tenses of a verb and the comparative and superlative degrees of an adjective. In total we created 47 entries in the inflection dictionary, two examples of which are shown below.

algorithm `algorithms algorithmic approach method process procedure`

approximate `approximated approximating approximates`
`approximative approximation rough near estimate guess`

Given the query "internet [algorithm]", the brackets indicate that the word "algorithm" should be expanded. The query extension module will search for inflections of the word "algorithm" in the inflection dictionary, and form new queries by replacing "algorithm" with the inflections. Therefore, the output of the query extension module is a list of these queries: "internet algorithm", "internet algorithms", "internet algorithmic", "internet approach", "internet method", "internet process" and "internet procedure", in this example.

2.5.3 Retrieval Module

As we know, a typical Web search engine consists of a crawler, an indexer, and a query server. In this thesis, we did not use a Web crawler to download Web pages, instead we implemented our algorithms on top of an existing search engine. This approach allowed us to focus our attention on trying to improve the search engine's effectiveness. We chose Google as the search engine to be used to compare against our system for several reasons: 1) Google is currently the largest search service provider in the world; it is reasonable to compare our approaches with it; 2) the fact that Google returns Web pages based on the occurrences of query words and not users' intention is one of the motivations of this thesis; 3) Google provides the *Google Web Search API* [1] that allows software developers to query the Google search engine from their own programs.

Upon receiving a query from a user and extending it with the query extension module, the retrieval module of our system sends the extended queries to Google, one by one. Google returns a list of URLs for each extended query. The Web pages corresponding to these URLs are then fetched by the retrieval module and stored in a MySQL [60] database, which has a C API that allows our program to access the database. The database contains one main table,

[1] http://code.google.com/apis/websearch/, which is now claimed to be deprecated, but will still be available for some time. In the early stages of this research we implemented a crawler to fetch Web pages, but we soon realized that it would be too time consuming to cache the whole Web and this was not the main purpose of this thesis: At that time Google allowed developers to fetch up to 2000 results for each query; we thought that this number of pages was enough for our research. Since then Google has changed the interface and the policy of its search service several times; in the course of this research we have had to re-implement this module of our program to reflect the API and policy changes.

called “PAGES”. The fields in the table PAGES are shown in Table 2.1.

Field	Type	Description
ID	unsigned int	Web page ID
URL	varchar	URL of the page
CONTENT	longtext	Hypertext content of the page

Table 2.1: Fields in the MySQL table PAGES.

2.5.4 Categorization Module

As mentioned in Section 2.4, the categorization module is used to obtain the category information of the user’s query, and to classify Web pages to decide which ones should be kept in the result set. We used the Bow library [56] as our text classifier and we implemented Perl scripts to parse Web pages and allow the Bow library to classify them.

Bow is a C library containing statistical text analysis and information retrieval tools which implements classification methods including Naive Bayes and SVM. It takes a training set in a-category-per-subdirectory fashion, i.e., the training-directory, or the directory which contains the samples in the training set; it consists of subdirectories, each of which holds the samples which belong to a category. This gives us flexibility to manipulate the classifier by organizing the subcategories as we require. For example to make Bow a binary classifier, we train Bow with a training-directory which consists of two subdirectories, each of which contains positive and negative samples for a category; if we put training samples belonging to several categories into different subdirectories, Bow will act as a multi-class classifier. After testing the performance of different classification methods provided in Bow on, both, Web pages and query words, we chose SVM to classify Web pages since SVM is very accurate for text classification, as explained in Section 1.2.3, and chose Naïve Bayes to classify query words because Naïve Bayes is very fast and its performance for word classification is similar to SVM.

The Web pages used to build our training set were collected from Google directory[2], which

[2]This service has been discontinued since July 20th, 2011 [31].

adopts the hierarchical structure of Open Directory Project (ODP), but gives more information than ODP does by including, for example, the Google PageRank of each Web page listed in the directory. To keep the implementation simple, we did not include all hierarchical subcategories of Google directory. Instead, we selected 64 second level categories in Google directory as our testing base and we collected Web pages whose PageRanks were higher than 0.5[3] under each category as the training samples. The list of 64 categories and the details of how the training set was selected are shown in Appendix A. For each category, we also constructed a set of negative samples by randomly picking the same number of pages from other categories. These negative samples along with the positive samples in each category compose a training set for the binary classifier with respect to the category.

Web pages are usually written in hypertext. As we have explained in Section 1.2.6, hypertext classification is one of the most challenging area in information retrieval. In our implementation we have tested 3 approaches for modeling hypertext: Either all HTML tags were kept (all tags were modeled in the feature vector) or discarded (only plain text was modeled in the feature vector); in the third approach the Web pages were converted into plain text by analyzing HTML tags so that navigational or structural tags were discarded while the informative text made-up by tags was up-sampled (by duplicating the text) in the feature vector. Navigational or structural tags are, for example, hyperlinks to other pages in the same website, or to different sections of the same page, The informative HTML tags include title, headings, emphatic or bold fonts, and URLs of hyperlinks inside anchor tags, i.e., the links embedded in <a href="http://···">, as we believe that the URL of a Web page is also informative. Our experimental results show that among the three approaches the latter one, which converts Web pages to plain text and up-samples the informative text, outperforms the other two and, thus is adapted in our test system. We implemented Perl scripts to pre-process the Web pages in the training set, and to train classifiers.

[3]The reason why we chose 0.5 is that the score of PageRank is log based, and 0.5 is a fairly good score for a Web page. If we collected Web page with higher PageRank, for example 0.6, the number of qualified pages in each category would dramatically decrease and the training set would be too small.

Before any user queries were processed, we collected the training set and trained the classifiers: The word classifier, the Naïve Bayes classifier for word classification, was trained using a training set containing all categories; the Web page classifier, the SVM binary classifier with 64 statistical models, each of which corresponds to one category was trained using training sets containing *positive* and *negative* samples for each of the corresponding categories.

The query categorization module is invoked once the user issues a query to our system, as explained in Section 2.4.2. The word classifier then assigns a probability for the query belonging to each category and the system presents the user with a list of categories whose probabilities are greater than zero, arranged in descending order of probability. The user can then pick one category from the list; if none of the categories in the list fits the user's intention, the user could ask to have all categories listed and then pick one. For example, in response to the query "matrix", our system presented to the user the following options.

```
1: Science/Math
2: Computers/Artificial_Intelligence
3: Arts/Movies
4: Society/Philosophy
5: Computers/Computer_Science
6: Science/Physics
7: Computers/Programming
8: Computers/Security
9: Computers/Multimedia
10: Computers/Graphics
```

After the user has selected a category, the system invokes the retrieval module to fetch Web pages satisfying the query and stores them in the result set. The Web pages in the result set are then classified by the Web page classifier. If a page is accepted by the classifier, it will be kept; otherwise, it will be discarded from the result set. Thus, the final result set will only consist of Web pages which belong to the user's selected category.

2.5.5 Ranking Module

The order of the Web pages in the result set determines their relevance to the user's intentions and the order is decided by a final score that we compute for each Web page. So it is very important for us to design a ranking system that accurately reflects what the user wants. Because we use Google to retrieve Web pages and Google returns pages in the highest-PageRank-first order, we assume that the Web pages in the result set are of relatively high quality. Hence, we assume that it is safe to apply text-indexing methods to rank Web pages in our system and thus keep the implementation simple. We assign each Web page a `tfidf`-like score (`tfidf`, or term frequency inverse document frequency, has been described in Section 1.2.1), then combine this score with this page's PageRank to compute the final score; the procedure is described as follows.

1. Computing term frequency. Given a Web page, we count the number of occurrences of the query words. If the query words appear in the Web page's URL, or if they are enclosed by HTML tags for title, headings, emphatic, bold, or underline fonts, we consider that these occurrence of the query words are more meaningful than the occurrences anywhere else in the text of the Web page. Thus, when computing the term frequency tf_k of a query word w_k, tf_k is incremented by some value e when the word is encountered in a Web page, and the value e is determined by the following rules:

   - $e = 1$, if w_k is in the page as plain text.
   - $e = 2$, if w_k is in emphatic, bold or underline font, or any headings.
   - $e = 3$, if w_k appears in the Web page's title.
   - $e = 5$, if w_k appears in the Web page's URL.

   In the case that the query is extended, the extended words are considered the same as the original query words.

2. Computing document frequency. The document frequency df_k of a query word w_k is computed at the same time as the term frequency t_k is computed. For every Web page in the result set, if w_k appears in this page, then df_k is increased by 1.

3. Computing `tfidf`. The `tfidf` of a query word w_k is then computed using

$$\texttt{tfidf}_k = tf_k \cdot log\frac{N}{df_k}, \tag{2.1}$$

where N is the size of the result set. Note that Equation (2.1) is equivalent to Equation (1.4) except that we assign different value e to query words according to their importance as indicated by HTML tags.

Based on the observation that some Web pages are informative but relatively short, and at a disadvantage if the `tfidf` is computed using Equation (2.1). We normalize `tfidf` by the length s of the Web page, and Equation (2.1) is revised as

$$\texttt{tfidf}_k = \frac{1}{s} \cdot tf_k \cdot \log \frac{N}{df_k}. \tag{2.2}$$

The score t of a Web page with respect to the query is the sum of `tfidf`s of all query words:

$$t = \sum_{w_k \text{ is a query word}} \texttt{tfidf}_k \tag{2.3}$$

After each score t of every Web page in the result set is calculated, all scores are then linear normalized to the interval [0, 1]: Let t_{max} denote the maximum score of Web pages in the result set and t_{min} denote the mininum score, the linear normalized score t' of score t is computed by

$$t' = \frac{t - t_{min}}{t_{max} - t_{min}}.$$

Note that we will keep using the notion t instead of t' as the normalized score in this thesis without leading to confusion.

4. Estimating PageRank. The computation of score t of a Web page in the previous step is based on only text-indexing and does not take into account the page's hyperlink information. As described in Section 1.1.2, the PageRank is a measure of the authority of a Web page based on hyperlinks. At early stages of this research, Google Web Search API provided the PageRank associated with each Web page in the result set, but the service was discontinued later. However, we know that Google returns the results in descending order of PageRanks, so we can estimate the PageRank p of a page as follows: We assign $p = 1$ to the first page in the result set returned by Google and $p = 0$ to the last page, the value p for the ith page in a Google's result set of size N_g is

$$p = 1 - \ln \frac{i}{N_g}. \tag{2.4}$$

When computing the final score of a page in our ranking scheme, we want text-indexing score t to be the main factor, but we also want to emphasize the scores of Web pages which are listed as the top ones in the result set returned by Google. Therefore, we chose a logarithmic scale for p so that p decreases rapidly when the order i increases and hence many pages are assigned a relatively small value p, and only the final scores of the top pages in Google's results are influenced by their p values.

5. Computing the final score. We compute the final score S of a Web page as the weighted sum of text-indexing score t and PageRank p:

$$S = 0.8 \times t + 0.2 \times p. \tag{2.5}$$

After performing a number of experiments to determine the relative weight of t and p, we decided to set the weight of p to 0.2. This way our ranking function can take advantage of the implicit information of the order of Google's results, but our ranking results are not influenced by the Google's order significantly.

After computing the score S for each page, our system presents the result set to the user, in descending order of S.

2.5.6 Testing

To assess the methods that we proposed in previous sections, we tested the performance in three regards; query extension, word classification, and Web page classification. We asked several graduate students in the Department of Computer Science at the University of Western Ontario to help us test our system. Each tester was asked to give 5 to 10 queries to the test system, and for each query the tester also specified one or more categories within which the tester wanted the result sets to be constrained. The test system returned a result set for each query and category combination; the tester then examined the Web pages in the result sets obtained by the test system and the pages returned by Google, and labeled each page "positive" or "negative" to indicate whether the page belongs to the category or not in their opinion.

Testing the Query Extension Module

We wish to point out that since Google Web Search API returns only 64 URLs for any given query, we cannot directly compare the result set obtained by Query Extension with that returned by Google, as Google Web search returns many more URLs than Google Web Search API does. By examining the result sets, we noticed that many Web pages in the result sets from Query Extension did not appear in the result from Google. For example, given the query "apocalypse", among the 64 URLs returned by Google Web Search API, only 3 of them contain the plural form "apocalypses", which is the only inflected form of "apocalypse" in the Inflection Dictionary. On the other hand, in the result set of "apocalypses" returned by Google Web Search API, the Web pages cover the same categories as those pages returned by "apocalypse": "Arts/Movies", "Society/Region_and_Spirituality", "Computers/Programming", and so on. Furthermore, the 2 result sets share only 2 Web pages, which are pages giving the definition of "apocalypse": www.wikipedia.org and www.merriam-webster.com. This means that even

in Google Web search, most of the URLs of Web pages which contain the word "apocalypses" will not show in the top 64 positions of the result set when searching "apocalypse", and this is an example of how Google focuses on the exact words of the query instead of its meaning. So by including both queries, Query Extension collects a result set which is a mixture of Web pages containing either "apocalypse" or "apocalypses", or both, and is a simple but effective way to expand the coverage when searching the WWW [4].

Testing the Word Classifier

We have shown an example of word classification in Section 2.5.4 for the query "matrix". When testing our system, the testers specified 30 queries, and 106 categories associated with these queries. When these queries were submitted to the system, among the categories suggested by our system, we found that 74 of them were in the 106 ones specified by the testers, and 21 out of the 74 categories were the top picks of the testers. Table 2.2 shows two examples of the categories specified by testers and those suggested by the system. For both queries "tree structure" and "GAP", the testers specified 4 categories and, our system suggested all 4 categories for "tree structure" but only 2 for "GAP". Even though only 2 out of 4 categories specified by the testers for "GAP" were suggested by our system, one of them was the top category picked by the testers: "Shopping/Clothing".

The purpose of word classification is to suggest possible categories to users. Even when word classification fails, users still have the option to browse the entire set of categories in the system and choose one. Hence, we did not try to use more sophisticated classification techniques like lexical and semantic analysis [38, 66, 37, 69]. Queries usually consist a few words and in the context of text classification, classifying extremely short documents is a hard

[4] A few months ago, after we had developed our system, Google modified its search engine to include in its result set Web pages containing synonyms of the original query words. However, the algorithm used by Google to select synonyms is not of the public domain. Even for this new version of Google's search engine, we noticed that for the previous query "apocalypse", among the first 100 Web pages returned by Google Web search, the word "apocalypses" appears only twice. When searching for the query "algorithm", we only found one inflected form "algorithms", which appeared once; remember that in our inflection dictionary, "algorithm" have 6 inflected forms or synonyms: `algorithms`, `algorithmic`, `approach`, `method`, `process`, and `procedure`. The above observations suggests that Google's selection of synonyms is not as effective as our inflection dictionary.

task because of the lack of features. When designing our system, we expected the performance of the word classifier to be rather poor. But the test results showed a precision of $74/106 = 0.7$, which we consider as acceptable. We believe that the reason why the word classifier performs well in our system is 1) the queries are usually simple, composed of only a few words, and the testers tend to pick those queries that obviously could belong to several different categories for testing; 2) the training set we have chosen consists of high quality Web pages, which give the classifier a good base to function.

It is worth mentioning again that, in our system, users are given the option to browse all categories and pick one, if the word classification does not suggest the category intended by the user.

Query	Categories Specified by the Testers	Categories Suggested by the WC
tree structure	1 Computers/Algorithms 2 Science/Biology 3 Home/Gardening 4 Society/Genealogy	1 Science/Biology 2 Computers/Algorithms 3 Science/Physics 4 Computers/Artificial_Intelligence 5 Society/Genealogy 6 Reference/Knowledge_Management 7 Computers/Programming 8 Home/Gardening 9 Society/Folklore 10 Society/Holidays
GAP	1 Shopping/Clothing 2 Business/Employment 3 Recreation/Outdoors 4 Society/Philosophy	1 Shopping/Clothing 2 Recreation/Roads_and_Highways 3 Science/Math 4 Recreation/Outdoors 5 Health/Addictions 6 Recreation/Roads_and_Highways 7 Computers/Hardware 8 Computers/Artificial_Intelligence 9 News/Media 10 News/Colleges_and_Universities

Table 2.2: Comparison of categories specified by testers with categories suggested by word classification.

Testing the Web Page Classifier

To evaluate Query Categorization, we asked testers to examine the result sets in two ways. The first one is to assess the *accuracy* of the results, which measures among all Web pages returned by our algorithm, how many of them are correctly classified into the category specified by the tester. The second one is to assess the *completeness* of the result set, which specifies among all Web pages which belong to the category specified by the tester, how many of them are correctly kept in the result set. In other words, the higher the *completeness* is, the more Web pages which belong to a category are included in the result set; the higher the *accuracy* is, the fewer Web pages which are irrelevant to the category are included in the result set.

Query	Category	# of URLs Correctly Classified by Our System	Total # of URLs in the Result Set of Our System	# of URLs Belonging to the Category in Google's Result Set
alcohol drive	Society/Crime	35 (63%)	56	18 (28%)
	Health/Addictions	26 (70%)	37	7 (11%)
	Recreation/Autos	17 (81%)	21	7 (11%)
	Computers/Systems	15 (83%)	18	15 (23%)
raptor	Recreation/Birding	76 (78%)	92	26 (56%)
	Sports/Basketball	34 (56%)	61	0 (0%)
	Society/Military	25 (74%)	34	8 (13%)
tree structure	Computers/ Algorithms	54 (65%)	83	51 (80%)
	Home/Gardening	32 (91%)	35	2 (3%)
	Science/Biology	10 (63%)	16	1 (2%)
	Society/Genealogy	7 (83%)	8	0 (0%)
zip	Shopping/Clothing	34 (62%)	55	22 (34%)
	Computers/Algorithms	10 (45%)	22	13 (20%)
	Recreation/Roads_ and_Highways	12 (71%)	17	5 (8%)

Table 2.3: Accuracy of Query Categorization.

To measure the accuracy, we asked the testers to manually identify in the result set the Web pages which belong to the categories specified by them. Then we compared the number of these pages with the total number of pages in the result set. Table 2.3 shows several examples of

such comparisons. For instance, a tester gave query "raptor" and specified "Society/Military" as the category; the testing system returned a result set containing 92 Web pages, among which 76 pages (82%) were considered by the tester to be relevant to "Society/Military". We also manually classified the Web pages in the result set returned by Google API, and listed the number of URLs that belong to the corresponding category in the 5th column in Table 2.3. For the previous example, we found that 8 out of 64 Web pages (13%)[5] returned by Google can be classified in the category "Society/Military" when searching for "raptor".

Note that in some cases the Web pages returned by Google concentrated in one category and thus had relatively high accuracy in this particular query-category combination (for instance, the query "tree structure" in the category "Computers/Algorithms" in Table 2.3). Nevertheless, the average accuracy of the result set from Google was pretty low because, after all, Google returns a mixture of URLs belonging to different categories without knowing the user's intention; for example, as we can see in Table 2.3 the number of pages which belongs to "Home/Gardening" or "Science/Biology" is next to nothing. It is also worth to mention that the number of Web pages in the result set returned by our algorithm was greater than that by Google, because we applied Query Extension when searching. For some categories Google returned none or very few pages; by applying Query Extension we expanded the result set for every category. For example, Google returned nothing about the basketball club "Raptors" if searching for "raptor", but as "raptors" is one of the inflection forms of "raptor", there are a lot of Web pages belonging to category "Sports/Basketball" in the result set returned by our algorithm, as shown in Table 2.3.

In summary, for the 30 queries and 106 query-category combinations, the overall accuracy of our system was 72%, which is a good score compared with other general Web page classifiers. For example, the precision of a Naïve Bayes classifier, when being applied to a relatively small corpus containing 755 Web pages, was less than 70% as reported in [74]. A lower precision of 23% was reported in an experiment performed using a SVM classifier on the Yahoo!

[5]Google Web Search API returns 64 pages for a query, so the percentage of Google's results in category "Society/Military" for query "raptor" is 8/64 = 13%, as shown in Table 2.3.

Directory [52]. We list the accuracy of our classifier for each category in Table 2.4. Regarding the result sets returned by Google, among the 6,779 Web pages selected by Google, we identified 1,671 pages which belonged to the categories specified by the testers, thus the accuracy of Google was only 24.6%.

In Table 2.4 we notice that although our Query Categorization has high accuracy on several categories, it did not perform well on others. We examined categories with lower accuracy, and found that the cause is mainly the inherent ambiguity of certain categories. For instance, "Arts/Movies" and "Arts/Television" cover all genre of movies, TV shows and dramas. Although it is easy for humans to identify keywords like *director*, *plot*, *cast*, and then label a Web page containing these words as "Arts/Movies" or "Arts/Television", it is a big challenge for a classifier to classify a page if it contains too much information about the plot itself, as the classifier is easily mislead into assigning the page to a category related to the plot, rather than "Arts/Movies" or "Arts/Television". The second reason is that several categories overlap each other and therefore are hard to classify. This situation tends to happen especially when dealing with categories under "Computers" and "Science".

Since we have employed Query Extension to include more Web pages in our result set, we want to keep as many relevant pages as possible after applying Web page classification, to make the completeness as high as possible. For the 30 queries and 106 query-category combinations used in our experiments, among the 1,671 Web pages which should have been included in the result sets, our system kept 1,433, or slightly over 85%; i.e., the completeness is 85%. Table 2.5 shows an example of how the completeness is measured. For instance, given the query "crash review", we manually classified 22 Web pages in the original result set into the category "Games/Video_Games"; after applying Web page classification, we identified 20 of them that were kept in the final result set and returned to the tester.

In our system we use Google Search API as the crawler, therefore, the result set returned by Google after Query Extension contains every Web page included in our result set. This fact means that we assume the completeness of Google is perfect, if the category information is not

Category	Accuracy(%)
Computers/Security	95.65
Home/Gardening	95.45
Sports/Soccer	95.24
Society/Genealogy	94.44
Home/Cooking	90.91
Games/Video_Games	90.48
News/Media	86.36
Business/Investing	85.71
Health/Animal	85.71
Shopping/Home/and_Garden	84.21
Recreation/Birding	82.61
Arts/Music	81.82
Computers/Systems	81.82
Recreation/Autos	76.19
Society/Military	73.68
Science/Math	73.68
Health/Addictions	72.73
Recreation/Pets	72.73
Recreation/Roads_and_Highways	72.73
Computers/Programming	71.43
News/Newspapers	70.00
Computers/Hardware	66.67
Society/Crime	65.00
Sports/Tennis	63.64
Sports/Motorsports	63.16
Computers/Artificial_Intelligence	63.16
Shopping/Clothing	61.90
Science/Biology	60.00
Recreation/Outdoors	59.09
Sports/Basketball	55.56
Computers/Algorithms	52.94
Arts/Television	45.00
Arts/Movies	42.86
Computers/Graphics	42.11
Overall	72.03

Table 2.4: Overall accuracy of Query Categorization.

Category	Number of URLs that should be kept	Number of URLs kept
Games/Video_Games	22	20
Arts/ Movies	15	12
Arts/Music	6	5
Business/Investing	4	4
Arts/Television	2	1
Health/Conditions_and_Diseases	2	2
Recreation/Outdoors	2	0
Computers/Hardware	1	1
Computers/Programming	1	1
Recreation/Autos	1	1
Science/Math	1	1

Table 2.5: Completeness of Query Categorization with Query "crash review".

taken into account. However, recall that our experiments show that the accuracy of Google is 24.6% with respect to query-category combinations; i.e., the user has to browse many irrelevant Web pages to locate useful information. On the other hand, our algorithm has a 72% accuracy and 85% completeness, which means that our result sets mainly contain Web pages which are highly relevant to the queries and the categories, and improve the effectiveness of Web search.

We reviewed those Web pages misclassified by our system and found that they were ambiguous or lack some information needed for the classifier. For example, a Web page that mentions the movie "Crash" but discusses the society issues behind the movie, and a Web page which contains a video clip of a video game and basically no much text on it, are missing from our result set with respect to the categories "Arts/Movies" and "Game/Video_Games" respectively.

2.6 Conclusions

Modern Web search engines suffer from problems such as lacking the ability to include Web pages which contain synonyms and variations of the original query words, or considering only the words of a query but ignoring the user's intentions when computing results. The first

problem will cause the absence of some relevant Web pages from the result set, and the second one will produce a result set which contains, both, Web pages that are related and pages that are irrelevant to the user's intentions. To overcome these issues, we introduced the concepts of Query Extension and Query Categorization. We proposed algorithms for implementing these concepts, and implemented a system to test our algorithms.

Query Extension includes synonyms, transformations and inflections of each query word specified by the user. Transformations of a word include the singular and plural forms of a noun, the present, preterit, and perfect tenses of a verb, the comparative and superlative degrees of an adjective, and so on. Our algorithm is able to retrieve Web pages which are related to the query but do not contain the exact words in the query. These Web pages are usually ignored by text-indexing search engines, but our test results show that some very informative and relevant Web pages are included in this group and hence, should not be omitted.

Query Categorization involves two phases: First we analyze the user's query and find the category which best reflects the user's intention; after retrieving Web pages we select those which belong to that particular category. The Query Categorization algorithm gives search engines the ability to return search results according to not only the query words themselves, but also the user's intent, hidden in the meaning of the words.

We implemented Query Extension and Query Categorization algorithms and compared our results with Google's. The comparison shows that our algorithms are effective in improving the performance of search engines by extending the query words to retrieve more relevant Web pages, and by refining the result set to meet the user's intention more precisely.

During the testing procedure, we have encountered several issues which might be worth to investigate in the future. We list them below.

When constructing the Inflection Dictionary and applying Query Extension, we realized that as English words are able to be categorized, their synonyms should also be able to be categorized. If we could sort synonyms by categories, then when a category is specified by the user, we should apply the synonyms of the query only in this category to extend the query, so

the extended result set would include Web pages which are more relevant to the query and the category.

In the tests we saw misclassification of Web pages during query categorization on both directions: Either we kept Web pages which do not belong to a given category in the result set, or we discarded Web pages which should have been kept. We mentioned in Section 1.2.6 approaches to classify Web pages by taking advantage of the hyperlinks embedded in the Web pages and the network structure. It would be interesting to see if applying these approaches to Query Categorization would improve the accuracy of the classification system.

Chapter 3

Subject-Driven Community Mining

3.1 Introduction

The Internet has dramatically changed the way in which people communicate with each other. In the past few years *online social networks* (OSNs) have gained significant popularity. They have become pervasive platforms of communication, and they have attracted a large amount of attention from computer scientists, the intelligence community, and sociologists, among others. One of the most important issues, and most active research areas in this field, is to discover online communities in OSNs. According to [65] an *online community* consists of four components:

1. Social interactions.

2. A shared purpose.

3. A common set of expected behaviors.

4. A platform which supports interaction among members via the Internet.

Since in the above four components, social interactions in OSNs mainly take place through expected behaviors, for example posting and commenting, and the OSNs are interactive plat-

forms themselves, the two essential components of an online community are shared interests and social interactions.

There are several reasons why researchers may be interested in finding communities in OSNs. For example, these communities provide valuable information resources: They represent the sociology of the Web and their study provides insights into the structure, organization, and interests of the different sectors of society. Identifying these communities can, among other things, help intelligence departments or advertising companies to find particular targets.

We have observed that one of the most common activities in an OSN is *posting* and *commenting*. A member of an OSN usually has some *space* on which the user can post content (articles, pictures, or video clips) that s/he considers interesting; if the space is open to the public, other members can read the content and leave comments. These posting and commenting activities compose the essential interactions among members of an OSN.

Since people join OSNs for different reasons, and people have different interests, they may interact with others on various topics. The "friend" relationships provided by most OSNs do not capture the variety of interactions among OSN members. Thus, we propose in this thesis a *subject-driven community* discovery approach, which focuses on the most common activities in OSNs: Posting and commenting on a given subject. We also introduce the *interaction graph* to model these posting and commenting activities. In an interaction graph a node represents a member who posts blogs on a subject and a link represents a commenting activity by another member to the blog. Note that the interaction graph captures two essential properties of a online community we mentioned earlier: Shared interests and social interactions among members. We use a modified version of the HITS algorithm (introduced in Section 1.1.1) to locate members on an OSN who are well connected with respect to a given subject, and this collection of members forms the core of a online community.

Our main contribution in this field lies in being the first work, to our knowledge, to focus on subject-driven communities on OSNs, and to mine such communities by modeling posting/commenting activities with an interaction graph. In particular, the concept of subject-driven

community and our approach to model a community are novel.

3.2 Related Work

As described in Section 1.3, before online social networks existed, sociologists already developed methods for *social network analysis* which use networks to model social relations and to explain the influence of these relations. However, the scale and complexity of social networks, in terms of popularity and geography, have never reached the same size as those of online social networks. Because of the popularity of OSNs and the diversity of online communities, it is not an easy task to mine them. Many OSNs provide services for members to create or join *groups* and to list *interests* in their profiles. Sometimes it is easy to identify communities formed in this way by simply listing the members of a group and checking the interests listed in all members' profiles. However, these explicit communities do not take members' activities into account and relay on the assumption that the profiles of members are accurate and reliable; this assumption might not be true because a member who joins a group may not necessarily be active in the group, or a member may list too many or too few interests in their profile. Furthermore, not all online communities have explicit groups: Thus, mining implicit communities based on interactions among members is an important task when studying OSNs.

Another important feature that most OSNs provide is a *friend* list for each member. Some recent research on mining online communities analyze the friend link structure among OSN members, and model an OSN as a *friendship graph* [47, 48, 57, 71], in which a node represents a member and a link represents the fact that a member lists another member as a friend. Although the friendship graph provides information about which members know each other, it does not reflect the shared interests and the interactions among these members [72, 75]. The friendship model also implies a simple relation between members, which might be much more complex in reality [12]: Two members may become "friends" in an OSN if, for example, they are colleagues, they have common friends, they share an interest, and so on.

Some researchers have been aware of the drawbacks of friendship graphs and have proposed methods to mine online communities by taking members' interactions into account. In [75], a graph based on members' activities on Facebook was introduced; however, the fact that the subject of the interaction was not captured when building the graph makes it only a general representation of an OSN's social activities, and not appropriate for identifying communities related to different subjects. Methods involving both content and linkage analysis were employed in [17] to locate hate groups in blogospheres; however, modeling *group co-memberships* and *subscriptions* as the links in the graph, and using *group descriptions* as the only input for content analysis, ignores OSN members' real activities: Posting and commenting. In [62, 86] different methods were proposed to discover groups inside explicit communities. Such approaches cannot be applied to general OSNs to mine implicit communities because the structure of a general OSN is much more complex than that of an explicit community. For example, when an expertise forum was studied in [86], a simple statistical method (in which members were asked questions and the more questions an individual answered, the more expert the individual was assumed to be) outperformed all other "advanced" link analysis methods; this situation does not hold when dealing with a general OSN.

3.3 Community Mining

A member of an OSN may interact with other members on a variety of subjects. For each subject, the same member may play different roles. For example, a member may be very active in some topics, may occasionally join discussions in others, but may not take part in some other topics at all. The friendship graph does not reflect the degrees of involvement of a member on different subjects. We introduce the concept of *subject-driven community*, which is a community on a given topic, to accurately model people's behavior on OSNs. To our knowledge, no research has been published which focuses on subject-driven communities, although it is usually suggested that communities are based on shared interests, i.e., subject-

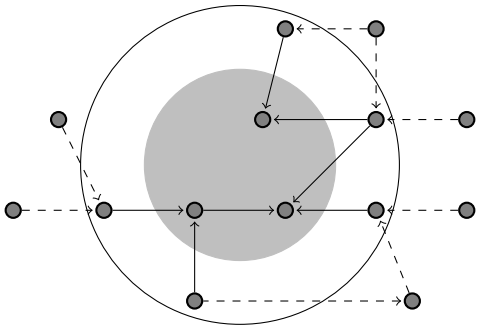


Figure 3.1: Initialization and expansion of the interaction graph.

related.

In this section, we introduce interaction graphs to model communities, and apply the HITS algorithm to locate the core, or most active members of a community.

3.3.1 Interaction Graph

As the friendship graph does not capture the interactions among OSN members, we formally define the *interaction graph* on a given subject as follows.

Interaction Graph: Given a subject t and an OSN, an interaction graph with respect to t is defined as a directed graph $G = (V, E)$, where V is a collection of nodes in which each node u denotes a member of the OSN who posts blogs, or comments on blogs relevant to t; E is a collection of links, where link (u, v) denotes that the member represented by u comments on the blogs of the member v. The weight of (u, v) represents the number of comments that u made on v's blogs, as described below.

It is worth to mention that 1) the term *blog* refers to the content a member posts on their space, which could be, for example, an article, a picture, or a video clip, as mentioned earlier; 2) the blog must be relevant to the subject t; and 3) the interaction graph is a weighted directed graph, which captures the posting and commenting activities of members with respect to the subject t.

We construct an interaction graph by using the following procedure.

Step 1: Given a subject t, we first look for blogs related to this subject, for example, by employing a search engine or the search service provided by the OSN and using the subject as the query. These blogs form the starting set for the interaction graph. A node is created for each member who posts or comments on these blogs and we add a link (u, v) if u comments on v's blog. We call this step *initialization*.

Step 2: For each node u in the graph, we examine all blogs that the member represented by u has posted. If a blog b is relevant to the subject t, we add to the graph nodes for all members who comment on b and we also add links representing the comments. Then, we update the weights of the links and repeat Step 2. We call this step *expansion*.

Step 3: When no more nodes are added in Step 2, we output the constructed graph as the interaction graph.

Noting that deciding whether a blog is relevant to a subject belongs to the area of data mining, we may employ text classification techniques to accomplish this task.

Figure 3.1 shows Steps 1 and 2 of the above procedure. Nodes represent members and arrows represent comments. In Step 1 a search engine returns a collection of blogs related to a given subject. The members who have written these blogs are represented by the nodes inside the shadowed area in the figure; these nodes along with the nodes for members who comment on these blogs compose the starting set, shown inside the outer circle in the figure. By following the comments that other members make on any blogs posted by the starting set (dashed arrows in the figure), more nodes are added to the graph.

In Step 2, the weight of a link is assigned as follows. Assume that the member represented by node u comments on the blogs posted by v; for each blog b posted by v we increment the weight of link (u, v) by 1 if u posts exactly one comment on b, or by 2 if u posts more than one comment on this blog. We adopted this weighting scheme to avoid assigning too much weight to one link when members u and v have a long conversation by commenting on each other's comments on one blog.

It is worth to mention that the initialization and expansion of the interaction graph are similar to the procedure of expanding the root set to a base set in the HITS algorithm. In fact we modified the procedure to keep the interaction graph subject-centric by applying text classification techniques.

3.3.2 Link Analysis

As described in Section 1.1.1, the HITS algorithm claims that there are two types of valuable pages on the Web: *Authority* pages, which are those pages that contain highly relevant information about a topic, and *hub* pages, which are those pages that contain links to the *authority* pages on a topic. In our context a good authority is a member of an OSN whose blogs are considered influential or highly relevant on certain subject by other members (i.e., these blogs receive a large number of comments), and a good hub is a member who comments a lot on blogs that are highly relevant on a subject.

The original HITS algorithm was designed to work on an unweighted graph, but the interaction graph we have constructed in Section 3.3.1 is a weighted one. We propose below a modified version of HITS that works on weighted graphs.

1. Consider a weighted directed graph $G = (V, E)$ represented by its adjacent matrix $\mathbf{W}$, where entry $\mathbf{W}[i, j]$ is the weight of the link (i, j) (which denotes "member i comments on member j's blogs"), or zero if there is no such a link.

2. Let $\vec{a}$ denote the authority vector (a vector giving the authority score of each node of G), and $\vec{h}$ denote the hub vector (a vector giving the hub score of each node of G). We initialized $\vec{h}$ so all its components are 1. We update $\vec{a}$ and $\vec{h}$ iteratively by using the following equations until convergence is achieved:

$$\vec{a} = \mathbf{W}\vec{h} \tag{3.1}$$

$$\vec{h} = \mathbf{W}^T\vec{a} \tag{3.2}$$

where $\mathbf{W}^T$ is the transpose of matrix $\mathbf{W}$.

It has been proven [33] that, given that the entries of W are non-negative real numbers, with appropriate renormalization this iterative process converges after a polynomial number iterations. We renormalize $\vec{a}$ and $\vec{h}$ after each iteration so that the sum of components of each vector is equal to 1.

Note that if the graph is unweighted and the matrix $\mathbf{W}$ contains only binary values 1 and 0 indicating the presence or absence of a link, then Equation (3.1) and Equation (3.2) are equivalent to Equations (1.1) and (1.2) in Section 1.1.1, respectively.

After converging, $\vec{a}[i]$ and $\vec{h}[i]$, the ith entries in $\vec{a}$ and $\vec{h}$, are the authority and hub score of the ith node in the interaction graph, respectively. We consider those members who have high authority and/or hub scores as the core of the community with respect to the given subject.

3.4 Experimental Results

3.4.1 Data Set

LiveJournal [39] is one of the largest OSNs in the world and one of the most popular data sets when studying OSNs [57, 62, 84]. We chose LiveJournal as our experimental data set not only because of its availability, but also because of its nature as a blog service that contains rich features for subject distillation (compared to other OSNs like Facebook and Flickr). LiveJournal allows members to create, join and leave explicit communities freely, which provides us with a base for comparison with the implicit communities that we are interested in discovering.

We implemented four crawlers to perform data collection: One to crawl the friendship network, one to fetch each member's interest list, one to crawl the explicit communities, and one to fetch and parse blogs and comments. The blogs and comments are extracted from LiveJournal's server and stored in HTML format. Obviously our crawlers have to obey LiveJournal's bot policy, so that data not available to the public or protected from our crawlers are not collected. Also our crawlers ignore the blogs and comments not written in English.

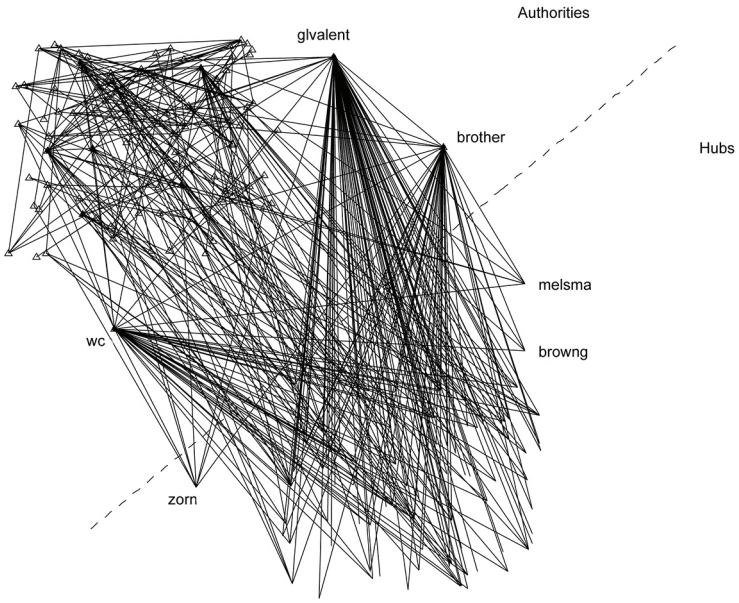


Figure 3.2: The core of the interaction graph representing the "astronomy" community in LiveJournal; the core is composed by the top 100 authorities and hubs. Authorities, marked as triangles, and hubs, are separated by the dashed line. The names of the top 3 authorities and hubs are also shown.

Initially, we used 200 random LiveJournal members as the seeds. The friendship crawler performed a breadth first search on LiveJournal and it fetched the information of more than 3 million members. The posts crawler fetched all blogs and comments that these members posted in the year 2010; in total we have collected more than 11 million blogs and 49 million comments. All collected data were stored locally in a MySQL database. The scheme of the database is given in Appendix B.

When constructing the interaction graph, we used the binary SVM classifier and the training set we have described in Section 2.5.4 to determine whether a blog is relevant to a given subject.

3.4.2 Results and Discussion

To test our approach for community discovery we first needed to select a suitable subject. We read blogs and comments from a number of LiveJournal members and discovered that a fair number of them were interested in astronomy. Hence, we decided to use "astronomy" as the

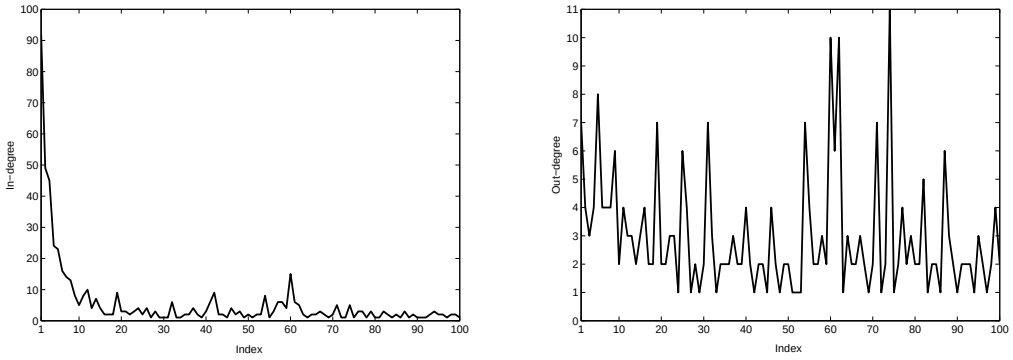
User ID	Authority Score	User ID	Hub Score
glvalentine	0.25279300	melsmarsh	0.00951980
brotherguy	0.19863059	browngirl	0.00852836
wcg	0.16937396	zornhau	0.00847445
dr_nebula	0.08875633	janetmiles	0.00841167
davidkevin	0.04641591	ysabetwordsmith	0.00799496
invaderxan	0.04329613	pameladean	0.00762778
beamjockey	0.03607167	dd_b	0.00684412
bobdeloyd	0.02450088	hhw	0.00655707
dewline	0.02252651	alankria	0.00650713
11011110	0.01648967	luscious_purple	0.00644369

Table 3.1: Members with top ten authority and hub scores in the astronomy community.

first subject for our study.

We used Google Web Search to obtain the initial set of members in the astronomy community by using "astronomy" as the query. Google returned a collection C of documents containing blogs from LiveJournal and we used the SVM classifier to select from C a subset C' of documents highly relevant to astronomy. The starting set of our interaction graph included those members who posted or commented on the documents in C'; we then expanded the interaction graph as described in Section 3.3.1. A Perl program selected $4,978$ blogs and $13,493$ comments from the database, and constructed a $4,739$-node interaction graph. Finally we computed the authority and hub scores for each node using Equation (3.1) and (3.2) until convergence was reached.

Figure 3.2 illustrates the core of the interaction graph, which consists of the members with the top 100 authority and hub scores (henceforth these members are called authorities and hubs, respectively). One point to note is that the top authorities have a large number of links from the top hubs, and the top hubs point to many top authorities, which closely follows the definition of authorities and hubs. Another interesting observation is that there are few links between hubs, and the interconnection between authorities is not as dense as the interconnection between hubs and authorities, which implies that some members mainly limit themselves to posting blogs while some other members mainly read and comment on others' blogs, and thus the



(a) In-degrees of top 100 authorities. (b) Out-degrees of top 100 hubs.

Figure 3.3: In and out-degrees of the top 100 authorities and hubs. Authorities and hubs are indexed in decreasing order of authority and hub score.

authority/hub model fits this pattern of behavior appropriately.

Table 3.1 lists the members with the top 10 authority and hub scores. We notice that the authority scores decrease rapidly, while the hub scores do not vary as much. This tendency can be observed in Figure 3.2 as well: A few authorities (the top 3) have numerous links, while the others reside in a relatively sparse portion of the graph; on the other hand, all hubs have a similar number of links. The in-degrees of the top 100 authorities (the in-degree of an authority a is the number of members who comment on a's blogs) and out-degrees of the top 100 hubs (the out-degree of a hub is the number of authorities on whose blogs the hub comments), are shown in Figure 3.3, and they also reflect the same fact: That the number of influential authorities is small, but they are known by many hubs.

The average degree of the interaction graph for the "astronomy community" is 2.7, while previous research [82, 84] and our collected data show that the average degree of a node in LiveJournal's friendship graph is around 15. The considerable difference between these two degrees implies that members do not interact with all their fiends on a particular subject; this difference also indicates that the friendship graph may not be suitable to capture the interactions among members on a given subject.

We carefully examined the explicit astronomy community in LiveJournal[1], consisting of

[1] http://community.livejournal.com/astronomy/

2,425 members, and found out that only 44 of these members appear in the implicit community that we discovered; furthermore, only 7 of them are among the top 100 authorities and hubs in the interaction graph. In addition, we analyzed the activity of this explicit astronomy community and discovered that only 186 members were active in the community in 2010 and they posted only 42 blogs and 318 comments, while the implicit astronomy community discovered by our approach consists of 4,739 members who posted 4,978 blogs and 13,493 comments relevant to astronomy in 2010.

We also checked the profiles of all 4,739 members in the interaction graph. Surprisingly only 369 of them list "astronomy" as one of their interests, although the total number of members in LiveJournal who claim "astronomy" as one of their interests is 8,624.

Besides "astronomy", we also used our approach to identify the LiveJournal community related to "martial arts". There are 6,090 members in LiveJournal who list martial arts as one of their interests, and, furthermore, there are 4 explicit LiveJournal communities related to martial arts: "_martial_arts", "martial artists", "fighting martial arts" and "teaching martial arts". We found out that only 49 members in these communities were active in discussions related to "martial arts" and they posted only 26 blogs and 105 comments in 2010. In contrast, our approach discovered an implicit "martial arts" community with 758 members who posted 297 blogs and 1,020 comments in 2010.

In our experimental results we noticed more than one order of magnitude difference between the activity of the implicit communities and LiveJournal's explicit communities, and about one order of magnitude difference between the number of members who list a subject as an interest and the number of members who are actually active in discussions related to that subject. This data strongly suggests that LiveJournal's explicit communities and user's profiles are not really useful to discover subject-related communities.

3.5 Conclusion

We have studied two properties of online communities on online social networks: Shared interests and user interactions. We then introduced a novel concept in community mining: Subject-driven communities, based on the premise that a community should be defined by the interactions among members, instead of friendship relations. We proposed the interaction graph to model a community, and we proposed an approach to keep a community subject-relevant by using text classification technology. We also designed a modification to the HITS algorithm to locate the core of a online community.

Our initial experiments on LiveJournal illustrate the inadequacy of explicit communities and members' profiles to mine online communities. Our results show more than one order of magnitude difference between the activity of the implicit communities discovered by our approach and LiveJournal's explicit communities. It is worth to mention that this difference supports what we claimed in Section 3.2: A member who joins a group may not necessarily be active in the group, and not all online communities have explicit groups. Our analysis of members' profiles shows a small overlap between members who list interests on a subject and members who are active in discussions related to that subject; this supports our second claim in Section 3.2 that a member who lists an interest may not interact with other members on that topic, and members who are active on a topic may not list the topic as one of their interests. We believe that our approach of using interaction graphs and the authority/hub model is effective in discovering communities.

In the course of this research, we have encountered several interesting issues which might be worth working on in the future.

1. The procedure of expanding the starting set assumes that all members who are interested in a given subject are reachable from the starting set. If there are other groups of members who are also interested in the subject but not directly connected with the community discovered by the expanding procedure, our approach will ignore them. To identify those

groups, i.e., isolated small communities, either a new approach to select the starting set, or a global analysis of all blogs made by members, is required.

2. In our research we use the HITS algorithm to locate the core members of a community. The assumption behind HITS is that authorities and hubs are two type of members, and authorities tend to post a lot while hubs tend to comments a lot. In our experiments, this assumption seems reasonable. However, we know that the PageRank algorithm does not distinguish authorities and hubs. Some interesting questions here are whether HITS and PageRank will discover the same core portion of a community; if the answer is no, in what aspect they differ from each ohter, and which core portion represents the community better.

Chapter 4

Conclusion

The Internet and the World Wide Web have changed many aspects of modern life, especially the way in which information is distributed and retrieved, and the way in which people communicate with each other. Nowadays, Web search engines are the most prevalent tools for finding information in the Web. On the other hand, online social networks are among the most popular services that take advantage of the widespread availability of the Internet. Identifying communities in online social networks gives insights into the structure of these networks and the social interactions among OSN members. In this thesis, we propose methods to improve the effectiveness of Web search engines and to identify communities in online social networks by combining techniques from data mining, network analysis and semantic analysis.

Observing that Web search engines select Web pages according to the occurrences of the query words in them, but without taking users' intentions into account, we propose Query Extension to include in the result set more pages relevant to the query, and Query Categorization to exclude pages irrelevant to the user's intention from the result set. Query Extension is achieved by expanding the result set so that it consists of not only the Web pages containing the original query words, but also the pages containing the synonyms and inflected forms of the original query words. The synonyms and inflected forms are compiled and kept in an inflection dictionary.

Given a query, a Web search engine always returns the same result set regardless of the particular meaning that a user has in mind when issuing a query. We propose Query Categorization to present the user only those Web pages relevant to the intended meaning for the query, as specified by the user. A category for a query is suggested by applying word classification techniques on the query and confirmed by the user. Text classification techniques are then applied on the result set to filter out the Web pages irrelevant to the chosen category.

We implemented our Query Extension and Query Categorization approaches and compared our results with those of a commercial Web search engine, namely Google. The comparison shows that our approaches are effective in improving the performance of search engines by retrieving more relevant Web pages through extending the query words, and by refining the result set to meet the user's intention more precisely.

In practice, Query Extension is able to improve the results of a Web search engines, as shown by our experiments. Recently, some commercial Web search engines have started to include in their result sets Web pages containing words related to the query words like synonyms, however the algorithms and implementation details of these search engines have not been publicly disclosed.

We believe that users will benefit if Query Categorization is applied by Web search engines. In our implementation the classification of Web pages is done before returning the result set to the user; commercial search engines could add category labels to Web pages during indexing, and leave Query Categorization as an optional feature. This way the users could enable Query Categorization only when necessary to refine the results, without compromising the fast-response of current Web search engines.

Our efforts in this thesis suggest two future research directions for information retrieval on the Web; one is how to keep the search user-centric, and the other is how to integrate text classification and Web search together. The history of Web search engines shows a switch in the focus of the searching procedure: Early text-indexing search engines were publisher-centric, as they merely focused on the content of Web pages; modern search engines are linkage-centric,

as they rank pages based on hyperlinks. The purpose of Query Extension and Query Categorization is to make Web search user-centric, i.e., to retrieve Web documents based on users' intentions not on query words.

On the other hand, even though the development of text classification and Web search techniques has taken place at the same time, the effort of integrating them has seldom been made. We propose Query Categorization to incorporate these two techniques and combine their benefits, i.e., a user can access the result set containing numerous Web pages in a short time due to the fast response of Web search techniques, but needs not to browse redundant pages irrelevant to the user's intention because of the systematic orderliness that text classification techniques deliver.

As online social networks have become one of the most popular platforms for social activities, computer scientists and sociologists are now facing the task of better understanding the interactions among OSN members and the social structure underneath them; mining subject-driven communities in online social networks gives insights into these interactions. Identifying these communities also provides valuable and reliable information which could lead to better knowledge of how information flows in online social networks.

Most recent approaches to identify communities in online social networks focus on the friendship graph, and ignore the interactions among members. We propose the notions of subject-driven communities which reflect a group of members who have not only shared interests but also social interactions. We consider posting and commenting, the most common activities in OSNs, as the core interactions among members. We propose to model posting and commenting activities using an interaction graph, and to keep the posts and comments related to a topic by filtering them using text classification techniques. We also propose a modified version of the network analysis algorithm HITS to identify the core of a community.

We collected data from an online social network and applied our algorithm on this data to identify communities in the OSN. Our initial experiments show that the explicitly posted communities and the publicly declared interests in a user's profile do not really help to identify

subject-driven communities.

Our contribution to community mining in online social networks lies on the introduction of the concept of subject-driven communities based on interactions. Sociologists use social network analysis to explain the influence of social relations; in online social networks, multiple relations among their members exist and thus deserve careful study. We point out several possible directions for further research in this area.

We mentioned earlier that the friendship graph was not a suitable model for the task of identifying communities in OSNs, and we proposed the interaction graph for this task. However, we believe that the interaction graph and the friendship graph are not irrelevant to each other. For example, a member of a community may influence some of her/his friends to join this community, and members in the same community may become friends after some interactions with each other. In this case, relationships in one graph are able to influence the structure of another graph. On the other hand, an OSN member who is active in different communities will appear in different interaction graphs; this type of members may act as bridges among these communities and these communities may interact with other communities through these bridges. Studying the relationship and interactions among the friendship graph and interaction graphs (with respect to different subjects but connected by one or more members) might lead to a better understanding of the structures and social interactions in OSNs.

OSNs evolve over time constantly, as members may join and leave OSNs at will, and may be active or idle in different periods of time. Tracing the evolution of communities in OSNs at different stages might provide an insight into how the structures and interactions of OSNs and, OSNs themselves, evolve. For example, studying how a community emerges and how other members become aware of this community may uncover the process how members look for and start to interact with others who have shared interests. We believe that our approach to identify subject-driven communities based on member interactions is a good start point to explore the properties and the evolutions of online social networks.

Bibliography

[1] R. Akbani, S. Kwek, and N. Japkowicz, "Applying support vector machines to imbalanced datasets," *Machine Learning*, pp. 39–50, 2004.

[2] "altavista," http://www.altavista.com/. Last accessed: Jul. 12th, 2011.

[3] G. Attardi, S. Di Marco, and D. Salvi, "Categorisation by context," *Journal of Universal Computer Science*, vol. 4, no. 9, pp. 719–736, 1998.

[4] G. Attardi, A. Gulli, and F. Sebastiani, "Automatic web page categorization by link and context analysis," in *Proceedings of European Symposium on Telematics, Hypermedia and Artificial Intelligence*, 1999, pp. 105–119.

[5] K. Bharat and M. Henzinger, "Improved algorithms for topic distillation in a hyperlinked environment," in *Proceedings of the 21st ACM International Conference on Research and Development in Information Retrieval*, 1998, pp. 104–111.

[6] "Bing," http://www.bing.com. Last accessed: Jul. 12th, 2011.

[7] R. Blood, "Weblogs: A history and perspective," http://www.rebeccablood.net/essays/weblog_history.html. Last accessed: Jul. 12th, 2011.

[8] P. Bonacich, "Power and centrality: A family of measures," *American Journal of Sociology*, vol. 92, no. 5, pp. 1170–1182, 1987.

[9] S. Borgatti, A. Mehra, D. Brass, and G. Labianca, "Network analysis in the social sciences," *Science*, vol. 323, no. 5916, pp. 892–895, 2009.

[10] A. Borodin, G. Roberts, J. Rosenthal, and P. Tsaparas, "Finding authorities and hubs from link structures on the world wide web," in *Proceedings of the 10th International Conference on World Wide Web*, 2001, pp. 415–429.

[11] S. Brin and L. Page, "The anatomy of a large-scale hypertextual Web search engine," *Computer Networks and ISDN Systems*, vol. 30, no. 1–7, pp. 107–117, 1998.

[12] D. Cai, Z. Shao, X. He, X. Yan, and J. Han, "Mining hidden community in heterogeneous social networks," in *Proceedings of the 3rd International Workshop on Link Discovery*, 2005, pp. 58–65.

[13] P. R. Center, "The state of the news media 2011." http://stateofthemedia.org/2011/overview-2/key-findings/.

[14] S. Chakrabarti, B. Dom, and P. Indyk, "Enhanced hypertext categorization using hyperlinks," in *Proceedings of ACM International Conference on Management of Data*, 1998, pp. 307–318.

[15] S. Chakrabarti, B. Dom, P. Raghavan, S. Rajagopalan, D. Gibson, and J. Kleinberg, "Automatic resource compilation by analyzing hyperlink structure and associated text," *Computer Networks and ISDN Systems*, vol. 30, no. 1-7, pp. 65–74, 1998.

[16] S. Chakrabarti, M. Joshi, and V. Tawde, "Enhanced topic distillation using text, markup tags, and hyperlinks," in *Proceedings of Research and Development in Information Retrieval*, 2001, pp. 208–216.

[17] M. Chau and J. Xu, "Mining communities and their relationships in blogs: A study of online hate groups," *International Journal of Human-Computer Studies*, vol. 65, no. 1, pp. 57–70, 2007.

[18] H. Chen and S. Dumais, "Bringing order to the web: automatically categorizing search results," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, 2000, pp. 145–152.

[19] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[20] H. Dougherty, "Facebook.com generates nearly 1 in 4 page views in the us." http://weblogs.hitwise.com/heather-dougherty/2010/11/facebookcom_generates_nearly_1_1.html.

[21] O. Drori and N. Alon, "Using document classification for displaying search results lists," *Journal of Information Science*, vol. 29, no. 2, pp. 97–106, 2003.

[22] S. Dumais, E. Cutrell, and H. Chen, "Optimizing search by showing results in context," in *Proceedings of SIGCHI Conference on Human Factors in Computing Systems*, 2001, pp. 277–284.

[23] "Excite," http://www.excite.com. Last accessed: Jul. 12th, 2011.

[24] Http://www.facebook.com. Last accessed: Jul. 12th, 2011.

[25] C. Fellbaum, *WordNet: An electronic lexical database*. The MIT press, 1998.

[26] P. Ferragina and A. Gulli, "A personalized search engine based on Web-snippet hierarchical clustering," *Software: Practice and Experience*, vol. 38, no. 2, pp. 189–225, 2008.

[27] W. B. Frakes and R. Baeza-Yates, *Information retrieval: data structures and algorithms*. Prentice-Hall, Inc., 1992.

[28] J. Fürnkranz, "Proceedings of exploiting structural information for text classification on the WWW," in *Proceedings of Intelligent Data Analysis*, 1999, pp. 487–498.

[29] G. Gromov, "The Roads and Crossroads of Internet History," http://www.netvalley.com/intval_intr.html. Last accessed: Jul. 12th, 2011.

[30] S. Gauch, D. Ravindran, and A. Chandramouli, "Keyconcept: Conceptual search and pruning exploiting concept relationships," *Journal of Intelligent Systems*, vol. 19, no. 3, pp. 265–288, 2010.

[31] "Google directory," http://www.google.com/dirhp. Last accessed: Jul. 12th, 2011.

[32] E. Glover, G. Flake, S. Lawrence, W. Birmingham, A. Kruger, C. Giles, and D. Pennock, "Improving category specific web search by learning query modifications," in *Proceedings of Symposium on Applications and the Internet*, 2001, pp. 23–32.

[33] G. Golub and C. Van Loan, *Matrix Computations*. Johns Hopkins Univ Pr, 1996.

[34] D. Goodwin, "May 2011 Search Engine Market Share from comScore, Compete, Hitwise," http://searchenginewatch.com/article/2080003/May-2011-Search-Engine-Market-Share-from-comScore-Compete-Hitwise. Last accessed: Jul. 12th, 2011.

[35] "Google," http://www.google.com. Last accessed: Jul. 12th, 2011.

[36] J. Greenberg, "Automatic query expansion via lexical–semantic relationships," *Journal of the American Society for Information Science and Technology*, vol. 52, no. 5, pp. 402–415, 2001.

[37] J. Hughes, "Automatically acquiring a classification of words," Ph.D. dissertation, School of Computing, University of Leeds, 1994.

[38] J. Hughes and E. Atwell, "A methodical approach to word class formation using automatic evaluation," in *Proceedings of AISB workshop on Computational Linguistics for Speech and Handwriting Recognition*, 1994, pp. 41–48.

[39] "Livejournal," http://www.livejournal.com/. Last Accessed: Jul. 12th, 2011.

[40] N. Japkowicz, "The class imbalance problem: Significance and strategies," in *Proceedings of International Conference on Artificial Intelligence*, vol. 1, 2000, pp. 111–117.

[41] A. Java, X. Song, T. Finin, and B. Tseng, "Why we twitter: understanding microblogging usage and communities," in *Proceedings of the 9th WebKDD and 1st SNA-KDD Workshop on Web Mining and Network Analysis*, 2007, pp. 56–65.

[42] T. Joachims, "Text categorization with support vector machines: learning with many relevant features," in *Proceedings of the 10th European Conference on Machine Learning*, 1998, pp. 137–142.

[43] J. Kleinberg, "Authoritative sources in a hyperlinked environment," in *Proceedings of the 9th ACM-SIAM Symposium on Discrete Algorithms*, 1998, pp. 668–677.

[44] B. Koester, "Conceptual knowledge retrieval with FooCA: Improving web search engine results with contexts and concept hierarchies," *Advances in Data Mining. Applications in Medicine, Web Mining, Marketing, Image and Signal Mining*, vol. 4065, pp. 176–190, 2006.

[45] M. Kubat and S. Matwin, "Addressing the curse of imbalanced training sets: one-sided selection," in *Proceedings of Machine Learning-International Workshop Then Conference*, 1997, pp. 179–186.

[46] B. Kules, J. Kustanowitz, and B. Shneiderman, "Categorizing web search results into meaningful and stable categories using fast-feature techniques," in *Proceedings of the 6th ACM/IEEE-CS Joint Conference on Digital Libraries*, 2006, pp. 210–219.

[47] R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins, "Trawling the Web for emerging cyber-communities," *Computer Networks*, vol. 31, no. 11-16, pp. 1481–1493, 1999.

[48] K. Lawrence, "Internet-Based Community Networks: Finding the Social in Social Networks," *Computing with Social Trust*, vol. 1, pp. 313–331, 2009.

[49] S. Lawrence and C. L. Giles, "Accessibility of information on the web," *Nature*, vol. 400, pp. 107–109, July 1999.

[50] F. Liu, C. Yu, and W. Meng, "Personalized web search by mapping user queries to categories," in *Proceedings of the 11th International Conference on Information and Knowledge Management*, 2002, pp. 558–565.

[51] H. Liu, H. Lieberman, and T. Selker, "Goose: a goal-oriented search engine with commonsense," in *Proceedings of Adaptive Hypermedia and Adaptive Web-Based Systems*, 2006, pp. 253–263.

[52] T. Liu, Y. Yang, H. Wan, H. Zeng, Z. Chen, and W. Ma, "Support vector machines classification with a very large-scale taxonomy," *ACM SIGKDD Explorations Newsletter*, vol. 7, no. 1, pp. 36–43, 2005.

[53] "Lycos," http://www.lycos.com. Last accessed: Jul. 12th, 2011.

[54] M. Maloof, "Learning when data sets are imbalanced and when costs are unequal and unknown," in *ICML-2003 Workshop on Learning from Imbalanced Data Sets II*, 2003.

[55] Maurice de Kunder, "The size of the World Wide Web ," http://www.worldwidewebsize.com/. Last accessed: Jul. 12th, 2011.

[56] A. McCallum, "Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering," 1996, http://www.cs.cmu.edu/ mccallum/bow.

[57] A. Mislove, M. Marcon, K. Gummadi, P. Druschel, and B. Bhattacharjee, "Measurement and analysis of online social networks," in *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement*, 2007, pp. 29–42.

[58] D. Moldovan and R. Mihalcea, "Using WordNet and lexical operators to improve Internet searches," *Internet Computing*, vol. 4, no. 1, pp. 34–43, 2000.

[59] "Merriam-Webster Online Thesaurus," http://www.m-w.com. Last accessed: Jul. 12th, 2011.

[60] "Mysql," http://www.mysql.com. Last accessed: Jul. 12th, 2011.

[61] B. Narayan, C. Murthy, and S. Pal, "Topic continuity for web document categorization and ranking," in *Proceedings of IEEE/WIC International Conference on Web Intelligence*, 2003, pp. 310–315.

[62] T. Nguyen, D. Phung, B. Adams, T. Tran, and S. Venkatesh, "Hyper-community detection in the blogosphere," in *Proceedings of the 2nd ACM SIGMM Workshop on Social Media*, 2010, pp. 21–26.

[63] M. NikRavesh, "Fuzzy conceptual-based search engine using conceptual semantic indexing," in *Proceedings of the Annual Meeting of the North America of the Fuzzy Information Processing Society*, 2002, pp. 146–151.

[64] "Open Directory Project," http://www.dmoz.org. Last accessed: Jul. 12th, 2011.

[65] J. Preece, *Online communities: Designing usability, supporting sociability*. John Wiley Chichester, UK, 2000.

[66] A. Roberts, "Automatic acquisition of word classification using distributional analysis of content words with respect to function words," School of Computing, University of Leeds, Tech. Rep., 2002.

[67] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Information processing & management*, vol. 24, no. 5, pp. 513–523, 1988.

[68] H. Schütze, D. Hull, and J. Pedersen, "A comparison of classifiers and document representations for the routing problem," in *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1995, pp. 229–237.

[69] P. Senellart and V. Blondel, "Automatic discovery of similarwords," *Survey of Text Mining II*, pp. 25–44, 2008.

[70] "Search Engine Watch," http://www.searchenginewatch.com. Last accessed: Jul. 12th, 2011.

[71] C. Tantipathananandh, T. Berger-Wolf, and D. Kempe, "A framework for community identification in dynamic social networks," in *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2007, pp. 717–726.

[72] M. Valafar, R. Rejaie, and W. Willinger, "Beyond friendship graphs: a study of user interactions in Flickr," in *Proceedings of the 2nd ACM Workshop on Online Social Networks*, 2009, pp. 25–30.

[73] "The world wide web consortium (w3c)," http://www.w3.org. Last accessed: Jul. 12th, 2011.

[74] Y. Wang, J. Hodges, and B. Tang, "Classification of web documents using a Naïve Bayes method," in *Proceedings of the 15th IEEE International Conference on Tools with Artificial Intelligence*, 2003, pp. 560–564.

[75] C. Wilson, B. Boe, A. Sala, K. Puttaswamy, and B. Zhao, "User interactions in social networks and their implications," in *Proceedings of the 4th ACM European Conference on Computer Systems*, 2009, pp. 205–218.

[76] L. Xin-fu, Y. Yan, and Y. Peng, "The method of text categorization on imbalanced datasets," in *Proceedings of ICCSN'09: International Conference on Communication Software and Networks*, 2009, pp. 650–653.

[77] "Yahoo!'s U.S. audience surpasses 25 million, outpacing leading broadcast and print media," http://docs.yahoo.com/docs/pr/release131.html. Last accessed: Jul. 12th, 2011.

[78] Y. Yang, "An evaluation of statistical approaches to text categorization," *Information Retrieval*, vol. 1, no. 1, pp. 69–90, 1999.

[79] Y. Yang and J. Pedersen, "A comparative study on feature selection in text categorization," in *Proceedings of the 14th International Conference on Machine Learning*, 1997, pp. 412–420.

[80] Y. Yang, J. Zhang, and B. Kisiel, "A scalability analysis of classifiers in text categorization," in *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2003, pp. 96–103.

[81] "Yahoo! directory," http://dir.yahoo.com. Last accessed: Jul. 12th, 2011.

[82] S. Ye, J. Lang, and F. Wu, "Crawling online social graphs," in *Proceedings of the 12th International Asia-Pacific Web Conference*, 2010, pp. 236–242.

[83] "Yahoo! search," http://search.yahoo.com. Last accessed: Jul. 12th, 2011.

[84] P. Zakharov, "Diffusion approach for community discovering within the complex networks: LiveJournal study," *Physica A: Statistical Mechanics and its Applications*, vol. 378, no. 2, pp. 550–560, 2007.

[85] D. Zhang and Y. Dong, "Semantic, Hierarchical, Online Clustering of Web Search Results," in *Proceedings of the 6th Asia-Pacific Web Conference of Advanced Web Technologies and Applications*, 2004, pp. 69–78.

[86] J. Zhang, M. Ackerman, and L. Adamic, "Expertise networks in online communities: structure and algorithms," in *Proceedings of the 16th International Conference on World Wide Web*, 2007, pp. 221–230.

[87] T. Zhang and F. Oles, "Text categorization based on regularized linear classification methods," *Information Retrieval*, vol. 4, no. 1, pp. 5–31, 2001.

Appendix A

List of Categories used by our Query Categorization Module

We considered several facts when we choosing the categories for our experiments. 1) Each of the 16 top level directories in Google Directory like “Arts” and “Business” covers a broad number of topics and overlaps with each other, so they are not suitable for Query Categorization. 2) We want to have a reasonable number of Web pages in each category to balance the performance and the speed of the classifier, and we wish the total number of categories to be moderate to make it easy for a user to browse them. After extensive experiments we believe that second level directories in Google directory fit our requirements. 3) Due to our own linguistic limitations, we excluded those Web pages belonging to the first level directories “Regional” and “World” because that they mostly contain pages in languages other than English. 4) Some of second level directories contain too few pages and some overlap with others, so when training the classifier, they may mislead it by introducing noise into the statistical model; hence, we decided not to keep these directories.

Bearing the above considerations in mind, we collected and examined Web pages in all second level directories, and decided to keep in the collection 250 Web pages whose PageRanks were greater than 0.5 from each directory. Directories which contain fewer than

150 pages were excluded from the list. Furthermore, we examined each Web page manually to discard undesirable ones, such as Web pages which contain little text, redirected pages, and broken links. We also tested the collection with the Bow classifier by applying cross-validation (as described in Section 1.2.5): Those directories which had low precision (< 0.8) were not kept; misclassified Web pages detected by leave one out cross-validation and examined by us were also discarded. Finally we collected a training set which consists of 64 categories and total 13,179 Web pages, as shown below. The name of each category is followed by the number of Web pages in the category.

1. Arts/Animation : 198
2. Arts/Art_History : 203
3. Arts/Movies : 203
4. Arts/Music : 218
5. Arts/Television : 200
6. Business/Accounting : 190
7. Business/Employment : 214
8. Business/Investing : 208
9. Computers/Artificial_Intelligence : 185
10. Computers/Computer_Science : 169
11. Computers/Graphics : 184
12. Computers/Hardware : 202
13. Computers/Multimedia : 208
14. Computers/Programming : 210

15. Computers/Security : 221
16. Computers/Systems : 213
17. Games/Gambling : 199
18. Games/Video_Games : 209
19. Health/Addictions : 215
20. Health/Alternative : 202
21. Health/Animal : 207
22. Health/Conditions_and_Diseases : 222
23. Health/Mental_Health : 222
24. Health/Nutrition : 158
25. Health/Pharmacy : 206
26. Home/Cooking : 217
27. Home/Gardening : 213
28. News/Colleges_and_Universities : 222
29. News/Media : 216
30. News/Newspapers : 193
31. Recreation/Autos : 210
32. Recreation/Birding : 225
33. Recreation/Outdoors : 216
34. Recreation/Pets : 212

35. Recreation/Roads_and_Highways : 211
36. Reference/Knowledge_Management : 212
37. Science/Astronomy : 205
38. Science/Biology : 202
39. Science/Math : 194
40. Science/Physics : 186
41. Shopping/Clothing : 219
42. Shopping/Food : 203
43. Shopping/Home_and_Garden : 196
44. Shopping/Sports : 204
45. Society/Crime : 206
46. Society/Folklore : 189
47. Society/Genealogy : 189
48. Society/Holidays : 223
49. Society/Military : 197
50. Society/Philosophy : 208
51. Society/Politics : 186
52. Society/Religion_and_Spirituality : 198
53. Sports/Basketball : 184
54. Sports/Cycling : 197

55. Sports/Equestrian : 195

56. Sports/Golf : 227

57. Sports/Martial_Arts : 160

58. Sports/Motorsports : 196

59. Sports/Soccer : 219

60. Sports/Tennis : 223

61. Sports/Track_and_Field : 226

62. Sports/Volleyball : 183

63. Sports/Water_Sports : 199

64. Sports/Winter_Sports : 225

Appendix B

Database Scheme for LiveJournal

The MySQL database consists of 5 tables: COMMENT, GROUPS, INTERESTS, KNOWS, POST and USER. We list their fields, data types and descriptions below.

Field	Type	Description
cid	int(11)	Comment ID
uid	varchar(100)	User ID
pid	int(11)	Post ID, to which the comment is given
content	text	Content of the comment
time	timestamp	When the comment is made
parent	int(11)	Parent comment ID, if applicable

Table B.1: Table COMMENT.

Field	Type	Description
groupname	varchar(100)	Name of the explicit community
uid	varchar(100)	User ID who is a member of the community
description	text	Description of the community

Table B.2: Table GROUPS.
Each entry contains data from an explicit community in LiveJournal.

Field	Type	Description
interest	varchar(200)	Name of the interest
uid	varchar(100)	User ID who lists this interest in the profile
description	text	Description of the interest

Table B.3: Table INTERESTS.
Each entry represents that the user with the given uid has this interest.

Field	Type	Description
usernumber	int(11)	Integer ID of the user, assigned by LiveJournal
uid	varchar(100)	User ID
Name	varchar(100)	User name

Table B.4: Table USER.

Field	Type	Description
uid	varchar(100)	User ID
fid	varchar(100)	Friend ID

Table B.5: Table KNOWS.
Each entry represents the fact that a user with ID uid lists the user with ID fid as a friend.

Field	Type	Description
pid	int(11)	Post ID
uid	varchar(100)	User ID who wrote this post
content	mediumtext	Content of the post
time	timestamp	When the post was written
URL	varchar(100)	URL of the post
title	text	Title of the post

Table B.6: Table POST.

Curriculum Vitae

Name:	Guo Mei
Post-Secondary Education and Degrees:	Xi'an Jiaotong University Xi'an, Shangxi, China 1988 - 1992 B.E. University of Western Ontario London, ON 2003 - 2005 M.Sc. University of Western Ontario London, ON 2006 - 2011 Ph.D.
Related Work Experience:	Research Assistant/Teaching Assistant The University of Western Ontario 2003 - 2011

Publications:

Subject-Driven Community Mining in Online Social Networks, ICOMP2011

Improvements on Existing Search Engines through Categorization, ICOMP2010