Intention-driven textual semantic analysis

Jie Li
University of Wollongong

Intention-driven Textual Semantic Analysis

A thesis submitted in partial fulfillment of the

requirements for the award of the degree

Master of Computer Science

from

UNIVERSITY OF WOLLONGONG

by

Jie Li

School of Computer Science and Software Engineering

July 2008

Dedicated to

This thesis is dedicated to my mom and dad.

Declaration

This is to certify that the work reported in this thesis was done by the author, unless specified otherwise, and that no part of it has been submitted in a thesis to any other university or similar institution.

Jie Li
July 18, 2008

Abstract

The explosion of World Wide Web has brought endless amount of information within our reach. In order to take advantage of this phenomenon, text search becomes a major contemporary research challenge. Due to the nature of the Web, assisting users to find desired information is still a challenging task. In this thesis, we investigate semantic analysis techniques which can facilitate the search process at semantic level. We also study the problem that short queries are less informative and difficult to convey the user's intention into the search service system. We propose a generalized framework to address these issues. We conduct a case study of movie plot search in which a semantic analyzer seamlessly works with a user's intention detector. Our experimental results show the importance and effectiveness of intention detection and semantic analysis techniques.

Acknowledgements

I wish to express my appreciation and gratitude to the thesis supervisors Dr. Jo Abrantes and Prof. Philip Ogunbona for their guidance and encouragement throughout the course of this research work. Dr. Jo Abrantes introduced me into the field of machine learning and offered me the chance to deepen my research. Prof. Philip Ogunbona gave me a lot of constructive suggestions and ideas. Especially they gave me as much freedom as I needed to get my research work done.

I gratefully thank my parents for their support. Without their understanding, encouragement and financial support, this particular goal would not have been achieved.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Today personal computers and smart portable devices are everywhere, which make it easier for people to access various sources of information. Also Internet technology provide people with access to endless and disorganized information. It is true that people cannot keep pace with the explosion of information. Most users are overwhelmed by the amount of data they need to manage while searching for specific information on the Internet.

The importance of developing intuitive methods of searching for information on the Internet is exemplified by the number of search services currently available on the Internet. Prominent search services available on the Internet are web search engines that collect, organize and search for information over the whole public Web. Gulli et al. [35] stated that the estimated size of the whole public Web is at least 11.5 billion pages by the end of January 2005; and Google claims to index more than 8 billion pages which is crawled by the largest search engine on earth. Despite all efforts, search engines still cannot cover the whole public Web. Meanwhile, many professional web

sites provide their own search services for users via their private data collection. More recent development has seen the merger of computer desktop search and Internet search services. Google desktop search can organize and search information resources on a personal computer including files, emails and contacts, images and video, etc. This tool can seamlessly extend its search across the Internet and combine the results of both desktop and Internet search. Peer-to-peer (P2P) networks now provide file sharing tools that can search and serve resources on participating computers. The convenience of mobile technology has prompted the extension of search services into this application domain. Mobile search services now allow users to search web information by using their cellular phones. These search services can be mainly identified as four search paradigms for finding text-based information [21].

Unassisted keyword search where the user enters one or more search terms and the associated search engine returns a list of document summaries. This paradigm is popularized by Google and AltaVista.

Assisted keyword search where the search engine provides the user with some help by expanding the initial query and offering the expansion as a suggestion.

Directory-based search also known as digital library, operates on the basis of having the information space divided into a hierarchy of categories that have been pre-constructed. This paradigm does not start with a posed query, but allows the user to navigate through the hierarchy of categories until specific information is found. A typical example of search engine employing this paradigm is Yahoo.

Query-by-example offers the user the initial candidate selecting from a retrieved list

where they find more relevant documents to their search.

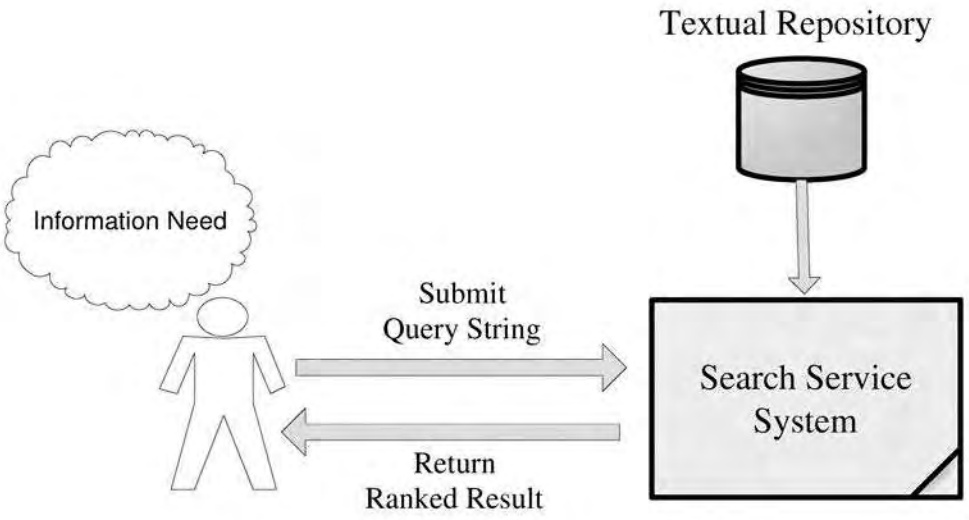


Figure 1.1: The Search Processing

It is important to note that these paradigms are probably never employed singly in a search engine. Rather, a combination of them will be used. The system depicted in Figure 1.1 shows the basic steps of the search process in a search service system. Usually the textual repository can be constructed by the crawler which is an important component of search engines. The search engine can collect and index millions of web pages a day by means of the crawler. Some advanced crawlers can specifically revisit previously indexed pages for the purpose of update. Some selective popular web sites can be checked more often [16]. Because the documents fetched by the crawler are stored on the local textual repository, the documents can be easily accessed for further processing and efficiently reduce the extra network communication during the query processing. The result of the processing is a properly indexed textual repository that can provide fast access to users. The use of semantic document representations can facilitate the query process and provide better results [20]. When performing a query

process, users translate their information need into query strings which is presented to the search service system. In a review of Web searching studies, Jansen et al. [41] found that the average Web query is about two terms. It is evident that such short queries do not possess the ability to capture the users' intention. Furthermore, the system receives the query and yields a set of terms, which are used to compute similarity measure between the query and document representations. According to the similarity measure, the system can return a list of documents that match the user query approximately.

There are three main challenges for traditional keyword-based search services. Firstly, it is difficult to capture the full intention underlying keywords. Secondly, it is difficult to capture users' intention in terms of keywords submitted by the user and thirdly it is difficult to solve the synonymy and polysemy problems. Because queries submitted by users often are too short to effectively reflect the users' information need, the search service system may not be able to capture the information need. Short keyword-based queries cannot generally provide a reliable basis for the purpose of finding best matches. Consider as an example, the keywords *love* and *flower*, a conventional search service treats these two keywords equally important and attempts to match the documents containing these keywords. However, these two words have different meanings depending on the context. The keywords *love* and *flower* may refer to different intentions in different contexts as one would expect when dealing with romantic or plant topics. Also some relevant documents without these keywords may not be matched to the users' information need.

Traditional search services lack certain inference capabilities. The selection of

ranked results is based on the assumption that the queries represent the users' information need. These systems assume that the queries can represent the users' intention. However, the query does not often reflect the users' information need. For instance, if the user types *rose birthday presents* into the search box, the keyword-based search just segments the query string as *rose*, *birthday* and *present*. Then search service systems simply look for documents containing these keywords. In contrast, intelligent search services would attempt to find documents related to *romantic* topics and assign these documents higher ranks. This problem could be resolved if we incorporate user intention detector into a search service system.

Conventional keyword-based search services suffer from synonymy and polysemy problems [20]. A keyword provided by the user may not appear at all in the document, whereas the document may be very much related to the keyword because the same thing can often be described in different ways by using different words in a natural language. For example, the keyword could be *actor* whereas the document may not exactly contain any instance of the word *actor* but contain the word *star* frequently. This is the synonymy problem in which some words have the same meaning. This problem can be solved to some extent by processing the document and replacing words with similar meaning with a chosen specific word. For example, the words *automobile, vehicle, and vehicular* can simply be replaced by the word *car* when they appear in a document. However, this solution may not be always practical because it is difficult to maintain a complete list of these words from a dictionary to facilitate the transformation. On the other hand, it is also possible that the same word may have different meanings in different contexts. For example, the word *star* has different meanings in the context

of *movie star* as compared to *star wars*. This situation is referred to as the polysemy problem in which one word can have multiple meanings in different context. Many query services suffer from these problems, and as a result, many unrelated documents may be in the returned search result. It is easy to see that relevant documents may not be in the result because they do not have the keywords in common with a given query.

This thesis studies the inference capabilities of a search service to detect the user's intention based on the queries. The proposed approach employs machine learning, information retrieval and natural language processing techniques. The information needs can be defined as user intentions which are unclearly encoded in posed queries. The queries vaguely represent users' intentions in the search process. We propose a model to predict user intention and as a case study, an intelligent web-based search service in the context of movie plots is developed. The application utilized a staged and sequential approach to search service modelling, concept search, topic search and content search. This search service can intelligently assist the user in finding the information they seek gradually. Given queries, the search service performs a concept search. At the same time, the query is transferred to the user intention detector. The intention detector is used as an assistant tool to predict the users' information need and reformulate the query. The topic search can be performed through taking into account the recommendations from the user intention detector. If the user found one desirable movie plot in the ranking list, the system allows the user to perform content search.

1.2 Contributions

The main contributions of this thesis are as follows:

- It is well known that semantic analysis techniques can perform well on analyzing informative queries at semantic level, but the semantic analysis technique cannot show satisfactory performance for short queries. The computationally expensive process is not worthy for short queries. The main reason is that the short queries are less informative and inadequately reflect the user's intention. However, short queries are the type of searches used more frequently over the Internet. So we provide a solution that aims at filling the gap between short query and latent semantic analysis. We argue that an intelligent search service system should contain two essential modules: intention detector and semantic analyzer. During user queries, the intention detector attempts to make prediction for the user intention by virtue of user query. Further the search service system can formulate informative query representation. The semantic analyzer can construct semantic document representation which is used to measure similarity with the informative query representation.

- We propose a generalized intention detector framework that can detect users' intention in a domain-specific setting. The framework is based on graphical models, which are suitable for inferencing human search behavior. Also we introduce a simplified intention detector for our application.

- As a case study of movie search, we designed a search service system with inference capabilities which can seamlessly combine a simplified intention detector

with a semantic analyzer.

- The latent semantic analysis technique is applied in our semantic analyzer. It can solve the synonym and polysemy problems effectively. Experiments have been conducted to compare the performance of search among the TF-IDF short query, the LSA short query and the LSA fragment text query.

- The naïve Bayes algorithm is employed in our simplified intention detector. The detector analyzes a users' query in order to predict a predefined state of user intention that the query belong. The system can utilize valuable information provided by the detector for further search processing.

1.3 Organization of the thesis

This thesis is organized as follows.

- In Chapter 2, a literature review of various existing search service models is given. The detailed discussion of semantic document representation and query representation is also presented.

- In Chapter 3, we present a novel framework for detecting user intention. We also introduce a simplified intention detector for implementation in our system.

- In Chapter 4, we present an intelligent search service system for movie plot search, and provide a detailed description of its components.

- In Chapter 5, we report experimental results for demonstrating and verifying the proposed system.

- In Chapter 6, conclusions and highlights of the research outcome are provided. Possible future directions are also enumerated.

Chapter 2

Literature Review

2.1 Introduction

In recent years, computer systems have greatly facilitated the analysis of texts which allow us to experience applications of natural language in new and interesting ways. There are many different applications for textual analysis which include text clustering, text categorization, machine translation, information extraction, text summarization, text search etc.

Text categorization [84] , also known as text classification, is the process of assigning documents to two or more pre-defined categories. There are wide ranges of machine learning methods which can be applied in text categorization. The popular methods include neural network, decision tree, support vector machine (SVM) etc. In these methods, the SVM method is currently considered to be more effective. Also it is based on statistical learning theory and has a strong theoretical foundation.

Text clustering [40][22] is the process of partitioning the documents into subsets which are not pre-defined. The documents in each subset share some common attribute based on some similarity measurement. In other words, documents within a subset are

more similar to each other than those belonging to a different subset. Unlike text categorization, we don't need to provide the text clustering application with labelled documents for the purpose of training. The clustering method such as K-means can group the document collection into meaning subsets automatically.

Machine translation [49] is the process of translating text from one language into another through a computer system. The computer system aims to minimize human intervention and automatically translate languages. In order to achieve good performance, current machine translation systems tend to interpret and analyze all the features of the text such as the grammar, semantics and syntax. There are some popular approaches to solve technical problems of machine translation, which include linguistic rules, statistical methods, dictionary-based methods etc. Although machine translation systems still heavily suffer from lexical ambiguity, there is a wide range of machine translation systems available to the public; Google Translate and AltaVista's Babelfish which are easily accessible online translation applications.

Information extraction (IE) [49][31] is the process of identifying specific information from texts. This process can automatically generate a summary of an unstructured document collection by filling in text template. The text template is defined as a list of slots which correspond to certain information extracted from each document. The text template is structural and can store in traditional database. Namely, this process can allow texts to be accessed using traditional data management methods. IE approaches aim at understanding part of text data and finding the relevant information that the user wants.

Text search [58][7][2] is concerned with developing algorithms and models for finding information from textual repository. A search system can be briefly expressed as an abstract model which consists of document representations and query representations which are used as inputs to similarity matching unit. Document representations focus on collecting, organizing and indexing documents in the text repository. Further the system can effectively and efficiently access document representations which are the machine-understandable form of original documents. Query representations are used to form the appropriate query which the search service system can process in terms of information need. The similarity matching unit focuses on ranking, scoring and matching algorithms that output the search results through comparing document representations with query representations.

Those applications of text analysis are involved in many machine learning and natural language processing techniques. In this thesis, we focus on text search and its related techniques. In this chapter, we provide the background for text search and some related machine learning methods. The first half of this chapter introduces existing similarity matching approaches and discusses semantic document representations. Later we review some related techniques on query representation.

2.2 Overview of Search Service Models

Search service models contain similarity matching process as the important part of a search service system. The notable models that have been proposed rely on fundamental theories of sets, linear algebra and statistics. The following discussions introduce the most representative search service models in the current literature.

2.2.1 The Boolean Model

The Boolean retrieval model [80][79][96] is based on set theory and Boolean algebra. A document is represented as a binary sequence, which indicates the presence or absence of given terms. The queries are specified as Boolean expressions combined with AND, OR and NOT operators. Because the Boolean expressions have rigorous mathematical meanings, the search process can obtain a precise set of documents that exactly match the query. A document is relevant if it satisfies a Boolean expression of terms. Documents are retrieved in random order, that is to say, the Boolean model does not include the ranking mechanism. The Boolean model is easy to implement by using traditional SQL databases. There are three salient problems in the Boolean model. First of all, the users need some skills to formulate the proper Boolean expression for the query. This model is more suited to skilled users such as librarians or computer engineers etc. For some other users, formulating proper Boolean queries may become difficult. Secondly, the index terms in the Boolean model are equally significant. Thirdly, a document is only predicted to be either relevant or non-relevant in the Boolean model. The lack of the ranking mechanism may adversely affect search performance. Despite these shortcomings, the Boolean model is still a popular solution for digital library applications which require precise query results.

2.2.2 The Vector Space Model

In the vector space model [81][97], all documents and queries are viewed as vectors of index terms. The elements of the vectors indicate the presence or absence of a term like in the Boolean model. The binary representation does not take into consideration

the importance of the term in a document and in the whole collection. A preferable alternative consists of using term weights to differentiate the importance of the terms. There are two popular metrics to consider when utilizing term weights. Term frequency (TF) is defined as the number of occurrences of a given term in a document. The inverse document frequency (IDF) is used to measure the inter-documents importance. The TF-IDF weight is the representative weighting method used in the vector space model. The model can use a TF-IDF weight to represent the importance of a term in a document or query. If a term appears in many documents in the collection, the term is less capable of differentiating the documents. Term frequency measures how well a term contributes to the document content. A term with a higher term frequency is more important than a term with a lower frequency. If the term does not exist or appears in all documents, the weight computed for a term in a document vector is zero. The index of textual repository can consist of millions of terms and any document only contains a limited set of terms. Because most documents do not have most of the terms in the index, the majority of the values in the document vector are zero. The TF-IDF weight method is shown in the following Equation 2.1. The weight for a term will have a high value when its term frequency in the corresponding document is high, and its document frequency in the collection is low.

$$d_{ij} = tf_{ij} * idf_{\ j} \tag{2.1}$$

$$w_{qj} = tf_{qj} * idf_{\ j}$$

$$idf_{\ j} = \log_2(\frac{N}{n_i})$$

where w_{qj} is the weight of query term j in query q, and d_{ij} is the weight of document

term j in document i. N is the number of documents in the collection and n_i is the number of documents having the given term j.

There are diverse methods to measure the similarity between a document and a user query in the vector space. For instance, the Euclidean distance evaluates the distance between two vectors as shown in Equation 2.2. A small distance implies high degree of similarity.

$$sim(q, d_i) = \sqrt{\sum_{i=1}^{j}(w_{qj} - d_{ij})^2} \tag{2.2}$$

where $w_{q\ j}$ and $d_{i\ j}$ have been defined above.

Another important measure to evaluate similarity is the consine similarity function which measures the angle between the query and document vectors in multi-dimensional Euclidean space as shown in Equation 2.3.

$$sim(q, d_i) = \frac{\sum_{i=1}^{j}(w_{q\ j} * d_{i\ j})}{\sqrt{\sum_{i=1}^{j} w_{q\ j}^2 * \sum_{i=1}^{j} d_{i\ j}^2}} \tag{2.3}$$

where $w_{q\ j}$ and $d_{i\ j}$ have been defined above.

The vector space model can sort the documents in the order of similarity, which can tailor the size of the ranking documents for the user's request. Note that the terms that are not in both the query and the document do not contribute to the result of consine similarity function. If the query is short, its vector would be extremely sparse, so this model is easy to implement and the cost of computation is low. But the downside of the TF-IDF weighting method is that it cannot solve synonymy and polysemy problems.

2.2.3 Probabilistic Models

Probabilistic Model [76][47] defines the degree of relevance of a document to a query in terms of the probability that the document is relevant to the query. The fundamental assumption of the probabilistic model is that it is feasible to estimate the probability of the relevance of a document to a given user's query q. Equation 2.4 defines the similarity measure of a document d to a query q as the ratio.

$$sim(d\ ,\ q) = \frac{P(R \mid d)}{P(\widetilde{R} \mid d)} \tag{2.4}$$

Where R is the set of documents known to be relevant, $\widetilde{R}$ is the set of non-relevant documents, $P(R \mid d)$ is the probability that document d is relevant to the query q, and $P(\widetilde{R} \mid d)$ is the probability that d is non-relevant to the query q.

2.2.4 The Link Analysis Models

The link analysis models include effective ranking algorithms which are used to evaluate the quality of web pages on the Internet by means of analyzing the hyperlinks of the web pages.

As shown in Figure 2.1, the World Wide Web is organized through hyperlinks of web pages. There are two representative link analysis models which are widely used in search engines such as Google, Yahoo!, MSN search etc. The most successful algorithm [10][11] is the PageRank algorithm which treats the web as a huge graph in which each web page is connected with its related web pages by hyperlinks. Each web page is assigned a score which depends on a probability distribution over all web pages among the hyperlink structure. When a user submits a query, the web search engine returns a ranked list

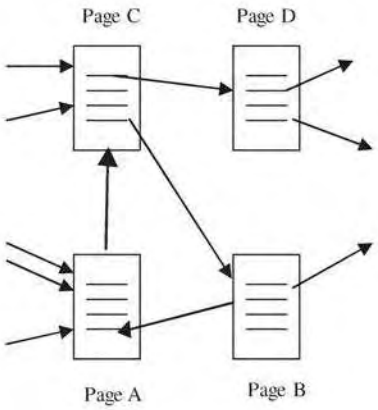


Figure 2.1: The Link Structure of World Wide Web

of web pages in terms of the keywords which are computed by the similarity measure based on the predefined web page's score. Another important ranking algorithm is the hypertext induced topic selection (HITS) algorithm [51][13]. Unlike PageRank algorithm, there are two scores assigned to each page, namely, authority score and a hub score. The essence of this algorithm is its recursive assumption: a good authority is a page that is directed by good hubs, and a good hub is a page that directs to good authorities. In other words, the hub score is determined by the authority scores related to the web page; the authority score is determined by the hub scores related to the web page. When a user submits the keywords, the search services create a subset of web pages related to the keywords; then it returns a ranked list of these web pages after computing their hub and authority scores.

2.3 Semantic Analysis Techniques

Conventional search service systems assume that the association of query and document can be expressed through a set of index keywords. Obviously this simplified

assumption may sacrifice semantic values in the document collection which contain rich information. In this section, we investigate the semantic analysis techniques which can provide semantic representations of the document collection. The explicit semantic analysis attempts to build semantic document representations based on natural language processing; the latent semantic analysis focus on constructing semantic document representations based on machine learning.

2.3.1 Explicit Semantic Analysis

Lexical chain analysis [50][66][86] can create semantic document representations by discerning the explicit semantic connections among words in the documents. A lexical dictionary such as WordNet is an essential part of this technique. The lexical chains represent clusters of related words within one document. It can also be used for disambiguating word senses and emphasize the content of documents.

The lexical chain analysis technique firstly determines the major types of relations between words. These relations include identity, synonym, hypernym, meronymy etc. Currently these relations are only applied to nouns. The representative terms in documents can be recognized through the use of part-of-speech tagger. The nouns are extracted from the documents and a set of candidate terms is created. For each candidate term, we look for the semantic relation with the existing lexical chains in WordNet. If the relation can be found, the term is inserted in the corresponding lexical chain accordingly. According to this criterion, the appropriate lexical chain clusters are generated. After constructing the lexical chain clusters, each cluster represents one concept. There are three important steps to build semantic document representations.

The initial weight estimation of the concepts is based on term relations, as shown in Equation 2.5 .

$$S_{concept}(C_x) = \sum_i S_{noun}(T_i) \tag{2.5}$$

where $S_{noun}(T_i)$ is the score of term T_i, $S_{concept}(C_x)$ is the score of concept C_x which is expressed as a lexical chain cluster. Equation 2.6 shows the weight re-estimation in the concept vector space.

$$\Psi_{T_j\ |D} = \Psi_{T_j\ |C_i} * \Psi_{C_i\ |D} \tag{2.6}$$

where $\Psi_{T_j\ |D}$ is the weight ratio T_j in the document D, $\Psi_{T_j\ |C_i}$ is the weight ratio T_j in the concept C_i, $\Psi_{C_i\ |D}$ is the weight ratio C_i in the document D. Equation 2.7 is used to build the semantic index.

$$\Omega_{W_j} \geq \theta * \frac{1}{m} \sum_i (\Omega_{W_i}) \tag{2.7}$$

where Ω_{W_j} is the term magnitude that is determined by concept C_j. The parameter m is the number of concepts. The parameter θ is used to adjust the number of index terms. Note that as the value of the parameter θ increases, a higher compression ratio of the index can be obtained.

The major downside of this technique is that a lot of valuable information in the documents is lost in that only the nouns in a document are considered. It heavily relies on the lexical dictionary which is mainly used for the ontology represented by nouns. The verbs, adjectives and adverbs have weak semantic connections in the lexical

dictionary. Namely, it is difficult to construct semantic clusters for different parts of speech.

2.3.2 Latent Semantic Analysis

The latent semantic analysis (LSA) technique [6][20][26][27][28] [4] attempts to solve the synonym and polysemy problems. The term occurrence in a document may not accurately reflect the semantic meaning of the document. The LSA technique can uncover the hidden semantic relations from the documents rather than just consider the term occurrences. Also this technique provides a good measure of semantic similarity among documents represented by terms. The main idea in this technique is to transform the documents vectors into a low dimensional space which can generate clusters called concepts. This approach can strengthen terms association which can be used to distinguish semantic clusters in documents.

The latent semantic analysis was introduced in 1990 by Deerwester, et al [20], and has since become a powerful tool used to analyze documents at semantic level. Its mechanism relies on singular value decomposition (SVD) theory which can convert term-by-document matrix into a low rank approximation representation [54]. During the LSA process, a conceptual vector space is created. The vector space reveals the latent semantic structure of association among terms and documents in the text collection. SVD approximates term-document associations by using only the k largest singular values and corresponding singular vectors. In the conceptual vector space, each point represents one word or one document. A neighborhood of points represents one concept. Landauer et al. [54] provide this description of the semantic relationship

generated by the SVD. A document can belong to several related concepts; also one term can belong to several concepts. The related terms and documents can become closer in the conceptual vector space. In other words, LSA can reveal semantic relationships among terms and documents in the conceptual vector space through singular value decomposition.

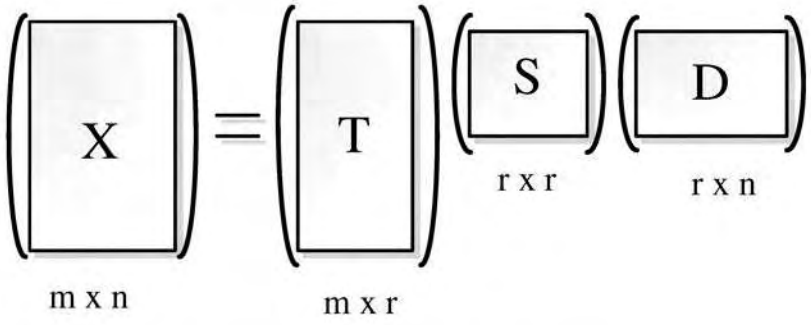
In LSA, the document collection is treated as a bag of words, without consideration for word order, sentences' logic, or morphology. Many text analysis studies perform a degree of morphological transformation on individual terms so that plural terms or verb forms are not treated as separate terms [98]. LSA ignores the impact of how words produce the meaning of a sentence to infer only how the variations in word choice and differences in the meaning of specific documents are related. Latent semantic analysis represents the meaning of a word as a combination of the meaning of the various documents that utilize that word, while the documents themselves are represented as a grouping of the words they contain. One of advantages of LSA lies in its ability to build the association of terms not inherently obvious in the documents and produce good performance results. Another advantage of LSA is its efficiency in clustering terms. To some extent, LSA satisfies the need for term clustering because it can process term-to-term comparisons and find combinations of similar words [57][85].

Some other studies relevant to this approach have been made in recent years. Syu et al. [90] reported that their neural network incorporating LSA technique outperform their previous thesaurus-based model. Latent Semantic Kernels [17] is another new approach that combine the LSA with Support Vector Machines (SVM). It applies the Gram-Schmidt orthogonalization instead of the eigenvalue decomposition. This avoids

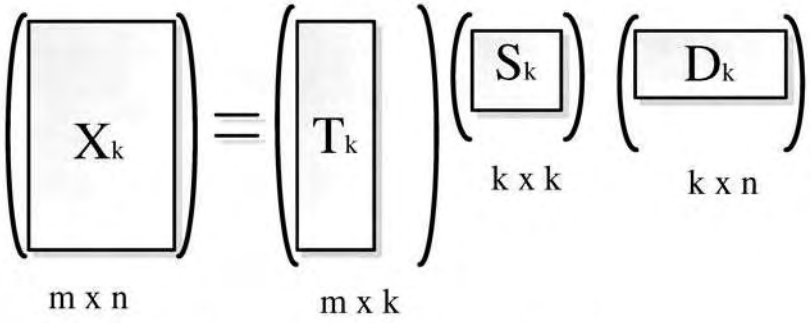
the computational expense of eigenvalue decomposition and is able to obtain good performance. Yehuda et al. [99] discuss the LSA's relationship to statistical regression and Bayesian methods. Ding et al. [23] construct a statistical model for the LSA using the cosine similarity measure. It shows that the term similarity and document similarity measures are formed during the maximum likelihood estimation and demonstrates that the LSA is the optimal solution to this model. The probabilistic latent semantic analysis (PLSA) [39] is another model based on a statistical foundation inspired by the LSA. It employed the method of maximum likelihood to reduce dimension for retrieval and has shown a promising performance in the text analysis area. The method predicts the influence of k approximation over the distribution of terms and documents in the conceptual space. The model is based on the Kullback-Leibler projection instead of the orthogonal projection. Compared with the LSA which is a kind of lossy approximation, Kai et al. [100] improve the LSA model through mapping the text features into a new feature space that keep the original text features and reveal the dependency of output dimensions.

The LSA is based on the theory of singular value decomposition [88]. In the first step, documents are converted into a term-document frequency matrix. Then, a local and global weighting scheme is used to normalize the element values in the matrix. Further, singular value decomposition (SVD) is performed to reduce the dimensions and build the semantic relationships. Finally, term–to-term comparison, document–to-document comparison and term-to-document comparisons can be performed to discern latent semantic information.

The LSA makes use of the singular value decomposition to construct the conceptual



(a) Singular Value Decomposition



(b) Rank k Approximation by Singular Value Decomposition

Figure 2.2: The Process of Latent Semantic Analysis

vector space. As can been seen in Figure 2.2, a given m by n matrix X can be decomposed into

$$X = TSD^T \tag{2.8}$$

where X is m-by-n matrix with singular values $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r \geq 0$, T and D are orthogonal matrices and $S = diag(\sigma_1, \sigma_2, \ldots, \sigma_r)$. Note that $TT^T = T^TT = I$ and $DD^T = D^TD = I$. Matrix T are the eigenvectors of XX^T. Matrix D are the eigenvectors of X^TX.

Equation 2.9 shows the rank k approximation of matrix X:

$$X_k = T_k S_k D_k^T \tag{2.9}$$

where X_k is a m-by-n matrix of rank k ($k < r$) with singular values $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_k \geq 0$, T_k and D_k are the first k columns of T and D, respectively, from Equation 2.9, and $S_k = diag(\sigma_1, \sigma_2, \ldots, \sigma_k)$.

After the rank k approximation of X is calculated, the LSA can proceed with further processing such as term–to-term comparison or document-to-document comparison.

In order to compare terms correlation, Equation 2.10 can measure terms similarity across documents in the collection. The original matrix calculation proceeds as follows:

$$\begin{aligned} XX^T &= (TSD^T)(TSD^T)^T \\ &= TSD^T DST^T \\ &= TSST^T \\ &= (TS)(TS)^T \end{aligned} \tag{2.10}$$

After approximation of matrix X, similarly the rank k matrix calculation can proceed as follows:

$$X_k X_k^T = (T_k S_k)(T_k S_k)^T \tag{2.11}$$

In order to compare documents correlation, the following formula can measure the similarity among documents. The original matrix calculations proceed as follows:

$$X^T X = (TSD^T)^T (TSD^T)$$

$$= DST^T TSD^T$$

$$= DSSD^T$$

$$= (DS)(DS)^T$$

After approximation of matrix X, similarly the rank k matrix calculation can proceed as follows:

$$X_k^T X_k = (D_k S_k)(D_k S_k)^T \tag{2.12}$$

2.4 Some Related Techniques In Search Services

2.4.1 Relevance Feedback

The relevance feedback technique is an effective approach to enhance the query representation through user interaction. When performing the search process, the user can mark the related documents in the result list as relevant to the user request. The system can refine the results by virtue of user feedback. According to user feedback, the Rocchio algorithm [93] provides the mechanism for reweighing the original query.

$$q_r = \alpha * q + \frac{\beta}{\mid R \mid} \sum_{d_j \in R} d_j - \frac{\gamma}{\mid \widetilde{R} \mid} \sum_{d_j \in \widetilde{R}} d_j$$

where q_r is the new query and q is the original query, α is the coefficient for the original query, β is the coefficient for relevant documents set R, γ is the coefficient for

irrelevant documents set $\widetilde{R}$.

The relevance feedback can produce good results in practice, but the users need to have the patience to give their feedback to the system. In addition, the values of the coefficient α, β and γ need to be assigned empirically.

2.4.2 Filtering Technique

Filtering systems attempt to create the profiles of users that capture their interests. The profiles can be treated as query representation. When the user visits the web site powered by a filtering system, the profile of the user is monitored. The filtering system can recommend documents to the user by virtue of the similarity of content of the documents and the user profile. The user is expected to register in the filtering system. Especially the user needs to provide the filtering system with some personal information that has personal privacy implications.

2.4.3 Question-Answering System

The Question-Answering system is a typical system of natural language processing in the search service system. The system aims to help users seeking answers in response to query representation expressed as a formal question in natural language like "who", "what", "when" and "how" question sentences. The strategy of the system is to employ natural language processing to analyze the question and retrieve documents relevant to the question, then extract the answer from those documents. One representative Question-Answering system is proposed by Dumais et al. [25]. In the first step, the system formulates queries whereby the syntactical structure of question sentence is

similar to that of answers. Secondly, rewritten queries are sent to search engine for getting the list of results. Then a result analysis of N-grams is performed and every N-gram token is assigned a weight for reliability. Regular expressions are used to find best-match N-grams. Finally, top ranking N-grams are merged to generate the answers. The Question-Answering system requires users to present the formal question as the query instead of keywords. Most users still prefer to post keywords as the query.

2.4.4 Text Classification

In this section, we introduce the text classification techniques that are widely used in the area of text analysis. Given a predefined set of category $C = \{c_1, c_2, ..., c_n\}$ and a set of document $D = \{d_1, d_2, ..., d_m\}$, a training sample is a pair (d_i, c_j), where d_i is a document represented as a vector of k terms $d_i =< t_1, t_2, ..., t_k >$; c_i is a label associated with d_i for the corresponding training sample. A training set S is a set of pairs of labelled training samples $S = \{(d_i, c_j) \mid d_i \in D \ , \ c_j \in C\}$. The goal of text classification is to learn the patterns from the training set S so that it can effectively classify an unknown test example d to its corresponding category with high accuracy [64].

Text classification is a broad research area in the field of machine learning. In this section, we discuss the naïve Bayes classification that is employed in our search service system. The naïve Bayes classification [64][24][77][36][12] is a powerful probabilistic classifier which is based on the Bayes's theorem and the naïve Bayes independence

assumption. The Bayes's theorem can be represented as:

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)} \tag{2.13}$$

where $P(A)$ is the prior probability of A, and $P(B)$ is the prior probability which can acts as a normalizing constant. $P(A \mid B)$ is the posterior probability depends on the specified value of B. $P(B \mid A)$ is the conditional probability of B given A.

Given a unknown test example $d =< t_1, t_2, ..., t_l >$, the classification function $f(d)$ is:

$$f(d) = \arg\max_{c_j \in C} P(c_j \mid d) = \arg\max_{c_j \in C} \frac{P(d|c_j)P(c_j)}{P(d)} \tag{2.14}$$

Bayes' theorem states that a document d should be assigned to the category c_i for which $P(c_i|d)$ is highest in order to achieve minimum classification error. The naïve Bayes independence assumption states that the probability of term t_i is independent from any other term t_j given that the class is known. This assumption simplifies the complexity and makes it easier to implement the naïve Bayes classifier in practice.

Because $P(d)$ cannot affect the result of $f(d)$, it can be treated as a constant and removed from $f(d)$. Also $P(d|c_j)$ means that $P(t_1, t_2, ..., t_l|c_j)$, and Equation 2.14 becomes:

$$f(d) = \arg\max_{c_j \in C} P(t_1, t_2, ..., t_l|c_j)P(c_j) \tag{2.15}$$

Because of the naïve Bayes independence assumption, we assume that every t_i is independent. The joint probability $P(t_1, t_2, ..., t_l|c_j)$ becomes:

$$P(t_1, t_2, ..., t_l|c_j) = \prod_{1 \leq t_i \leq l} P(t_i|c_j)$$

So, $f(d)$ becomes:

$$f(d) = \arg\max_{c_j \in C} (\prod_{1 \leq t_i \leq l} P(t_i|c_j) * P(c_j)) \tag{2.16}$$

At the training phase, the classifier estimates the category prior probabilities $P(c_j)$ and the conditional probability $P(t_i|c_j)$ of each term t_i given the category c_j. The estimated $P(c_j)$ is:

$$P(c_j) = \frac{|c_j|}{|D|} \tag{2.17}$$

where $|c_j|$ denotes the number of training documents in category c_j and $|D|$ is the total number of training documents.

The estimated $P(t_i|c_j)$ is:

$$P(t_i|c_j) = \frac{a_{i\ ,\ j}}{\sum_{a_{k\ ,\ j} \in B_j} a_{k\ ,\ j}} \tag{2.18}$$

where $a_{i\ ,\ j}$ denote the frequency of term t_i in documents of category c_j. Also $B_j = \{a_{i\ ,\ j}\}$ denote the set of frequency of terms in documents of category c_j.

If $P(t_i|c_j)$ is zero, $f(d)$ become zero and meaningless. In order to avoid zero value of $P(t_i|c_j)$, the revised $P(t_i|c_j)$ become:

$$P(t_i|c_j) = \frac{a_{i\ ,\ j} + 1}{\sum_{a_{k\ ,\ j} \in B_j} (\ a_{k\ ,\ j}\ +\ 1)} \tag{2.19}$$

Due to its simplicity and effectiveness, the naïve Bayes classifier is acceptable for heavy computational tasks. It is well suited for many applications.

2.5 Chapter Summary

In this chapter, we provided background and discussed related work in the development of search service systems. We introduced some popular algorithms for similarity matching in search service systems. Some techniques for document representations and query representation were also discussed. In the next chapter, we propose an intention detection model which can detect the user intention to some degree by virtue of the user's queries in a specific domain. From our point of view, if the user's intention can be inferred by the search service system, an informative query representation can reasonably be constructed to produce better search results.

Chapter 3

Intention Detector Construction

3.1 Introduction

In this chapter, we introduce a generalized intention detector framework based on a directed graph model [55][42][67][48][62]. It derives from the combination of graph theory and probability theory. From the perspective of machine learning, directed graph models can perform complex probabilistic inference and learning tasks. It has played a significant role in machine learning in recent years [38]. A directed graph model is made up of nodes and edges with each node denoting a variable that can be hidden or observable, and the edges representing the relationships between these variable. In particular, the hidden Markov chain [8] is one of the classic directed graph models that has been widely used in many applications such as speech recognition, robot motion planning and biomedical systems. There exist some well-verified powerful algorithms that can be used for the specific applications directly. We propose a new application of directed graph models within the context of the text search problem.

The hidden Markov chain is characterized by hidden variables and observable variables. Given a finite sequence of observations o(1), o(2),...,o(n) where each observation

is a discrete value drawn from a finite set O. Then the sequence of observations can be modelled as a probabilistic output of an underlying order of one Markov chain h(1),h(2),...,h(n). Their state transitions are not directly observable, therefore the Markov chain is represented by hidden variables.

We attempt to construct an intention detector that can assist users in finding information. It is difficult to define a user's intention within a user's query especially in a web search. Different people have different ideas, and a given query could have diverse intentions. In practice, we can narrow down a user's intentions in specific domains like job search, movie search, dating match etc. When a user visits a specific web site, he/she has some vague intentions that the intelligent query system have already detected in context. For example, when one user accesses the movie search web site, it implies that he/she wants to find some information related to movie. The query system can confine the user's intention to certain scope and predict possible user's intention. In other words, user intentions should be limited. Usually a user needs to issue queries several times to gain satisfactory results for one vague information need. This behavior provides opportunities for the search service system to predict user intention. Our focused problem is how to determine the specific type of user intention and detect the underlying intentional states. In this way, user intention detection is of great importance in developing a search service system with inference capabilities that will facilitate an advanced online web service. As can be seen in Figure 3.1, the process of user intention detection can be divided into two stages: classification of the type of user intentions and detection of the state of the specific user intention. In the scenario of job search, a job seeker has a job request and tries to submit the queries

for a good job fit. The type of user intention can be industry type such as information technology, engineering, administration etc. The state of specific user intention can be job type in one specific industry type such as programmer, network engineer, helpdesk etc. in the type of information technology. The physical meaning of type and state of user intention can be defined by domain experts.

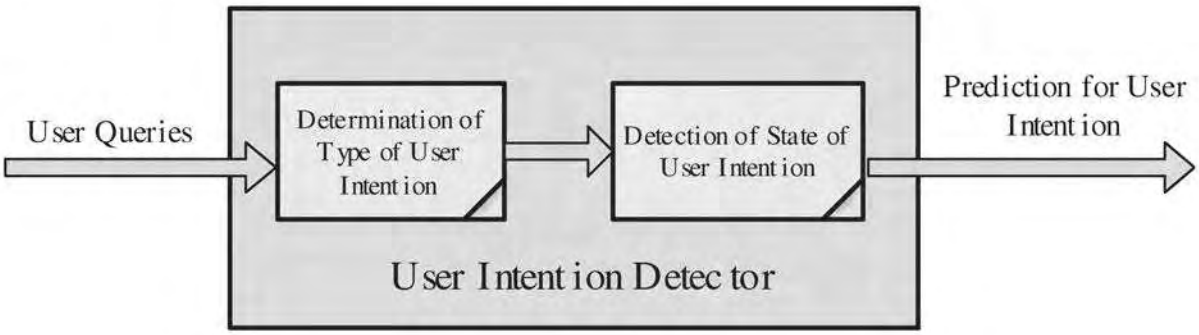


Figure 3.1: User Intention Detection Procedure

The user intention is modelled as a hidden Markov chain generated by two related mechanisms, an underlying Markov chain having a finite number of states and each state of the chain associated with the corresponding observable query. The values from the function of the queries are observation likelihoods. The model is characterized by parameters which consist of the state transition probabilities, the observation likelihoods and the initial state distribution. Those parameters can be estimated by a variation of the Expectation-Maximization (EM) algorithm, namely the Baum-Welch algorithm [74] for the hidden Markov chain. In the inference stage, the state of the user intention can be identified with the highest probability.

3.2 Determination of Type of User Intentions

Suppose that there are m types of user intentions. We take the observed variable to be the query q, and the hidden variable to be a desired user intention. Let $q_{1:\ n+1} = \{q_{\ 1}, ..., q_{\ n+1}\}$ be a sequence of observed queries. Given a sequence of queries, the model focuses on classifying the user intention in context. It provides basis for predicting the state of the specific user intention in next stage. We assume that current user intention is only dependent of previous user intention. The model is shown in Figure 3.2

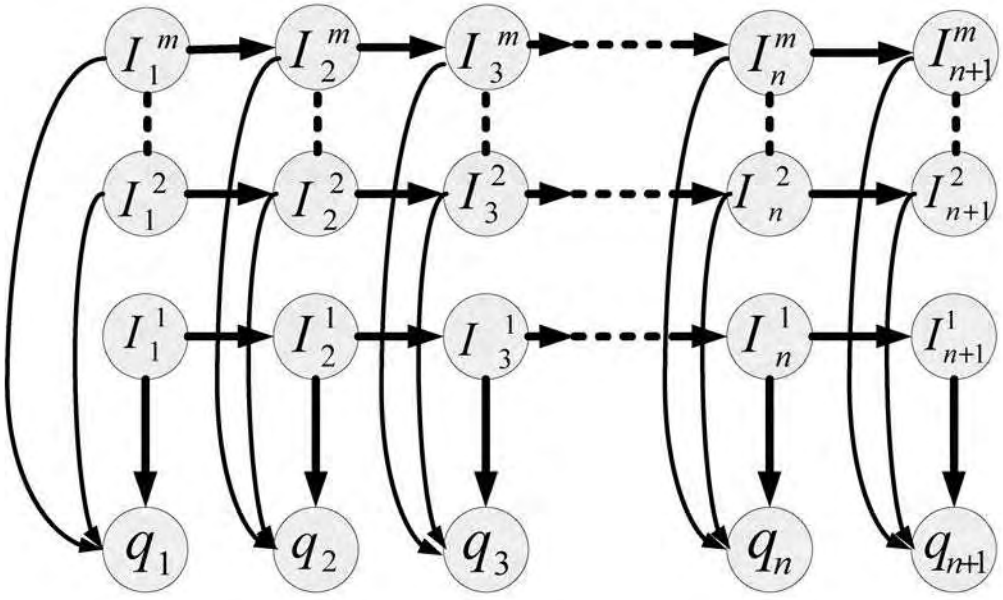


Figure 3.2: Illustration of Determination of Type of User Intentions

In Figure 3.2, an individual user issues a sequence of queries for a certain period. The model consists of m different first order hidden Markov chains which depict m types of user intentions. In the case of movie search, user intentions mean that the user intend to search for the movie topic, user rating, user comment etc. The intention node $I_{\ k}^{\ i}$ denotes the kth node for the i type intention. The intention node $I_{\ k}^{\ i}$ is a hidden variable and only dependent on the intention node $I_{\ k-1}^{\ i}$. Also the query node

q_k is an observable variable and dependent on the intention node series $I\ ^1_k, I\ ^2_k, ..., I\ ^m_k$.

Firstly, we compute the best intention sequence $I\ ^*_i$ in each type intention $I\ ^i_1, I\ ^i_2, ..., I\ ^i_{n+1}$ given the query sequence $q_{1:\ n+1}$ by using Viterbi's algorithm [74].

$$I\ ^*_i = \arg \max_{I\ ^i_1, I\ ^i_2, ..., I\ ^i_{n+1}} P(I\ ^i_1, I\ ^i_2, ..., I\ ^i_{n+1} |\ q_{1:\ n+1}) \tag{3.1}$$

$$= \alpha P(q_{n+1} |\ I\ ^i_{n+1}) * \arg \max_{I\ ^i_n} (P(I\ ^i_{n+1} \mid I\ ^i_n) * \arg \max_{I\ ^i_1, I\ ^i_2, ..., I\ ^i_n} P(I\ ^i_1, I\ ^i_2, ..., I\ ^i_n |\ q_{1:\ n}))$$

where α stands for the normalizing constant, $P(q_{n+1} |\ I\ ^i_{n+1})$ is the observation likelihood for query q_{n+1}, and $P(I\ ^i_{n+1} \mid I\ ^i_n)$ is the transition probability for the states of user's intention,

In particular, $\arg \max_{I\ ^i_1, I\ ^i_2, ..., I\ ^i_n} P(I\ ^i_1, I\ ^i_2, ..., I\ ^i_n |\ q_{1:\ n})$ is the previous sequence probability.

Then we find the best type of user intention by computing the largest probability of query sequence $q_{1:\ n+1}$ given every best intention sequence $I\ ^*_i$ for each type of intention.

$$P\ ^*_i = \max_i\ P(q_{1:\ n+1} |\ I\ ^*_i) \tag{3.2}$$

$$= \max_i \prod_{\substack{1 \le k \le n+1 \\ state_k \in I\ ^*_i}} P(q_k | state_k)$$

,AAr1c5

3.3 Detection of the State of the Specific User Intention

In the second stage, suppose that the intention detector has found the best type i of user intention in the first stage. For an individual user, intention can be described

as the stochastic process which consists of a set of states of intention. The process is determined by the transition probabilities whereby it may transit from one state to another or remain at the same state. The intention detector needs to predict the state of intention I_{n+1} in type i when the user submits the latest query q_{n+1}. The model is shown in Figure 3.3. For this purpose, we can use the forward algorithm [74] to predict the intention I_{n+1}.

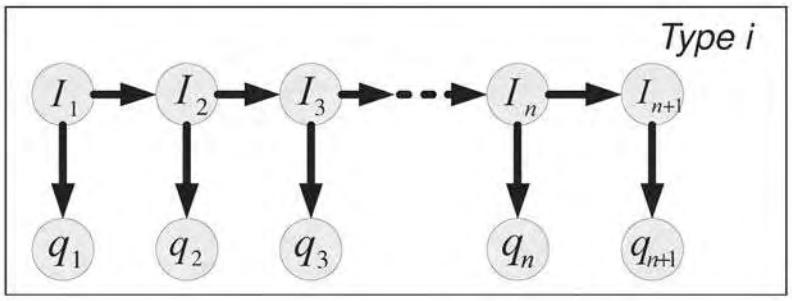


Figure 3.3: Illustration of First Order Hidden Markov Chain for Prediction

$$
\begin{aligned}
P(I_{n+1}|\, q_{1:\,n+1}) &= P(\, I_{n+1}|\, q_{1:\,n}, q_{n+1}) \\
&= \alpha\; P(q_{n+1}|I_{n+1}, q_{1:\,n}) * P(I_{n+1}|\, q_{1:\,n}) \\
&= \alpha\; P(q_{n+1}|I_{n+1}) * P(I_{n+1}|\, q_{1:\,n}) \\
&= \alpha\; P(q_{n+1}|I_{n+1}) \sum_{I_n} \{\, P(I_{n+1}|\, I_n\, , q_{1:\,n}) * P(\, I_n \mid q_{1:\,n})\} \\
&= \alpha\; P(q_{n+1}|I_{n+1}) \sum_{I_n} \{\, P(I_{n+1}|\, I_n\,) * P(\, I_n \mid q_{1:\,n})\}
\end{aligned}
\tag{3.3}
$$

where α denote the normalizing constant, $P(q_{n+1}|I_{n+1})$ is the observation likelihood for query q_{n+1}, and $P(I_{n+1}|\, I_n\,)$ is the transition probability for the states of user's

intention, $P(\ I_n \mid q_{1:\ n})$ is the previous sequence probability. $q_{1:\ n}$ denote the query vector from query q_1 to query $q_{\ n}$

Even though the first order hidden Markov chain model can predict the state of intention I_{n+1}, it does not consider the uncertainty of user satisfaction when doing the query.

The user could be satisfied by the result after submitting the query once. In Figure 3.4 ,the independent node model can more effectively reflect this scenario. From Figure 3.4, we can write:

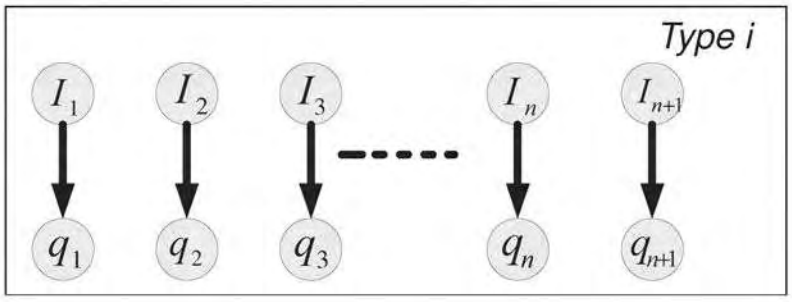


Figure 3.4: Illustration of Independent Node Model for Prediction

$$P_0(I_{n+1} \mid q_{n+1}) = \alpha P(q_{n+1} \mid I_{n+1}) P(I_{n+1}) \tag{3.4}$$

where α stands for the normalizing constant, $P(q_{n+1}|I_{n+1})$ is the observation likelihood for query q_{n+1}, and $P(I_{n+1})$ is the prior probability.

Let us assume that the user could only obtain a satisfactory result after submitting the query thrice. The model shown in Figure 3.5 effectively reflect this scenario. Note that the additional edge from I_{n-1} to I_{n+1} comes from the definition of second-order

Markov chain. It means that a particular hidden variable I_{n+1} depends on the values of the two previous hidden variable I_n and I_{n-1}. In our setting, it means that the previous successive user intentions would provide important associated information in predicting the current user intention.

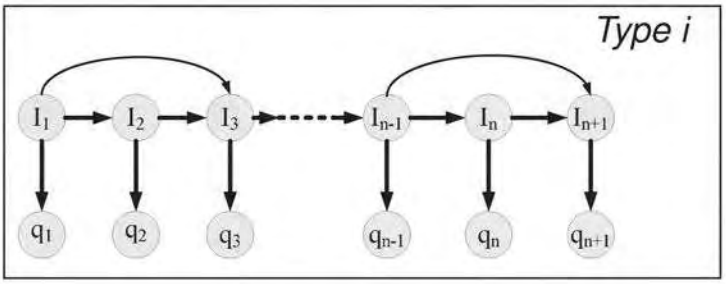


Figure 3.5: Illustration of Second Order Hidden Markov Chain for Prediction

$$
\begin{aligned}
P_2(I_{n+1}|\, q_{1:\,n+1}) &= P(\, I_{n+1}|\, q_{1:\,n}, q_{n+1}) \qquad (3.5)\\
&= \alpha\; P(q_{n+1}|I_{n+1}, q_{1:\,n}) * P(I_{n+1}|\, q_{1:\,n})\\
&= \alpha\; P(q_{n+1}|I_{n+1}) * P(I_{n+1}|\, q_{1:\,n})\\
&= \alpha\; P(q_{n+1}|I_{n+1}) \sum_{I_n}\{\, P(I_{n+1}|\, I_n\,, q_{1:\,n}) * P(\, I_n \mid q_{1:\,n})\}\\
&= \alpha\; P(q_{n+1}|I_{n+1}) \sum_{I_n}\{\, P(I_{n+1}|\, I_n\,) * \sum_{I_{n-1}}\{\, P(I_n|\, I_{n-1}) * P(\, I_{n-1} \mid q_{1:\,n-1})\}\}
\end{aligned}
$$

where α denote the normalizing constant, $P(q_{n+1}|I_{n+1})$ is the observation likelihood for query q_{n+1}, $P(I_n|\, I_{n-1})$ and $P(I_{n+1}|\, I_n\,)$ is the transition probability for the states of user's intention, and $P(\, I_{n-1} \mid q_{1:\,n-1})$ is the previous two sequence probability. $q_{1:\,n}$ denote the query vector from query q_1 to query $q_{\,n}$.

In order to make better predictions, let us introduce a more general model to

combine the three models discussed above.

$$P_{mix}(I_{n+1} \mid q_{n+1}) = \theta_0 * P_0(I_{n+1} \mid q_{n+1}) + \theta_1 * P_1(I_{n+1} \mid q_{1:\, n+1}) + \theta_2 * P_2(I_{n+1} \mid q_{1:\, n+1}) \tag{3.6}$$

subject to $0 \leq \theta_i \leq 1$ and $\sum_i \theta_i = 1$.

$P_0(I_{n+1} \mid q_{n+1})$ is the independent node model given a query at time n+1. $P_1(I_{n+1} \mid q_{1:\, n+1})$ is the first order Markov chain model given a query at time n+1. $P_2(I_{n+1} \mid q_{1:\, n+1})$ is the second order Markov chain model given a query at time n+1. The parameter θ_i is the ratio for the user's search behavior. The value of the parameter θ_i represents the percentage of query sequences which correspond to the respective model, hence this value can be assigned according to statistical analysis of professional web sites' records.

3.4 The Simplified Intention Detector

The real raw data of sequence queries are a valuable asset for professional web sites. It is difficult to obtain this kind of raw data for doing experiments in a small laboratory setting. In this section, we simplify our framework and introduce one practical model. In the next chapter, we design a system to demonstrate how to incorporate this simplified intention detector into the semantic analyzer.

In the generalized framework, Figure 3.2 shows that the parameter m is used to determine the number of layers of the hidden Markov chain. When m is equal to 1, it means that the intention detector is only able to detect one major type of user intention. In the case of movie search, the detector only can predict the topic related

to the movies. The parameter θ_i can be assigned to a special case. When θ_0 amounts to 1, θ_1 and θ_2 become zero. It means that the user intentions are independent of each other. In other words, the user is satisfied or rejects the result when submitting the query once. As shown in Figure 3.6, the simplified model is the basis of other complex variants. We describe this model in more details in the next chapter. The node I denotes the user's intention, and the node q denotes the query which the query system can detect. The detector only infers the user's intention in terms of the query.

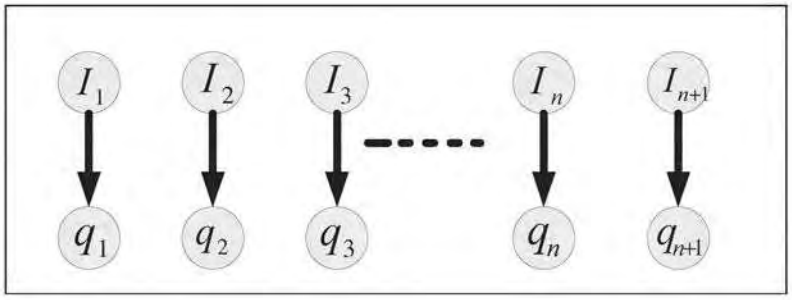


Figure 3.6: Illustration of the Simplified Intention Detector

In the independent node model, all the user queries' intentions are independent, thus each query can be processed separately. Because the query q is made up of keywords that the user type in, for each independent node, the simplified detector can be expressed as shown in Figure 3.7. Notice that the t_i stands for one of keywords that the user supplies.

The model can be interpreted as naïve Bayes for implementation in the next chapter. The $state_j$ denotes the state of the user's intention. For example, the $state_j$ means the user wants to find the movies related to a specific topic. We will discuss the

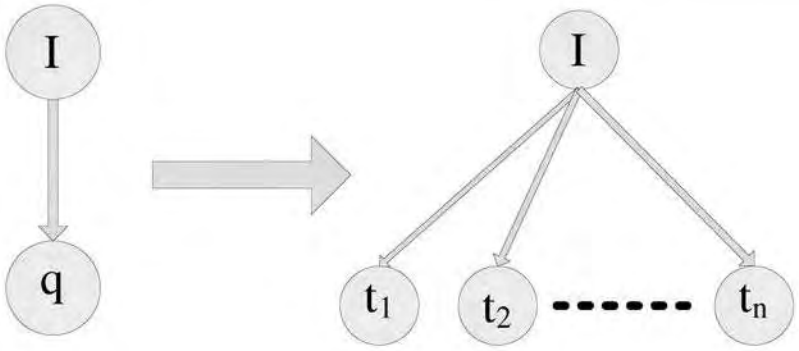


Figure 3.7: Illustration of Keywords Representation of the Simplified Intention Detector

implementation in detail in the next chapter. The classifier function $f(q)$ is as follows:

$$
\begin{aligned}
f(q) &= \arg \max_{state_j \in I} P(state_j|q) \qquad (3.7)\\
&= \arg \max_{state_j \in I} \frac{P(q|state_j)P(state_j)}{P(q)}\\
&= \arg \max_{state_j \in I} P(t_1, t_2, ..., t_n|state_j)P(state_j)\\
&= \arg \max_{state_j \in I} (\prod_{1 \leq t_i \leq n} P(t_i|state_j) * P(state_j))
\end{aligned}
$$

3.5 Selection Of Feature Words

Each user intention state can be described by some representative feature words. After the intention detection, the selection of feature terms can be performed to formulate new queries. In this section, we introduce two approaches for selecting feature words based on the semantic similarity which is based on WordNet. WordNet [63] is an electronic dictionary which can represent the relations among English words. WordNet has been widely used in the field of natural language processing, artificial intelligence, language generation and machine translation. The basic unit in WordNet is the synset

which is a set of words that share the same sense (synonyms). The synsets are related to each other by different types of relationships. Synsets contain either nouns, or verbs, or adjectives, or adverbs. Each synset consists of synonym words and pointers to the hypernymy, hyponymy, antonymy,entailment, and meronymy. Synsets can contain a certain amount of information and the pointers represent the association between a word in one synset and other synsets. We can utilize these relations to measure semantic similarity.

For semantic similarity based on WordNet, Resnik [75] stated that nodes near the leaves of the taxonomy are more informative than the nodes near the root of the taxonomy. The nodes near the leaves of the taxonomy can be assigned to high values. For instance, the term "eagle" is more informative than the term "bird". The degree of the information content can be evaluated through the frequency counts. The information content is defined as:

$$IC(s) = -\log(P(c)) \tag{3.8}$$

where $P(c)$ is the probability of synset of word c occurring in the WordNet.

Later on, Jiang et al. [44] proposes the semantic similarity between a pair of synsets as:

$$sim_J = \frac{1}{IC(c1) + IC(c2) - 2 * IC(sp(c1, c2))} \tag{3.9}$$

where $IC(c1)$ and $IC(c2)$ is nodes' information content, The $sp(c1, c2)$ is the node which represent the shared parents of both nodes.

Lin [60] presents another different method to measure the semantic similarity between a pair of synsets as:

$$sim_L = \frac{2 * IC(sp(c1, c2))}{IC(c1) + IC(c2)} \tag{3.10}$$

where $IC(c1)$, $IC(c2)$ and $sp(c1, c2)$ have defined above. These two similarity metrics can be utilized to measure the similarity to the words between feature set and user query at the semantic level.

3.6 Chapter Summary

In this chapter, we assume that users often issue several queries to search the ideal result for a specific purpose. This search behavior can be characterized as sequence of queries each associated with one corresponding user intention. Then we propose a generalized framework of user intention detector. Further, we introduce a simplified intention detector which is discussed in details in next chapter. It should be noted that the inference capability of the intention detector is currently limited to one domain because people have diverse intentions cross different domains which the intention detector cannot capture. In the next chapter, we design a search service system with inference capabilities in a concrete scenario. The intention detector can work as a component in the system.

Chapter 4

A Case Study Of Movie Plot Search

4.1 Introduction

In this chapter, we introduce a search service system with inference capabilities. The goal of this system is to assist users in searching movie plots which could be of interest to them. Figure 4.1 shows the architecture of our system. The movie plot search service processes user queries in the following ways:

Firstly, the user issues keywords as a query, which the query parser module can identify as a keyword query before sending it to the formulation module. The query formulation module translates the keywords query into a concept query by using synonym sets. Then the system performs the concept search and presents the initial result. Meanwhile, the intention detector works as an assistant tool and makes a prediction of the user's interests. The user can choose to perform a topic search in terms of the intention detector prediction. Furthermore, the user can choose a movie plot in the previous search result as a fragment-of-text query. Because the fragment-of-text query is more informative and may best reflect the user interests, it can produce a more satisfactory result. We will describe each module of the architecture in detail in this

chapter.

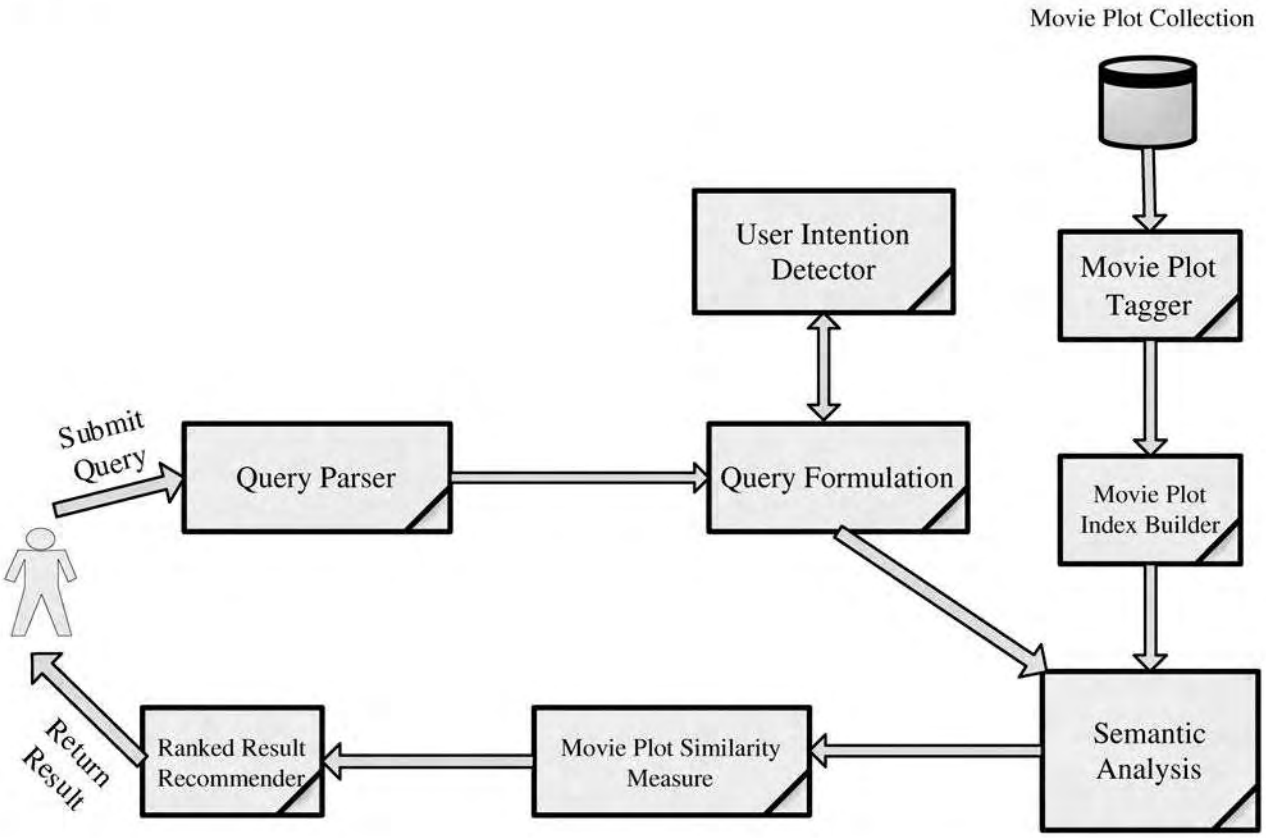


Figure 4.1: Architecture of Intelligent Search Service System

4.2 Query Parser

The query parser is used for identifying the type of query supplied by the user. The user can provide keywords or the name of a movie as the query. The query parser identifies the type of query using the regular expression technique for scanning the list of movie names. If the query string is considered to be keywords, the concept search process is performed. Alternatively, if the string is considered to be a movie name the content search process is performed, in which the movie plot is used as the query itself.

4.3 Movie Plots Tagger

Our system focuses on analyzing the contents of movie plots, without considering some linguistic issues such as the ordering of the terms in the movie plots. For example, although the phrases " Jack loves Kerry" and " Kerry loves Jack" have different meanings, these two phrases refer to the same topic, which is the love between two persons. Generally, users intend to search for similar topics in movie plot collections and rarely care about the exact details of movie plots. In this scenario, noun and verb terms appearing in movie plots are considered to be more valuable than other terms such as adjective and adverb terms. Because movie plots are more concerned with things, events and persons, the movie plots are tagged before the index-building stage. Every noun and verb are highlighted by using a part-of-speech tagger [92]. During the building of the index, noun and verb terms are assigned higher weights.

4.4 Movie Plot Index Builder

The index builder is used for constructing the inverted index which contains meaningful representative terms in the movie plot collection. Each term points to the list of movie plots that contain the term and its frequency in each movie plot. Term frequency is an indication about movie plots' relevance to that term, and is therefore used for measuring movie plot relevance. Each term is stored in the index with statistical information related to their appearance in movie plots. This statistical information is used to compute a weight for each term based on the frequency of a term in a movie plot and in a collection of movie plots. Before building the index, we need to perform

a preprocessing step which includes tokenization, removing stop words and stemming. After the preprocessing, all terms in the collection of movie plots are indexed for further latent semantic analysis as illustrated in Figure 4.2.

4.4.1 Tokenization and Stopword Removal

The first step in the index builder process is the tokenization. Tokenization breaks up movie plots in such way that every individual token is identified and treated as a separate term. It separates compound words based on punctuation marks, abbreviation, and case.

Stopwords are terms that occur very frequently in the movie plots stored in the collection. In our case, stopwords do not carry any useful information. Articles, prepositions, and conjunctions such as "in", "of", "the", etc. are natural candidates for a list of stopwords. Stopword removal can effectively reduce the complexity of the index. Although some stopwords such as "not" may affect the meaning of a movie plot, they do not change the topic of the movie plot.

4.4.2 Stemming

Stemming is the process of removing affixes including prefixes and suffixes. This process unifies the form of the terms in the movie plots and effectively reduces the number of different terms in a movie plot. For instance, "loved", "loving" and "loves" all are converted into their base form "lov". The most well known stemming algorithm is the Porter algorithm [72] which has been integrated into our system.

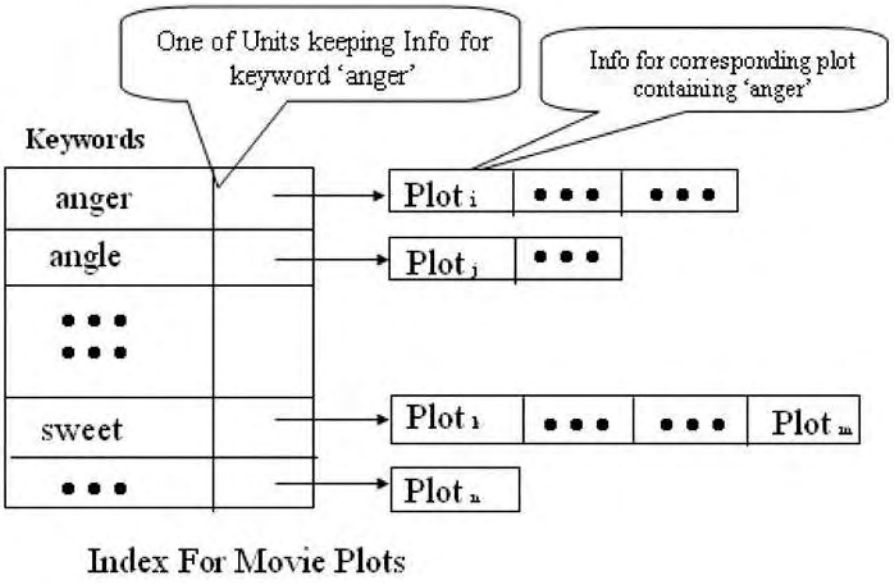


Figure 4.2: Index for Movie Plots

4.5 User Intention Detector

In this section, we describe the construction of the simplified user intention detector implemented in the course of this research. If the system can make predictions for the user information needs, the system will be able to provide a better service to the user. In this scenario, the simplified user intention detector attempts to predict the topics relevant to the user queries. For instance, the user issues a query like "rose birthday present". the intention detector may guess that the user could be interested in the "romance" topic. The inference of the user intention is a difficult task even for human beings and is very challenging for the intention detector. The intention detector in our system is used as an assistant tool to interact with the users. In our viewpoint, if the user accepts the hints from the intention detector, the system can improve the search service in two ways. On one hand the system can efficiently narrow the number of

movie plots to search. In other words, in our example above the system only matches the query with the movie plots related to the "romance" topic, but the movie plot collection must be categorized into different topics in advance. On the other hand, the system can formulate the new query according to the hints provided by the user intention detector. In our system, we prefer to choose the second method.

In this method we firstly build the keywords feature set to describe plot topics. We assume that queries that are related to a specific topic may share some common feature keywords. The query formulation module can construct a new query by means of these common feature keywords. For example, from the query "rose birthday present" it can be implied that the user intends to find some movie plots related to the "romance" topic. If the user accepts the recommendation from the intention detector, the intention detector can send the corresponding feature keywords to the query formulation module. Then the query formulation module can create a new query for example as "rose birthday present date sweet love darling sweetheart honey....". Furthermore, the new query can be sent to the semantic analysis module for semantic processing.

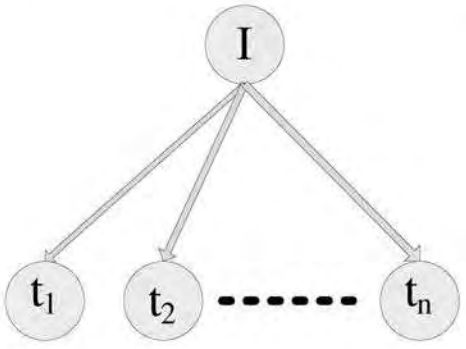


Figure 4.3: Illustration of the Simplified Intention Detector

As mentioned in Chapter 3, the simplified intention detector can be interpreted as a naïve Bayes classifier. Figure 4.3 shows the Bayes model where I represents the user

intention and t_i represents the terms in the user query. The parameters in the Bayes model need to be learnt before the simplified intention detector can be used in the system. In other words, the system must know the values of $P(t_i|state_j)$ and $P(state_j)$ shown in the naïve Bayes classifier Equation 4.1

$$f(q) = \arg \max_{state_j \in I} (\prod_{1 \leq t_i \leq n} P(t_i|state_j) * P(state_j)) \quad (4.1)$$

In the learning stage, we firstly define the states of user intention for topic search such as "Romance" "Action" "Adventure"and "Thriller". Then we need to label the movie plots for each user intention state. For each state, there are typical movie plots which can provide good description for the state. As shown in Algorithm 4.1, the learning process is as follows.

Algorithm 4.1 The Learning Process

At search service initialization
S *represents the set of states*
$state_i$ is element i of state S
D represents *the total number of the Movie* Plots in *the collection*
D_i stands for the number of Movie Plots in $state_i$
T *represents the set of all terms in the Movie Plots*
t_i *is element i of* T
$a_{i,j}$ *is the frequency of term* t_i *in corresponding Movie Plots in* $state_j$.
$B_j = \{a_{i,j}\}$ *is the set of frequency of terms in corresponding Movie Plots of the* $state_j$
For each $state_j$
$P(state_j) = \frac{D_j}{D}$
For each t_i
$P(t_i|state_j) = \frac{a_{i,j}+1}{\sum_{a_{k,j} \in B_j}(a_{k,j}+1)}$

When the topic search is performed, the detector predicts the user's intention in terms of the query. The query is tokenized and the state of intention is inferred by classifier in equation 4.1, where q represents the query.

In our implementation, we have reformulated Equation 4.1 to avoid floating point underflow during program execution. Given that $\log(xy) = \log x + \log y$, Equation 4.1 can be replaced as follows:

$$f(q) = \arg \max_{state_j \in I} (\sum_{1 \leq t_i \leq n} \log(\ P(t_i|state_j)\ ) + \log(\ P(state_j))) \tag{4.2}$$

According to the value of $f(q)$, the system predicts that the user could be interested in the corresponding topic.

4.6 Query Formulation

The query formulation module is used to construct the query for either the concept search, or the topic search.

- Query formulation for concept search

We take advantage of WordNet to construct synonym sets for the original query. WordNet provides the application interface (API) for accessing its relational dictionary. When the query formulation module obtains synonym sets, these are appended to the original query. Let the original query vector be $q =< t_1, t_2, ..., t_m >$ and synonym sets vector is $S =< k_1, k_2, ..., k_n >$, the expanded query will become $q^c =< t_1^c, t_2^c, ..., t_m^c, k_1, k_2, ..., k_n >$ where $t_i^c = \alpha * t_i$ and α is a coefficient which provides a larger weight to the original query vector in the new query. In other words, the original query still has a central role in the new query.

- Query formulation for the topic search

In each user intention state, there exist some representative feature words which can describe the state of user intention in detail. These feature words can be collected and stored as the feature set of the states of user intention. For example, the features of the "Romance" state can be " sweet love dear honey beloved family marry darling date..."; the features of the "Thriller" state can be " scare horrible frighten fear panic anxiety afraid ..."; the features of the "Crime" state can be " violent crime kill gun police murder death...". The intention detector predicts the topic of the user query as an assistant tool, users can choose to perform the topic search from the recommendation of the intention detector. The corresponding feature words are appended to the original query and a new query can be constructed. Let the original query be $q =< t_1, t_2, ..., t_m >$ and the corresponding feature words are $T =< l_1, l_2, ..., l_n >$, the expanded query vector becomes $q^t =< t_1^t, t_2^t, ..., t_m^t, l_1, l_2, ..., l_n >$ where $t_i^t = \alpha * t_i$ and α is coefficient which gives a larger weight to the original query vector in the new query. Therefore, the original query is still highlighted in the new query.

4.7 Semantic Analysis

When the latent semantic analysis processes the inverted index, a matrix X can be established in terms of the inverted index. The columns of this matrix represent the movie plots and the rows of the matrix represent the terms in the matrix. The element a_{ij} stands for term i in the movie plot j.

In order to increase the discriminative power of these terms in the movie plots, we have chosen to implement a combination of local and global weighting mechanisms.

Each element a_{ij} in matrix X is defined as:

$$a_{ij} = \beta * L(i,j) * G(i) \tag{4.3}$$

where $L(i,j)$ is the local weight, which has the same meaning as the term frequency and $G(i)$ is the global weight, which has the same meaning as the inverse-document frequency. β is a coefficient whereby a different part of speech for the terms can be discriminated. In the context of movie plots, the noun and verb terms can represent more semantic meanings than other terms. The local weight is based on the notion that terms that occur frequently in a single movie plot are more valuable than terms that appear infrequently.

$$L(i,j) = tf_{ij} \tag{4.4}$$

where tf_{ij} is the frequency of term i in document j.

The global weight is based on the notion that terms that have a low occurrence frequency over the entire movie plot collection are considered to be more valuable.

$$G(i) = \log_2(\frac{N}{n_i}) \tag{4.5}$$

where G is a weighting and N is the total number of movie plots in the collection and n_i is the number of movie plots containing term i in the movie plots collection.

A similarity measure is performed in the conceptual vector space of the latent semantic analysis. The theory of latent semantic analysis have been described in section 2.3.2.

In the search service system, the query q can be treated as a movie plot when performing a search. Thus the query q must mapped to the conceptual vector space before computing the similarity measure. Given that:

$$X_k = T_k S_k D_k^T \implies S_k^{-1}\ T_k^T\ X_k = D_k^T \implies D_k = X_k^T\ T_k\ S_k^{-1} \tag{4.6}$$

Then:

$$q^* = q^T\ T_k\ S_k^{-1} \tag{4.7}$$

Where q^* is the query q in the conceptual vector space.

For each d_i, we use the cosine measure as similarity metric. Higher ranks assigned to documents mean that the documents are considered more relevant.

During the concept search and topic search, the cosine formula is used to calculate the similarity between expanded query q and movie plot d_i.

$$sim(q^*, d_i) = \frac{q^{*T} X_k e_i}{\|q^*\|_2 \|X_k e_i\|_2} \tag{4.8}$$

where e_i is a canonical vector of a dimension equal to the number of movie plots.

During the content search, the cosine formula is used to calculate the angle between movie plot d_j and movie plot d_i.

$$d_j^* = d_j^T\ T_k\ S_k^{-1} \tag{4.9}$$

$$sim(d_j^*, d_i) = \frac{d_j^{*T} X_k e_i}{\|d_j^*\|_2 \|X_k e_i\|_2} \tag{4.10}$$

where d_j is the movie plot that the user requested. and e_i is a canonical vector of a dimension equal to the number of movie plots.

4.8 Ranked Result Recommender

This module collects the ranked results and outputs the results. It allows users to choose and browse the results. Users can access the search service through Internet browser.

4.9 Chapter Summary

This chapter introduced a search service system with inference capabilities for movie plot search. We developed three types of search services to assist users in the search of information: concept, topic and content search. We investigated the formulation of queries that lead to the satisfaction of the user information needs. Furthermore, we studied the application of latent semantic techniques which can construct semantic document representations to facilitate the search process. In the next chapter, we conduct some experiments to verify our research approaches.

Chapter 5

Experimental Results

In this chapter, we conduct several experiments to demonstrate and verify the approaches proposed in the previous chapters. The aim of the first experiment is to validate our implementation of the latent semantic analysis technique which is a fundamental and important part in our research. The second experiment attempts to illustrate the capabilities of our search scheme applied to a web-based search service system for the movie plot search. In this experiment, the informative query constructed by our proposed method is compared with other two different informative approaches. The third experiment evaluates the performance differences between short query and informative query. Our fourth experiment is an experimental expansion of our approach to the construction of a more generalized intention detector.

5.1 A Visualization Experiment

This experiment aims to validate the effectiveness of the latent semantic analysis which plays an important role in our search service system. It is necessary to confirm that this technique is suitable in our experimental setting. In the initial stage, we start our research from the text semantic analysis technique. Later, we found that the latent

Title	Contents
c1	Human machine interface for ABC computer applications
c2	A survey of user opinion of computer system response time
c3	The EPS user interface management system
c4	System and human system engineering testing of EPS
c5	Relation of user perceived response time to error measurement
m1	The generation of random, binary, ordered trees
m2	The intersection graph of paths in trees
m3	Graph minors IV: Widths of trees and well-quasi-ordering
m4	Graph minors: A survey

Table 5.1: One Experiment for Two Small Document Collections

	c_1	c_2	c_3	c_4	c_5	m_1	m_2	m_3	m_4
Human	1	0	0	1	0	0	0	0	0
Interface	1	0	1	0	0	0	0	0	0
Computer	1	1	0	0	0	0	0	0	0
User	0	1	1	0	1	0	0	0	0
System	0	1	1	2	0	0	0	0	0
Response	0	1	0	0	1	0	0	0	0
Time	0	1	0	0	1	0	0	0	0
EPS	0	0	1	1	0	0	0	0	0
Survey	0	0	0	0	0	0	0	0	1
Trees	0	0	0	0	0	1	1	1	0
Graph	0	0	0	0	0	0	1	1	1
Minors	0	0	0	0	0	0	0	1	1

Table 5.2: The Term-document Matrix

semantic analysis technique performs well on the informative queries. There is a gap between the informative queries and ill-posed short queries. That is our motivation to propose the methods of intention detection. The experiment allows the visualisation of the LSA technique through a simple text collection extracted from Deerwester's work [20]. Documents can be represented by some terms that they contain. These documents can be represented by a numeric term-document matrix. Each row of the matrix corresponds to one term and each column corresponds to one document. Note that the documents from c_1 to c_5 belong to category 1 and the documents from m_1 to m_4 belong to category 2.

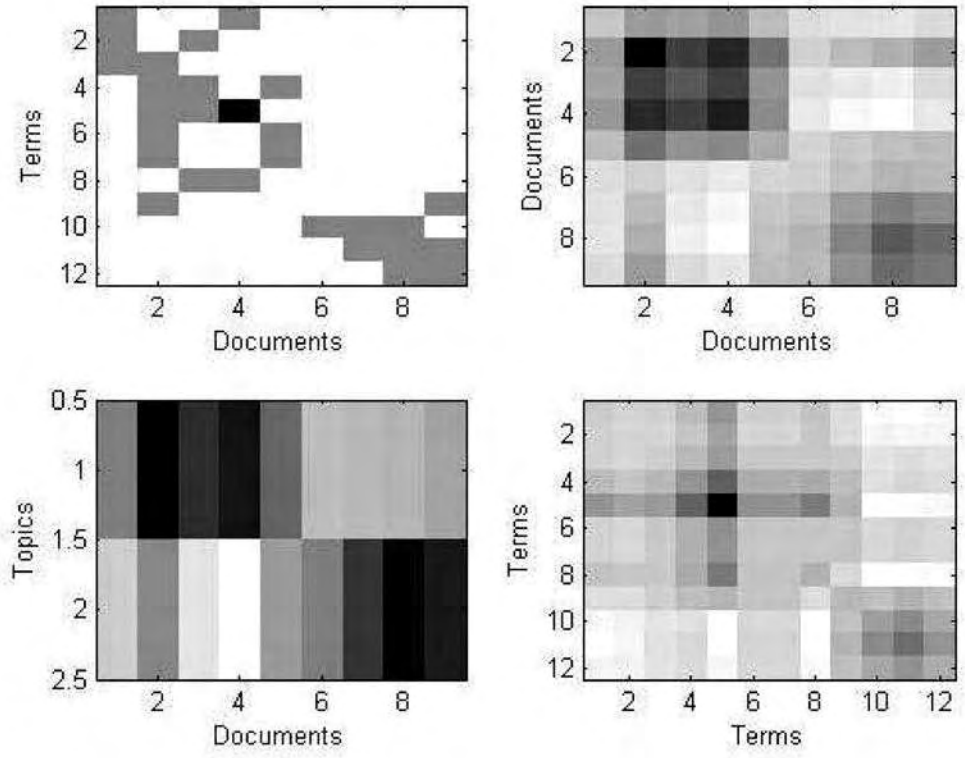


Figure 5.1: Visualization of Latent Semantic Analysis

As illustrated in Figure 5.1, the experimental results show that LSA can group the documents into two clusters which represent two different topics corresponding to the predefined categories. The LSA can differentiate these documents effectively and automatically. Also it can be clearly observed that the LSA can construct the clusters in documents comparison and terms comparison. This experiment shows that related documents or terms move closer in the conceptual vector space.

5.2 An Intelligent Search Service Demonstration

This section presents a search service system with inference capabilities, which is built on the Webobject platform for the movie plot search. This experiment has two main purposes. Firstly, the experiment is designed to compare three different informative query formulation approaches (i) the concept search, which is based on the synonym expansion (ii) the topic search, which is based on the proposed simplified intention detector, and (iii) the content search, which is performed in terms of the search results obtained by the two previous approaches. The informative query formulated by the topic search performs better than the one formulated by the concept search. The content search achieves the best result. Secondly, we show how our search service system performs search processes through the interaction with the user. The system comprises the movie plot collection which is composed of unstructured text. In this experiment, there are 586 movie plots which are categorized into eight different topics. Note that one movie can have different movie plots written by different movie reviewers. This aspect of the movie plot collection can be used to verify the effectiveness of our system.

The three different query formulation approaches are implemented as follows. Firstly, we issue keywords as a short query processed by the query parser as discussed in Section 4.2. Then the system proceeds to the concept search through the generation of synonym sets in the query formulation module as mentioned in Section 4.6. We found that the terms of the concept query may not be closely related to each other to reflect the user information needs. This informative query approach is used for the first search step and later subjected to further processing. Subsequently, the similarity matching proceedes as described in Section 4.7. The results of this experiment show the relatively poor performance of similarity matching with the semantic presentation of movie plots which is processed by the movie plot index builder as described in Section 4.4. The second query approach used consisted of performing the topic search through constructing the informative query in the query formulation module as mentioned in Section 4.6 from the recommendation of the intention detector, as presented in Section 4.5. Because this informative query can be more specific to the user interests and the terms of the topic search are typically higher inter-related than those used in the concept search experiment, the topic search can produce better performance on similarity matching with semantic presentation of movie plots. This was verified by the experiments. The third approach is called the content search through user selection from the previous search results as described in Section 4.7. Namely, users can browse the contents of movie plots in previous search results and choose movie plots as new queries. The movie plot selected by the user can provide a detailed description of the user information need. As movie plots are written by human beings, the terms in the movie plots tend to be coherent and highly relevant to the user search. The content

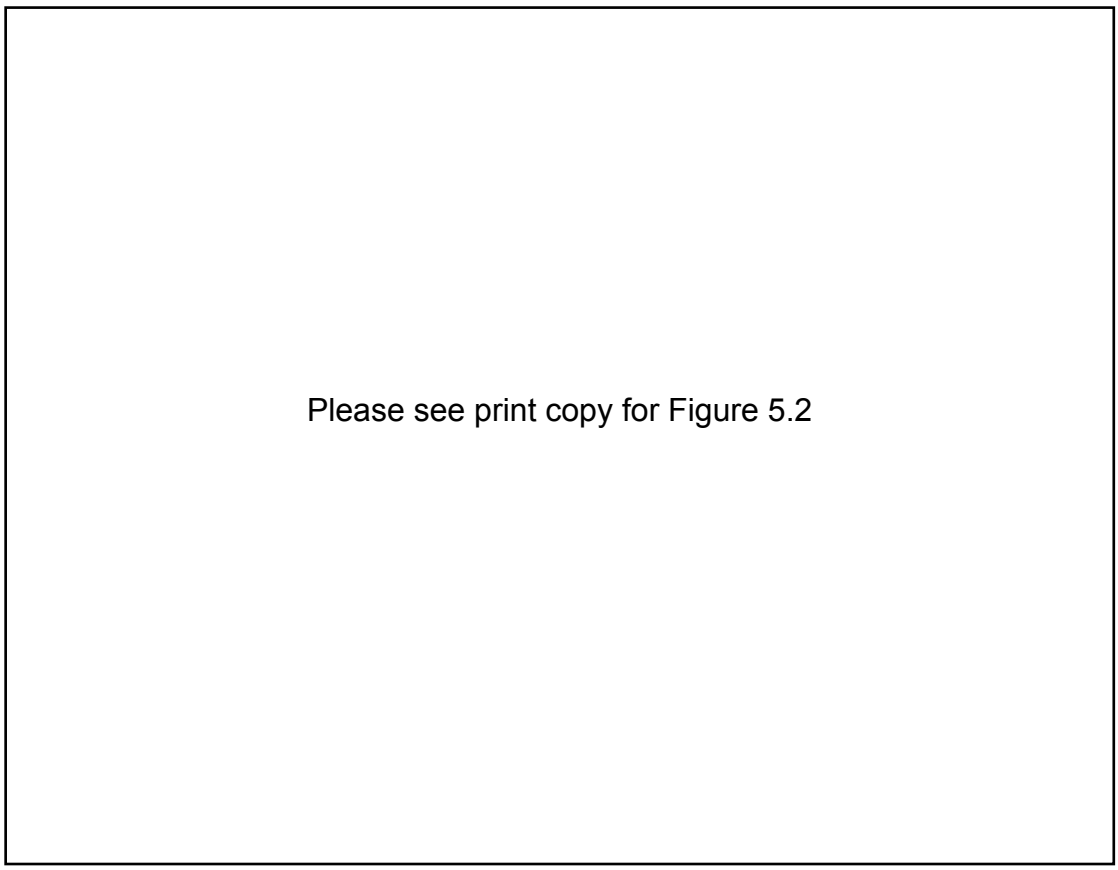


Figure 5.2: The Initial Search User Interface

search has the potential to obtain the best performance by virtue of using previous search results in the reformulation of user queries. The experiment also shows that it is still difficult to construct appropriate informative queries which can accurately reflect the user information need. Through the user interaction, it is feasible to guide the user to construct the informative query step by step. All the search approaches are performed on the search service system as presented in Chapter 4.

The search process of the system works as follows. Firstly, a user enters a short

query into the search service system as shown in Figure 5.2. In this case, "beauty girl" is typed into the search box. The query parser determines if the input intends to be a concept search or a content search. If the input is identified as keywords, the system performs the concept search process. The keywords are transformed into synonym sets in the query formulation module. Meanwhile, the intention detector makes prediction for the user initial query as a state of intention. The semantic analysis module maps synonym sets into the conceptual vector space. The similarity measure module is used to measure the similarity between the informative query and the movie plots, then it ranks the search results and transfers the ranked results to the recommender module.

Figure 5.3 shows the result of the concept search. In this example, the short query is "beauty girl" , after the query formulation, the informative query becomes "beauty girl lady sweetheart miss female daughter girlfriend ...." which is a synonym set of the original keywords. The relevance score on the right column of the table presents the degree of the similarity matching. The distribution of scores shows that the concept search cannot perform high similarity matching with the movie plots. The informative query constructed by synonym sets can represent the general meaning of the keywords, but it lacks the inference capability and isn't specific to the information need. In other words, the concept search cannot generate coherent content for the informative query.

Meanwhile, the intention detector predicts that the user could intend to seek some movie plots related to the "romance" topic. The system can provide the intention detector (shown above the result table in Figure 5.3) which allows the user to perform topic search. The user can accept the intention detector and choose topic search for further processing. When the user clicks the hyperlink recommended by the intention

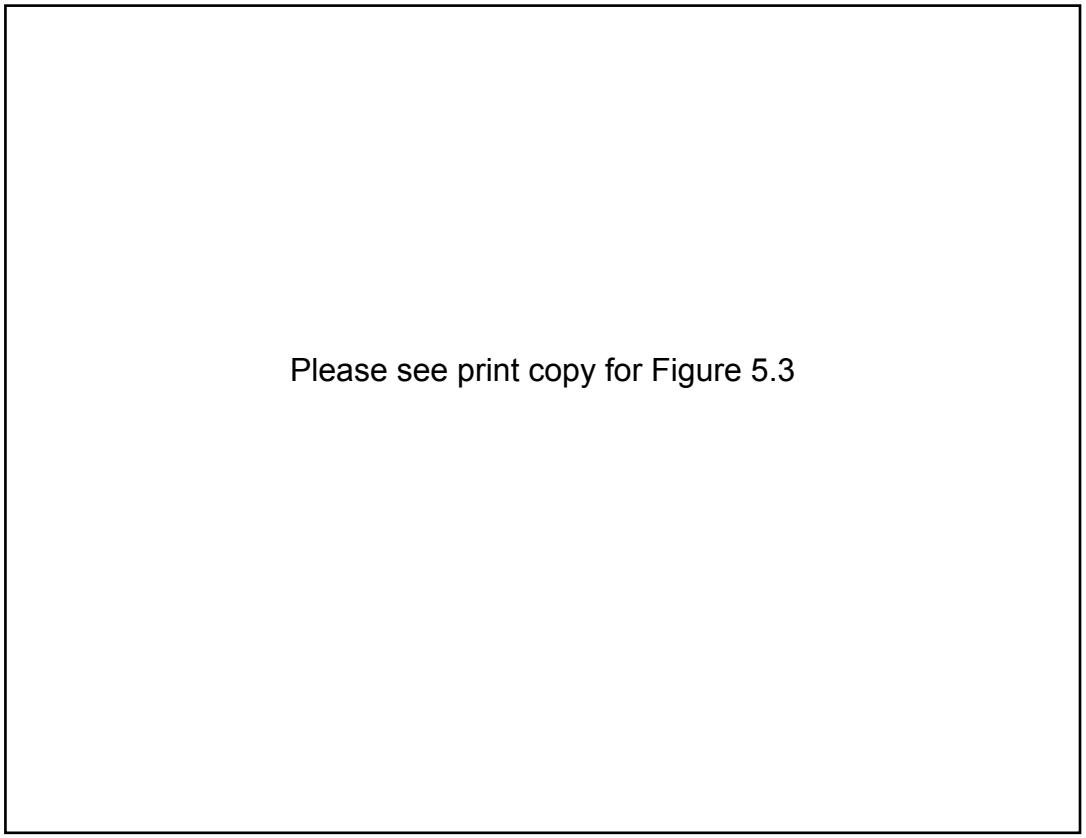


Figure 5.3: The Concept Search Results

detector, the topic search is performed. The informative query becomes " beauty girl sweet love dear honey beloved family marry darling date ...".

Figure 5.4 shows the result of topic search. The distribution of scores indicates that the similarity matching of the topic search is slightly better than that of concept search. The informative query of the topic search can also be more specific to the user information need.

The user can read the content of movie plots presented by the recommender module.

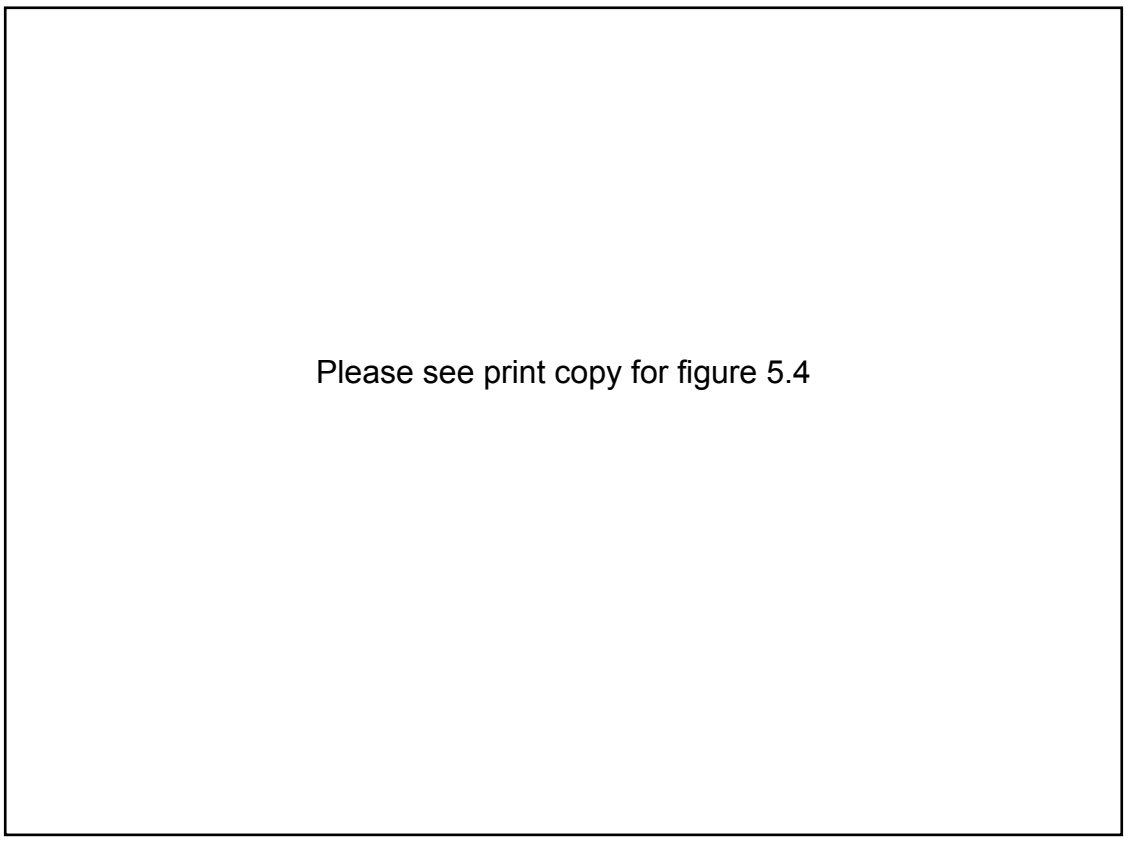


Figure 5.4: The Topic Search Results

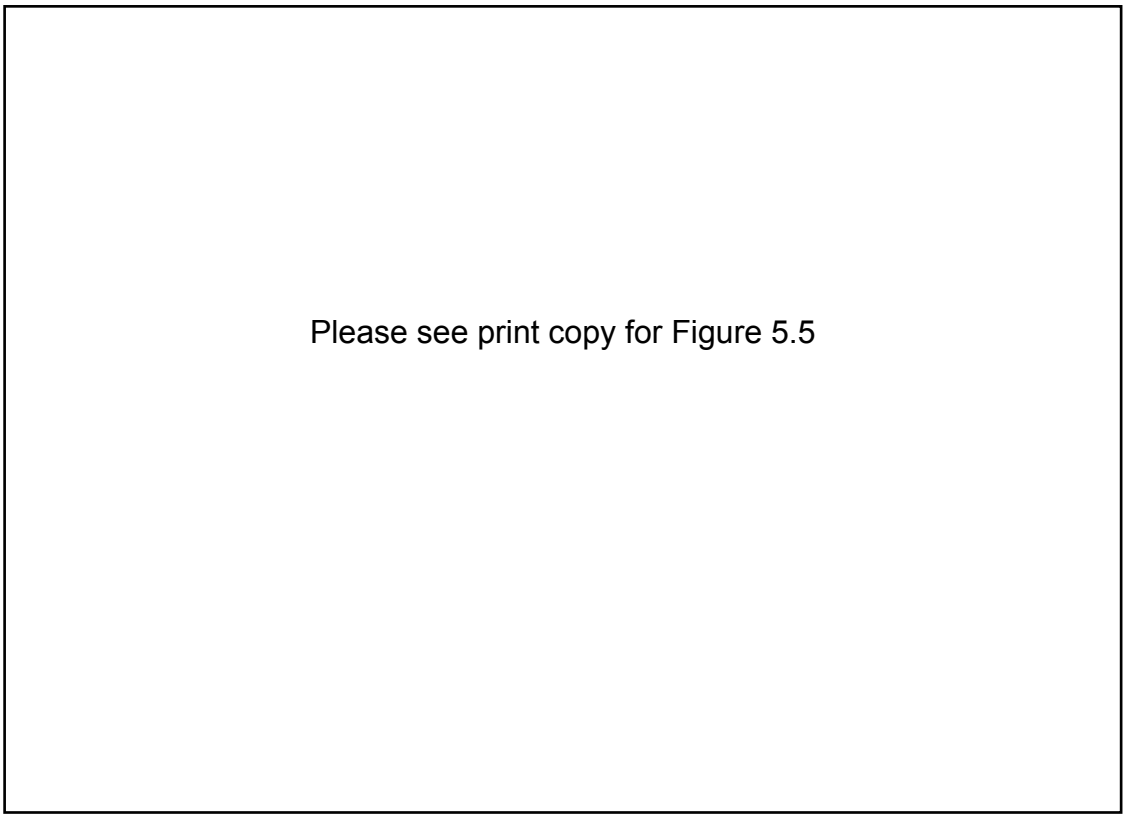


Figure 5.5: The Content Search Results

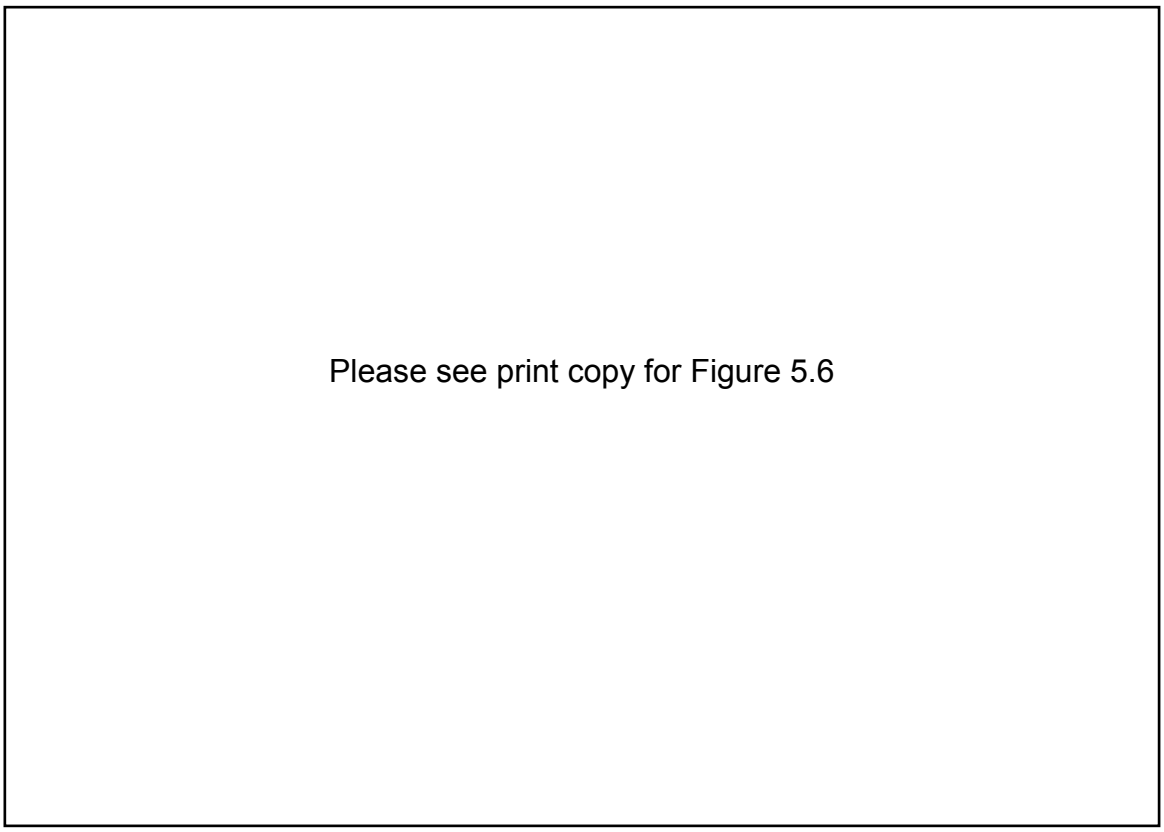


Figure 5.6: The Content Search Results

The movie plot result is displayed in a hyperlink list created by the ranked result recommender module. The hyperlink list displays the name of the movie plot with an associated link. To get a better understanding of the movie plots, the user can select one hyperlink, then the content of the corresponding movie plot is returned from the search service system. In this example, suppose that the user is interested in a movie plots such as "leon_d", The movie plot "leon_d" can be chosen as a content search through the search box. When the content search is performed, the search service system uses the content of this movie plot as the informative query. The semantic analysis module can map this informative query into the conceptual vector space and measure similarity within movie plots. Figure 5.5 shows the result of content search. The distribution of scores indicates that the similarity matching of the content search has the best performance. The highly related movie plots "leon_a", "leon_b" and "leon_c" which are descriptions of the same movie written by different persons can be found at the top of the ranking list. Similarly, when the movie plot "manhattan_a" is chosen as a content search, the system performs well as shown in Figure 5.6, where the two "manhattan" plots are at the top of the table. When the user chooses the movie plot as an informative query, because the plot was written by a human being, the content of this informative query is coherent and natural, and it can produce the sound result observed.

5.3 An Evaluation Experiment

There are two effective metrics which can measure the performance of a search service system. These are precision and recall.

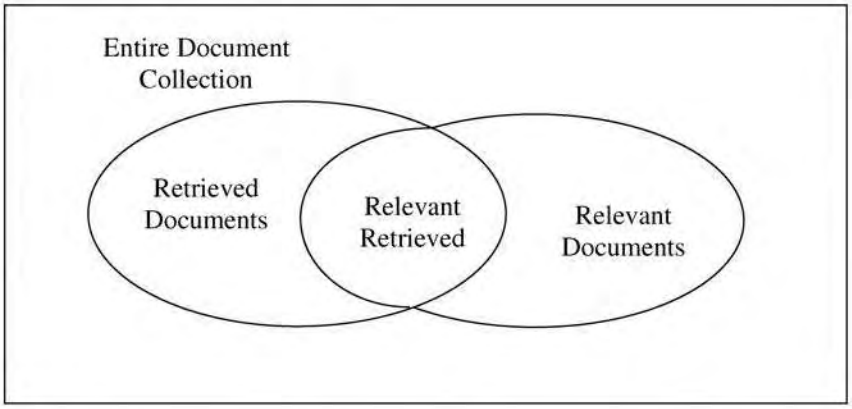


Figure 5.7: The Precision and Recall

As illustrated in Figure 5.7, precision is the ratio of relevant retrieved documents to the total number of documents retrieved. In contrast recall is the ratio of relevant documents retrieved to all relevant documents in the whole collection. The quantitative process is depicted by Equation 5.1.

$$\begin{aligned} \mathrm{Pr}\,ecision &= \frac{\mathrm{Re}\,levant\ \mathrm{Re}\,trieved\ \mathrm{Documents}}{\mathrm{Re}\,trieved\ \mathrm{Documents}} \\ \mathrm{Re}\,call &= \frac{\mathrm{Re}\,levant\ \mathrm{Re}\,trieved\ \mathrm{Documents}}{\mathrm{Re}\,levant\ \mathrm{Documents}} \end{aligned} \tag{5.1}$$

In Section 5.2, we illustrate three types of informative query including concept search, topic search and content search. In order to study the effectiveness of informative query, we conduct our experiments on the Time dataset [52] which collects short articles from the Time Magazine. There are 423 documents in the dataset which are not categorized. The experiment is conducted on our system with the concept and topic search functions disabled because the dataset is not for the specific domain. The content search can be used as the informative query in the experiment. In this experiment, TF-IDF short query is the simple keyword search using the TF-IDF method;

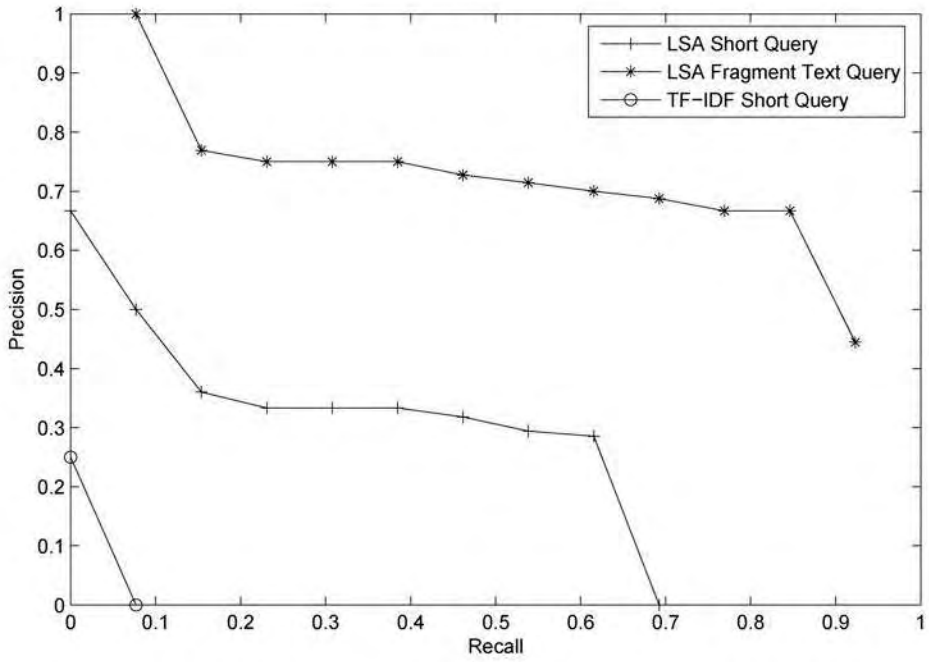


Figure 5.8: The Precision and Recall Results

LSA short query is the simple keyword search with latent semantic analysis; LSA fragment text query is the content search in which the informative query is constructed from the initial result of the LSA short query. This experiment attempts to evaluate the difference between short query and informative query.

As it can be seen in Figure 5.8, we compare the performance among the TF-IDF short query, LSA short query and LSA fragment text query. When the user submits the short query to the search service system, the semantic values in TF-IDF short query method are shallow in document representations. Moreover, the search service system cannot capture information need due to the short query. Therefore, the precision and recall are extremely low. The precision only reaches about 25% and less than 10% of the relevant documents can be found. In the LSA short query method, the performance

can be better. The main reason is that the system can maintain semantic document representations. The precision is stable at around 35%. More than 60% of relevant documents can be found. In LSA fragment text query, the query is informative and the system has the semantic document representations. Thus, the precision is stable at 70% and more than 90% of relevant documents can be found.

5.4 Some Experiments About User Queries

Previous experiments show that the quality of queries can strongly affect search results. In spite of building the semantic index for documents, more efforts need to be made for the effective match between documents and user queries. In our experiences, the query process has two important characteristics in terms of user search behaviors. First, the length of user query is short. Second, users would like to submit several queries to clarify their vague intention through interacting with the search service. In this experiment, we design more experiments to investigate the user queries' issues.

The experiments consist of three parts which include query set generation, an intention detector and query formulation.

In first part, the query set in real world is valuable. There are few query set freely available to the public. Before constructing an intention detector, we need to generate the simulated query set for further processing. The process is described below.

1. Collect the data set about movie plots and label movie plots based on their topic. Currently we assume that user intention is only related to the topic.

2. Clean noisy terms through removing stopwords.

3. Calculate the posterior probability for each keyword in each topic from labelled

movie plots.

$$posterior(t) = P(t|c_j) * P(c_j) \tag{5.2}$$

where $P(t|c_j)$ is the conditional possibility for each word in each topic, and $P(c_j)$ is the prior probabilities.

4. Sort the keywords based on the value of the posterior probability.

5. Find the representative keywords from the top rank of keywords for each topic. The representative keywords is the features which can sketch the properties of each topic.

6. Create the pool of query words through randomize the representative keywords.

7. Generate the sequence of simulated queries from the pool of query words.

As shown in Table 5.3, the result of the experiment indicates that some terms are highly correlated with several different topics. For example, the word "love" is closely related to the topic "Comedy", "Romance" and "Drama"; the word "blood" is closely related to the topic "Thriller", "Crime" and "Mystery". The word "love" or "blood" is submitted as the user query, it is difficult to effectively identify the user query as one corresponding state of intention. For one query, it can be identified as several possible intention states. The simplified intention detector cannot handle this complicated situation, which is its salient limitation. It is the better way to analyze the query in the context of previous queries.

In the second part, we designed an intention detector based on the first order hidden Markov chain. After the simulated data generation process, a set of query sequence

Comedy		Thriller		Crime		Adventure	
life	0.0414	alien	0.0264	life	0.0499	time	0.0209
love	0.0386	planet	0.0202	sword	0.0394	gold	0.0177
dream	0.0303	blood	0.0124	jail	0.0394	machine	0.0161
man	0.0303	ship	0.0123	prison	0.0342	outlaw	0.0145
story	0.0302	power	0.0118	gun	0.0289	power	0.0145
christmas	0.0301	violence	0.0109	father	0.0263	farmer	0.0113
family	0.0276	greed	0.0107	blood	0.0261	greed	0.0112
hope	0.0248	traitor	0.0103	lives	0.0237	army	0.0112
home	0.0246	space	0.0102	murder	0.0230	future	0.0110
time	0.0221	blackmail	0.0100	cult	0.0210	find	0.0109

Romance		Drama		Mystery		Action	
love	0.0789	death	0.0442	identity	0.0423	batman	0.0308
happy	0.0363	hope	0.0326	legend	0.0423	kill	0.0280
woman	0.0341	friend	0.0307	beautiful	0.0421	battle	0.0252
beautiful	0.0341	father	0.0288	secret	0.0420	revenge	0.0252
angel	0.0277	gift	0.0250	accident	0.0370	army	0.0251
eve	0.0277	war	0.0249	robbery	0.0370	man	0.0251
wonderful	0.0235	friendship	0.0230	blood	0.0369	emperor	0.0250
french	0.0213	story	0.0211	gun	0.0369	death	0.0249
wife	0.0213	family	0.0210	detective	0.0368	fight	0.0247
home	0.0149	love	0.0192	murder	0.0367	war	0.0246

Table 5.3: Top Ten Most Representative Words

and associated probability function are obtained. Then an intention detector can be designed. The design of a intention detector consists of two parts. One is to find the pattern of different intentions via query samples. This process is called the learning stage. The other part is to infer the user intention by submitting new query sequences. This process is called the inference stage.

Each intention is characterized by the state transition probabilities which is estimated in the learning stage. In the initialization, the queries can be assigned initial

values to eight different states of intention. The user is more likely to stay at the previous intention state. In the case of the eight states, we can assume that the user has 65% chance to stay at previous stage and 5% chance to jump to other state. After the learning stage, we can find out the associations between different states of intention.

We found that the experiment is difficult to conduct at the current stage. We tend to build a more advanced intention detector by using the simulated sequence of user queries. The size of the simulated sequence of user queries is so small that the model cannot learn. Furthermore, there is no real user intentions in the simulated query dataset. The query set in the real world is required in order to present the convincing results.

In third part, the new query is constructed through the semantic similarity measurement among the old query and the feature words in the corresponding state. As described in the Section 3.5, we use the information content(IC) approaches to seek the related words to formulate new query. In one example, when the user submits the query "break prison", the user intention detector has identified the user query as the "Crime" topic. There are 20 related words that depict the property of the "Crime" topic. We need to seek the most related words among these words and formulate the query. As can be seen from Table 5.4, top three related words can be selected via Jiang's method and formulate the new query " break prison jail life escape rape murder" to reflect the user information need. In Table 5.5, top three words can be selected via Lin's method and formulate the new query " break prison jail life escape rape murder" to meet user information need. Although the rank of those related words is slightly different between Jiang's and Lin's methods, the selection of most related

words can obtain the very similar result.

Features	Query	Jiang's Method	Features	Query	Jiang's Method
prison	prison	1.0000	escape	break	0.4563
jail	prison	0.5206	rape	break	0.1579
life	prison	0.0894	murder	break	0.1149
gun	prison	0.0829	blood	break	0.0937
father	prison	0.0806	life	break	0.0904
drug	prison	0.0768	crime	break	0.0881
addiction	prison	0.0763	father	break	0.0832
sword	prison	0.0763	loyalty	break	0.0824
lawyer	prison	0.0698	corruption	break	0.0823
corruption	prison	0.0671	gun	break	0.0804
escape	prison	0.0662	police	break	0.0778
blood	prison	0.0652	offense	break	0.0771
crime	prison	0.0621	drug	break	0.0760
police	prison	0.0613	prison	break	0.0726
murder	prison	0.0579	jail	break	0.0697
rape	prison	0.0574	addiction	break	0.0685
offense	prison	0.0564	cult	break	0.0637
loyalty	prison	0.0546	lawyer	break	0.0617
cult	prison	0.0506	sword	break	0.0570
illegal	prison	0.0000	illegal	break	0.0000

Table 5.4: Find Most Related Words Via Jiang's Method

This section is an experimental expansion to construct a more generalized intention detector. We constructed an intention detector based on a hidden Markov model. However, intention is in the human mind and therefore prediction of user intention is subjective. For this reason it is difficult to find a suitable data set of user queries for our experiments. Real-world user search data are not usually available to the public; these valuable data are maintained and accessed by large commercial websites. If we can collaborate with the professional websites, user queries can be collected for a particular domain. Then, for each sequence of user queries, the user intentions can be identified and interpreted by humans. Some domain knowledge could be used. This process is very costly and requires a lot of human efforts. A valuable feature introduced in our approach is the selection of representative words based on the semantic similarity

Features	Query	Lin's Method	Features	Query	Lin's Method
prison	prison	1.0000	escape	break	0.8934
jail	prison	0.8988	rape	break	0.6898
life	prison	0.3820	murder	break	0.5008
father	prison	0.3579	blood	break	0.4498
addiction	prison	0.3454	gun	break	0.4473
corruption	prison	0.3170	corruption	break	0.4262
gun	prison	0.2995	loyalty	break	0.3724
sword	prison	0.2823	life	break	0.3527
escape	prison	0.2546	father	break	0.3360
lawyer	prison	0.1534	prison	break	0.3342
blood	prison	0.1353	addiction	break	0.3213
rape	prison	0.1296	crime	break	0.3027
drug	prison	0.0970	cult	break	0.2825
murder	prison	0.0000	offense	break	0.2753
cult	prison	0.0000	drug	break	0.0888
police	prison	0.0000	lawyer	break	0.0766
loyalty	prison	0.0000	sword	break	0.0713
crime	prison	0.0000	jail	break	0.0688
illegal	prison	0.0000	police	break	0.0000
offense	prison	0.0000	illegal	break	0.0000

Table 5.5: Find Most Related Words Via Lin's Method

measure.

5.5 Chapter Summary

This chapter described several experiments, which validate and expand our approach and implementation.

Chapter 6

Conclusions and Future Work

A search service system with inference capabilities should be capable of capturing the semantic values in the documents collection and comprehend user information need in terms of user queries. However, conventional search services existing today lack the power of inference capabilities to detect user intention. In this thesis, we explored certain feasible approaches to improve the quality of search service. The objective of this thesis is to investigate some text search issues which can allow the computer system to better understand user information need. We proposed a generalized intention detection framework to infer user information need. After that we implemented a demonstration prototype to verify our study. New query formulation approaches are presented, which effectively process user requests expressed in the form of a short query. The Bayesian inference and semantic similarity measurement are applied in our approach. In our experiment, we examine the effectiveness of semantic document representations and validate the different query formulations based on synonym set and related feature words. We conclude that informative queries and semantic document representations are two important factors which influence search results. We also show that the intention detector is applicable to constructing the informative query.

There is significant potential for future improvements to the search service system, which require the availability of real world query set in a specific domain. When real world query set can be obtained, more improvements could be achieved by exploring the pattern of user intentions in the query set. Collaborating with professional web sites to obtain raw data of user queries in the real world, the aim is to find meaningful patterns in the past user behavior, which can be used to predict future user intentions in a specific domain. We can conduct more advanced experiments to make predictions for user intention. In the thesis, we present a basic proof-of-concept search service system. Much work remains to be done in exploring the intention of user queries in a search service. More advanced approaches are required for capturing a user's intention. Further improvement should be achieved by introducing hidden Markov models using real world user queries. There is a need to consider different domain problems and multiple types of intentions. There is also a need to consider more semantic features which may represent some information about the user intention in the query. It might be helpful to utilize information from the user profile provided by the user.

Bibliography

[1] Ricardo A. Baeza-Yates. Applications of Web Query Mining. pages 7–22, 2005.

[2] P.F. Baldi, P. Frasconi, and P. Smyth. *Modeling the Internet and the Web: Probabilistic Methods and Algorithms.* John Wiley and Sons, 2003.

[3] MJ Bates. An exploratory paradigm for online information retrieval. Intelligent Information Systems for the Information Society. *Proceedings of the 6th International Research Forum in Information Science*, pages 91–99, 1986.

[4] Jerome R. Bellegarda. Latent Semantic Mapping. *IEEE Signal Processing Magazine*, 70, 2005.

[5] Yoav Benjamini and Moshe Leshno. Statistical Methods for Data Mining. In *The Data Mining and Knowledge Discovery Handbook*, pages 565–587. 2005.

[6] Berry, Dumais, and O'Brien. Using Linear Algebra for Intelligent Information Retrieval. volume 37, 1995.

[7] M.W. Berry and M. Browne. *Understanding Search Engines: Mathematical Modeling and Text Retrieval.* Society for Industrial & Applied, 1999.

[8] C.M. Bishop. *Pattern recognition and machine learning.* Springer, 2006.

[9] K.D. Bollacker, S. Lawrence, and C.L. Giles. CiteSeer: an autonous Web agent for automatic retrieval and identification of interesting publications. *Proceedings of the second international conference on Autonomous agents*, pages 116–123, 1998.

[10] S. Brin and L. Page. The Anatomy of a Large-Scale Hypertextual Web Search Engine. *WWW7 / Computer Networks*, 30(1-7):107–117, 1998.

[11] S. Brin, L. Page, R. Motwami, and T. Winograd. The PageRank citation ranking: bringing order to the web. *Proceedings of ASIS'98*, pages 161–172, 1998.

[12] S Chakrabarti. *Mining the Web: Discovering Knowledge from Hypertext Data.* Morgan Kaufmann, 2002.

[13] S. Chakrabarti, B.E. Dom, D. Gibson, J. Kleinberg, R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Mining the Link Structure of the World Wide Web. *IEEE Computer*, 32(8):60–67, 1999.

[14] H. Chen and S. Dumais. *Bringing order to the Web: automatically categorizing search results.* ACM Press New York, NY, USA, 2000.

[15] K. Church, M.T. Keane, and B. Smyth. Towards More Intelligent Mobile Search. *Proceedings of the conference IJCAI-05*, 2005.

[16] N. Craswell, F. Crimmins, D. Hawking, and A. Moffat. Performance and cost tradeoffs in Web search. *Proceedings of the fifteenth conference on Australasian database-Volume 27*, pages 161–169, 2004.

[17] Nello Cristianini, John Shawe-Taylor, and Huma Lodhi. Latent Semantic Kernels. *Journal of Intelligent Information Systems*, 18(2/3):127–152, 2002. Special Issue on Automated Text Categorization.

[18] D.R. Cutting, D.R. Karger, J.O. Pedersen, and J.W. Tukey. Scatter/Gather: a cluster-based approach to browsing large document collections. pages 318–329. ACM Press New York, NY, USA, 1992.

[19] H. Davalcu, S. Vadrevu, S. Nagarajan, and IV Ramakrishnan. OntoMiner: bootstrapping and populating ontologies from domain-specific Web sites. *Intelligent Systems, IEEE*, 18(5):24–33, 2003.

[20] S. Deerwester, S.T. Dumais, G.W. Furnas, T.K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

[21] S. Dennis, P. Bruza, and R. McArthur. Web searching: A process-oriented experimental study of three interactive search paradigms. *Journal of the American Society for Information Science and Technology*, 53(2):120–133, 2002.

[22] I.S. Dhillon and D.S. Modha. Concept Decompositions for Large Sparse Text Data Using Clustering. *Machine Learning*, 42(1):143–175, 2001.

[23] C.H.Q. Ding. A similarity-based probability model for latent semantic indexing. pages 58–65. ACM Press New York, NY, USA, 1999.

[24] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification.* John Wiley & Sons Press, 2001.

[25] S. Dumais, M. Banko, E. Brill, J. Lin, and A. Ng. Web question answering: is more always better? *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 291–298, 2002.

[26] S.T. Dumais. Latent semantic indexing (LSI): TREC-3 report. volume 219, 1994.

[27] ST DUMAIS. Latent semantic indexing(LSI) and TREC-2. pages 105–115. National Institute of Standards and Technology, 1994.

[28] ST Dumais, GW Furnas, TK Landauer, S. Deerwester, and R. Harshman. Using latent semantic analysis to improve access to textual information. pages 281–285. ACM Press New York, NY, USA, 1988.

[29] Susan Dumais, Edward Cutrell, Raman Sarin, and Eric Horvitz. Implicit queries (IQ) for contextualized search. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2004.

[30] Susan T. Dumais. LSI meets TREC: A status report. In *TREC*, pages 137–152, 1992.

[31] L. Eikvil. Information Extraction from World Wide Web-A Survey. *Norwegian Computing Center*, 114, 1999.

[32] Steve Fox, Kuldeep Karnawat, Mark Mydland, Susan Dumais, and Thomas White. Evaluating implicit measures to improve web search. *ACM Transactions on Information Systems*, 23(2):147–168, 2005.

[33] W.B. Frakes and R. Baeza-Yates. *Information retrieval: data structures and algorithms.* Prentice-Hall, Inc. Upper Saddle River, NJ, USA, 1992.

[34] Kevin R. Gee. Using latent semantic indexing to filter spam. In *SAC '03: Proceedings of the 2003 ACM symposium on Applied computing*, pages 460–464, 2003.

[35] A. Gulli and A. Signorini. The indexable web is more than 11.5 billion pages. *International World Wide Web Conference*, pages 902–903, 2005.

[36] Jiawei Han and Micheline Kamber. *Data Mining: Concept and Techniques.* Morgan Kaufmann, 2001.

[37] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning.* Springer, 2003.

[38] R. Herbrich and T. Graepel. Introduction to the Special Issue on Learning Theory. *Journal of Machine Learning Research*, 4(5):755–757, 2004.

[39] Thomas Hofmann. Probabilistic Latent Semantic Analysis. In *Uncertanity in Artificial Intelligence*, 1999.

[40] AK Jain, MN Murty, and PJ Flynn. Data clustering: a review. *ACM Computing Surveys (CSUR)*, 31(3):264–323, 1999.

[41] B.J. Jansen and U. Pooch. A review of Web searching studies and a framework for future research. *Journal of the American Society for Information Science and Technology*, 52(3):235–246, 2001.

[42] F.V. Jensen. *Bayesian networks and decision graphs.* Springer, 2001.

[43] J. Jiang, MW Berry, JM Donato, G. Ostrouchov, and NW Grady. Mining consumer product data via latent semantic indexing. *Intelligent Data Analysis*, 3(5):377–398, 1999.

[44] J.J. Jiang and D.W. Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. *Proceedings of International Conference on Research in Computational Linguistics*, pages 19–33, 1997.

[45] Rong Jin and Susan Dumais. Probabilistic combination of content and links. pages 402–403, 2001.

[46] Thorsten Joachims. Text Categorization with Suport Vector Machines: Learning with Many Relevant Features. In *ECML*, pages 137–142, 1998.

[47] K.S. Jones, S. Walker, and SE Robertson. A probabilistic model of information retrieval: development and comparative experiments. *Information Processing and Management: an International Journal*, 36(6):779–808, 2000.

[48] M.I. Jordan. *Learning in Graphical Models.* MIT Press, 1999.

[49] D. Jurafsky and J.H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition.* MIT Press, 2000.

[50] B.Y. KANG and S.J. LEE. Document indexing: a concept-based approach to term weight estimation. *Information Processing and Management*, 41:1065–1080, 2005.

[51] J.M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, 46(5):604–632, 1999.

[52] R. Krovetz and W.B. Croft. Lexical ambiguity and information retrieval. *ACM Transactions on Information Systems (TOIS)*, 10(2):115–141, 1992.

[53] D. Laham. Latent Semantic Analysis approaches to categorization. *Proceedings of the 19th Annual Conference of the Cognitive Science Society*, 1997.

[54] Foltz P. W. & Laham D. Landauer, T. K. Introduction to Latent Semantic Analysis. *Discourse Processes*, 25:259–284, 1998.

[55] S.L. Lauritzen. *Graphical models: Clarendon Press.* Clarendon Press, 1996.

[56] S. Lawrence, K. Bollacker, and C.L. Giles. Indexing and retrieval of scientific literature. *Proceedings of the eighth international conference on Information and knowledge management*, pages 139–146, 1999.

[57] D.D. Lewis. An evaluation of phrasal and clustered representations on a text categorization task. pages 37–50. ACM Press New York, NY, USA, 1992.

[58] D.D. Lewis and K.S. Jones. Natural language processing for information retrieval. *Communications of the ACM*, 39(1):92–101, 1996.

[59] H. Lieberman. Letizia: An agent that assists web browsing. pages 924–929, 1995.

[60] D. Lin. An information-theoretic definition of similarity. *Proceedings of the 15th International Conference on Machine Learning*, pages 296–304, 1998.

[61] J. Lu and J. Callan. Content-based retrieval in hybrid peer-to-peer networks. *Proceedings of the twelfth international conference on Information and knowledge management*, pages 199–206, 2003.

[62] D.J.C. MacKay. *Information Theory, Inference and Learning Algorithms.* Cambridge University Pr., 2003.

[63] G.A. Miller. WordNet: A Lexical Database for English. *COMMUNICATIONS OF THE ACM*, 38:11–39, 1995.

[64] T Mitchell. *Machine Learning.* McGraw Hill, 1997.

[65] M. Mitra, C. Buckley, A. Singhal, and C. Cardie. An analysis of statistical and syntactic phrases. volume 97, pages 200–214, 1997.

[66] J. Morris and G. Hirst. Lexical cohesion computed by thesaural relations as an indicator of the structure of text. *Computational Linguistics*, 17(1):21–48, 1991.

[67] R.E. Neapolitan. *Learning Bayesian Networks.* Pearson Prentice Hall Upper Saddle River, NJ, 2004.

[68] H.T. Ng and H.B. Lee. Integrating multiple knowledge sources to disambiguate word sense: An exemplar-based approach. *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, 47, 1996.

[69] B. Pang, L. Lee, and S. Vaithyanathan. Thumbs up?: sentiment classification using machine learning techniques. pages 79–86. Association for Computational Linguistics Morristown, NJ, USA, 2002.

[70] LD Paulson. Search Technology Goes Mobile. *Computer*, 38(8):19–22, 2005.

[71] T. Pedersen, S. Patwardhan, and J. Michelizzi. Wordnet:: Similarity-Measuring the relatedness of concepts. *Demonstrations of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 2–7, 2004.

[72] M.F. Porter. An algorithm for suffix stripping. 1980.

[73] J. Qin, Y. Zhou, and M. Chau. Building domain-specific Web collections for scientific digital libraries: a meta-search enhanced focused crawling method. *Digital Libraries, 2004. Proceedings of the 2004 Joint ACM/IEEE Conference on*, pages 135–141, 2004.

[74] LR Rabiner. A tutorial on hidden Markov models and selected applications inspeech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[75] P. Resnik. Using information content to evaluate semantic similarity in a taxonomy. *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, 1:448–453, 1995.

[76] S.E. Robertson and K.S. Jones. Relevance weighting of search terms. *Taylor Graham Series In Foundations Of Information Science*, pages 143–160, 1988.

[77] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz. A Bayesian approach to filtering junk e-mail. volume 62. Madison, Wisconsin: AAAI Technical Report WS-98-05, 1998.

[78] G. Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer.* Addison Wesley, 1989.

[79] G. Salton, EA Fox, and H. Wu. Extnded Boolean Information Retrieval. *Communication of the ACM*, Vol. 36, No. 11:1022–1036, 1983.

[80] G. Salton and M.J. McGill. *Introduction to Modern Information Retrieval.* McGraw-Hill, Inc. New York, NY, USA, 1986.

[81] G. Salton, A. Wong, and CS Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[82] H. Schutze. Automatic word sense discrimination. *Computational Linguistics*, 24(1):97–123, 1998.

[83] F. Sebastiani. A Tutorial on Automated Text Categorisation. pages 7–35, 1999.

[84] F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys (CSUR)*, 34(1):1–47, 2002.

[85] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, 2002.

[86] H.G. Silber and K.F. McCoy. Efficiently computed lexical chains as an intermediate representation for automatic text summarization. *Computational Linguistics*, 28(4):487–496, 2002.

[87] A. Singhal. Challenges in Running a Commercial Web Search Engine. *IBMs Second Search and Collaboration Seminar*, 2004.

[88] Gilbert Strang. *Introduction To Applied Mathematics.* Cambridge Press, 1986.

[89] Gilbert Strang. *Linear Algebra And Its Applications.* 1986.

[90] Inien Syu, Sheau-Dong Lang, and Narsingh Deo. Incorporating Latent Semantic Indexing into a Neural Network Model for Information Retrieval. In *CIKM*, pages 145–153, 1996.

[91] C. Tang, Z. Xu, and M. Mahalingam. Peersearch: Efficient information retrieval in peer-to-peer networks. *Proceedings of HotNets-I, ACM SIGCOMM*, 2002.

[92] D. Tufis and O. Mason. Tagging Romanian Texts: a Case Study for QTAG, a Language Independent Probabilistic Tagger. *Proceedings of the First International Conference on Language Resources and Evaluation (LREC)*, pages 589–96, 1998.

[93] CJ Van Rijsbergen. Information Retrieval. Butterworth-Heinemann Newton, MA, USA, 1979.

[94] V.N. Vapnik. *Statistical learning theory.* Wiley New York, 1998.

[95] D.H. Widyantoro, T.R. Ioerger, and J. Yen. An adaptive algorithm for learning changes in user interests. *Proceedings of the eighth international conference on Information and knowledge management*, pages 405–412, 1999.

[96] SK Wong, W. Ziarko, VV Raghavan, and PC Wong. Extended Boolean query processing in the generalized vector space model. *Information Systems*, 14(1):47–63, 1989.

[97] SKM Wong and V.V. Raghavan. Vector space model of information retrieval: a reevaluation. *Proceedings of the 7th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 167–185, 1984.

[98] M.R. Wulfekuhler and W.F.P. III. Finding Salient Features for Personal Web Page Categories. *WWW6 / Computer Networks*, 29(8-13):1147–1156, 1997.

[99] R. Yehuda, RA Levengood, J. Schmeidler, S. Wilson, SG Ling, D. Gerber, J. Devooght, OF Smidts, and RE Story. An explanation of the effectiveness of latent semantic indexing by means of a bayesian regression model. *Information Processing and Management*, 32(3):329–344, 1996.

[100] Kai Yu, Shipeng Yu, and Volker Tresp. Multi-label informed latent semantic indexing. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 258–265, New York, NY, USA, 2005. ACM Press.