NAIST-IS-DT0161021

Doctor's Thesis

Studies on Improving Retrieval Accuracy in Web Information Retrieval

KAZUNARI SUGIYAMA

February 6, 2004

Department of Information Systems
Graduate School of Information Science
Nara Institute of Science and Technology

Doctor's Thesis
submitted to Graduate School of Information Science,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
DOCTOR of ENGINEERING

KAZUNARI SUGIYAMA

Thesis committee: Shunsuke Uemura, Professor
Hiroyuki Seki, Professor
Yuji Matsumoto, Professor
Masatoshi Yoshikawa, Professor (Nagoya University)

Studies on Improving Retrieval Accuracy in Web Information Retrieval *

KAZUNARI SUGIYAMA

Abstract

With the rapid spread of the Internet, anyone can easily access a variety of information on the World Wide Web (WWW). Since information resources on the WWW continue to grow, it has become increasingly difficult for users to find relevant information on the WWW. In this situation, Web search engines are one of the most popular methods for finding valuable information effectively. However, users are not satisfied with the retrieval accuracy of search engines. We consider that the following two facts cause this problem:

(i) accurate indexing of Web pages by considering their contents is not performed,

(ii) most search engines cannot provide search results that satisfy each user's information need.

The aim of this thesis is to address the above problems in order to improve retrieval accuracy in Web information retrieval.

In information retrieval systems based on the vector space model, the TF-IDF scheme is widely used to characterize documents. However, in the case of documents with hyperlink structures such as Web pages, it is necessary to develop a technique for representing the contents of Web pages more accurately by exploiting the contents of their hyperlinked neighboring pages. Therefore, in order to address the problem (i), we propose several approaches to refining the TF-IDF scheme for a target Web page by using the contents of its hyperlinked neighboring pages. Experimental results show

*Doctor's Thesis, Department of Information Systems, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-DT0161021, February 6, 2004.

that, generally, more accurate feature vectors of a target Web page can be generated in the case of utilizing the contents of its hyperlinked neighboring pages at levels up to second in the backward direction from the target page.

On the other hand, Web search engines help users find useful information on the WWW. However, when the same query is submitted by different users, typical search engines return the same result regardless of who submitted the query. Generally, each user has different information needs for his/her query. Hence, the search result should be adapted to users with different information needs. Therefore, to address the problem (ii), we propose several approaches to adapting search results according to each user's need for relevant information. Experimental results show that more fine-grained search systems that adapt to user's preferences can be achieved by constructing user profiles based on modified collaborative filtering.

We believe that, in the field of Web information retrieval, our proposed approaches contribute for indexing a target Web page more accurately, and allowing each user to perform more fine-grained search that satisfy his/her information need.

Keywords:

WWW, information retrieval, hyperlink, TF-IDF scheme, information filtering, relevance feedback, user modeling

Acknowledgements

I am most grateful to Professor Shunsuke Uemura for giving me a chance to study Information Retrieval in Database Labolatory. He not only gave me lots of advices derived from his vast range of knowledge but also often encouraged me. I am also grateful to Professor Masatoshi Yoshikawa at Nagoya University, Associate Professor Jun Miyazaki, Assistant Professor Kenji Hatano and Assistant Professor Toshiyuki Amagasa for giving me a lot of comments and advices.

I sincerely thank the members of the thesis committee including Professor Hiroyuki Seki and Professor Yuji Matsumoto for reviewing this thesis. They gave me useful comments which I had not noticed.

I would like to thank Professor Hiroshi Nakagawa (he is currently at The University of Tokyo), Associate Professor Tatsunori Mori and Dr. Koichi Yamada (he is currently at Tokyo Denki University) at Yokohama National University where I had studied in undergraduate and master course days. They gave me informative knowledge about information retrieval, natural language processing, image processing, and machine learning.

I appreciate Professor Shlomo Moran at Technion (Israel Institute of Technology) who gave me valuable comments at my poster presentation in *VLDB2002*. I also appreciate Mr. Chen Yongjun, FUJITSU Corporation, who asked me questions at *DEXA2002* in France. It was quite unexpected that, in France, a person who works for a Japanese company asked me questions. I gratefully appreciate Dr. Krishna Bharat at Google who was my mentor of *SIGIR2003*. Due to a lot of his suggestive and valuable comments, I could refine the quality of my paper although it was not accepted at *SIGIR2003*. Furthermore, at *HyperText'03*, I sincerely appreciate Dr. Kevin McCurley at IBM Almaden Research Center who gave me valuable comments for my research, Professor Andruid Kerne at Texas A&M University who highly praised my research after my presentation, and conference committees who nominated my paper for one of the Ted Nelson Newcomer Award candidates. I would like to acknowledge the contribution of Dr. Ayse Goker at Robert Gordon University in reviewing my paper as my mentor of *SIGIR2004*. She gave me a lot of informative advice that I had not noticed. I would also like to express my sincere appreciation to anonymous reviewers of several conferences. I could evolve my study due to a lot of their helpful comments.

I also thank former and current members in Database Labolatory. Discussion with

them provided me with new and innovative ideas for my research.

I thank my best friends, Keisuke Fujimori at KAJIMA Corporation, Kunihiko Futami at Olympus Corporation, Masaki Kobayashi at Seiko Epson Corporation, and Yoshihisa Shimizu at Nippon Oil Corporation. They encouraged me whenever we met together. Thanks also go to the members of swimming club in my high school days. They always have great zeal in obtaining their own goal. Their such attitudes always inspire me to greater efforts.

Finally, I thank my family for their support and encouragement throughout my doctoral course days at Nara Institute of Science and Technology.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

The World Wide Web (WWW) is a useful resource for users to obtain a great variety of information. However, it is obvious that the number of Web pages continues to grow. Therefore, it is getting more and more difficult for users to find relevant information on the WWW. Under these circumstances, Web search engine is one of the most popular methods for finding valuable information effectively. In the search engines developed in the early stages of the Web, only terms contained in Web pages were utilized as indices. Following these search engines, in order to achieve more higher retrieval accuracy, in recent search engines, the hyperlink structures of Web pages are adopted to index Web pages. For example, HITS (Hypertext Induced Topic Search) [50] and PageRank [66] are well-known algorithms using the hyperlink structures of Web pages. However, the main shortcomings of these algorithms are (1) the weight for a Web page is merely defined; and (2) the relativity of contents among hyperlinked Web pages is not considered. Consequently, the problem of Web pages irrelevant to a user's query often being ranked highly still remains. Therefore, in order to provide users with relevant Web pages, it is necessary to develop a technique for representing the contents of Web pages more accurately. Taking this point into account, we propose several approaches to refining the TF-IDF scheme [78] for a target Web page using the contents of its hyperlinked neighboring pages in order to represent the contents of the target Web page more accurately. We propose the following three approaches:

- the approach relies on the contents of all Web pages at levels up to ${L_{(in)}}^{th}$ in the backward direction and levels up to ${L_{(out)}}^{th}$ in the forward direction from the target page p_{tgt},
- the approach relies on the centroid vectors of clusters generated from Web page groups created at each level up to ${L_{(in)}}^{th}$ in the backward direction and each level up to ${L_{(out)}}^{th}$ in the forward direction from the target page p_{tgt},
- the approach relies on the centroid vectors of clusters generated from Web page groups created at levels up to ${L_{(in)}}^{th}$ in the backward direction and levels up to ${L_{(in)}}^{th}$ in the forward direction from the target page p_{tgt}.

We evaluate retrieval accuracy of the refined feature vector obtained from each approach and show that the third approach is the most effective. These are described in Chapter 4.

The system employing the method in Chapter 4 allow users to get relevant information for their query. However, search results are returned in spite of each user's information need. In other words, when the same query is submitted by different users, most search engines return the same result regardless of who submits the query. In general, each user has different information needs for his/her query. Therefore, we consider that Web search results should adapt to users with different information needs. Several information systems have been proposed in order to personalize information or provide more relevant information for users. For example, (1) systems using relevance feedback [73], (2) systems in which users register their interest or demographic information, and (3) systems that recommend information based on users' ratings. In these systems, users have to register personal information beforehand, or users have to provide feedback on relevant or irrelevant judgements, ratings, and so on. These types of registration, feedback, or ratings can become time consuming and users prefer easier methods. Therefore, we propose several approaches that can be used to adapt search results according to each user's information need without any effort from the user. We propose the following two approaches:

- pure browsing history,
- modified collaborative filtering.

We evaluate retrieval accuracy of each approach and show that the second approach is the most effective. These are described in Chapter 5.

We believe that, in the field of Web information retrieval, our proposed approaches described in this thesis contribute for indexing a target Web page more accurately, and allowing each user to perform more fine-grained search that satisfy his/her information need.

1.2 Organization of this Thesis

This thesis is organized as follows: In Chapter 2, we first describe the framework of information retrieval, and then briefly show the techniques for information retrieval. Then, we review related work on Web search on the basis of its hyperlink structures, and Web search on the basis of user's preferences in Chapter 3. In Chapter 4, in order to represetnt the contents of the Web pages more accurately, we propose several approaches to refining TF-IDF scheme for a target Web page by using the contents of its hyperlinked neighboring pages. In Chapter 5, in order to provide information that satisfies each user's information need, we propose several approaches to adapting search results according to each user's need for relevant information. Finally, we conclude the thesis with a summary and directions for future work in Chapter 6.

Chapter 2

Basic Techniques in Information Retrieval

In this chapter, we first describe the framework of information retrieval, and then briefly show the techniques for information retrieval such as extraction of index terms, retrieval models, and retrieval evaluation. Finally, we describe related techniques of information retrieval.

2.1 Introduction

Information retrieval has been characterized in a variety of ways, ranging from a description of its goals, to relatively abstract models of its components and processes. Generally, the goal of an information retrieval system is for the user to obtain information from the knowledge resource which helps him/her in problem management. Such functions, or goals, of information retrieval have been described in general models of the type shown in Figure 2.1. This model illustrates basic entities and processes in the information retrieval situation.

In this model, a person with some goals and intentions related to, for instance, a work task, finds that these goals cannot be attained because the person's resources or knowledge are somehow inadequate. A characteristic of such a situation is an *anomalous state of knowledge* or information need, which prompts the person to engage in active information-seeking behavior, such as submitting a *query* to an information retrieval system. The query, that must be expressed in a language understood by the

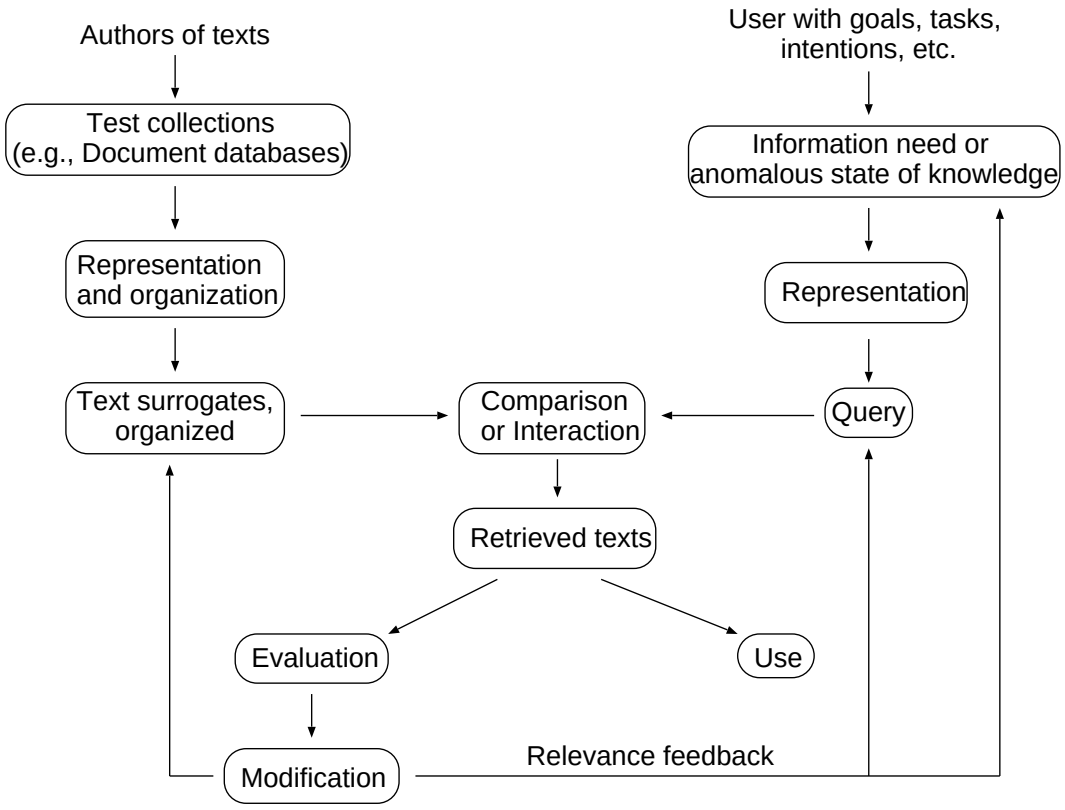


Figure 2.1. A general model of information retrieval.

system, is a representation of the information need. This is shown in the right-hand side of Figure 2.1. Due to the inherent difficulty of representing anomalous state of knowledges, the query in an information retrieval system is always regarded as approximate and imperfect.

On the other side of Figure 2.1, the focus of attention is the information resources that the user of the information retrieval system will eventually access. Here, the model considers the *authors* of texts; the grouping of texts into *collections* (e.g., *databases*); the *representation* of texts; and the *organization* of these representations into databases of *text surrogates*. A typical surrogate would consist of a set of *index terms* or keywords.

The comparison of a query and surrogates, or in some cases, direct interaction between the user and the texts or surrogates (as in hypertext systems), leads to the selection of possibly relevant *retrieved texts*. These retrieved texts are then *evaluated* or used, and either the user will leave the information retrieval system, or the evaluation leads to some *modification* of the query, the information need, or, more rarely, the surrogates. The process of query modification through user evaluation is known as *relevance feedback* [73] in information retrieval. In Section 2.5.1, this process is described in detail.

2.2 Extraction of Index Terms

Not all words are equally significant for representing the semantics of a document. In written language, some words carry more *meaning* than others. Usually, *noun* words or groups of noun words are the ones that are most representative of a document content. Therefore, it is usually considered worthwhile to preprocess the text of the documents in the collection to determine the terms to be used as *index terms*. During this preprocessing phase, other useful text operations can be performed such as elimination of stopwords, stemming (reduction of a word to its grammatical root). Those text operations are described in this section.

2.2.1 Document Preprocessing

Document preprocessing is a procedure which can be divided mainly into four text operations or transformations:

(1) Lexical analysis of the text with the objective of treating digits, hyphens, punctuation marks, and the case of letters.

(2) Elimination of stopwords with the objective of filtering out words with very low discrimination values for retrieval purposes.

(3) Stemming of the remaining words with the objective of removing affixes (i.e., prefixes and suffixes) and allowing the retrieval of documents containing syntactic variations of query terms (e.g., connect, connecting, connected, etc).

(4) Selection of index terms to determine which words/stems or groups of words will be used as an indexing elements. Usually, the decision on whether a particular word will be used as an index term is related to the syntactic nature of the word. In fact, noun words frequently carry more semantics than adjectives, adverbs, and verbs.

In the following, each of these phases is explained in detail.

2.2.2 Lexical Analysis of the Text

Lexical analysis is the process of converting a stream of the text of the documents into a stream of words (the candidate words to be adopted as index terms). Thus, one of the major objectives of the lexical analysis phase is the identification of the words in the text. At first glance, all that seems to be involved is the recognition of spaces as word separators. However, there is more to it than this. For instance, the following four particular cases have to be considered carefully: digits, hyphens, punctuation marks, and the lower and upper case of the letters.

Numbers are usually not good index terms because they are inherently vague without a surrounding context. Thus, it is wise to disregard numbers as index terms. However, we also have to consider that digits might appear mixed within a word. For instance, '510B.C.' is a clearly important index term. In this case, it is not clear what rule should be applied. A preliminary approach to treating digits in the text might be

to remove all words containing sequences of digits unless specified otherwise through regular expressions.

Hyphens pose another difficult decision to the lexical analyzer. Breaking up hyphenated words might be useful due to inconsistency of usage. For instance, this allows treating 'state-of-the-art' and 'state of the art' identically. However, there were words which include hyphens as an indispensable part such as 'gilt-edge,' 'B-49,' and so on. Again, the most suitable procedure seems to adopt a general rule and specify the exceptions on a case by case basis.

Punctuation marks are usually removed entirely in the process of lexical analysis. While some punctuation marks are an integral part of the word (for instance, '510B.C.'), removing them does not seem to have an impact in retrieval performance because the risk of misinterpretation in this case is minimal. In fact, if the user specifies '510B.C.' in his query, removal the dot both in the query terms and in the documents will not affect retrieval. However, very particular scenarios might again require the preparation of a list of exceptions. For instance, if a portion of a program code appears in the text, it might be wise to distinguish between the variables 'x.id' and 'xid.' In this case, the dot mark should not be removed.

The case of letters is not usually important for the identification of index terms. As a result, the lexical analyzer normally converts all the text to either lower or upper case. However, once more, very particular scenarios might require distinction to be made. For instance, when looking for documents which describe details about the command language of a Unix-like operating system, the user might explicitly desire the non-conversion upper cases because this is the convention in the operating system. Moreover, part of the semantics might be lost due to case conversion. For instance, the words *Bank* and *bank* have different meanings. This fact is common to many other pairs of words.

All these text operations can be implemented without difficulty. However, careful thought should be given to each one of them because they might have a profound impact at document retrieval time.

2.2.3 Elimination of Stopwords

Words that are too frequent among the documents in the collection are not good discriminators. In fact, a word which occurs in 80% of the documents in the collection is

useless for purposes of retrieval. Such words are often referred to as *stopwords* and are normally filtered out as potential index terms. In addition, the stopwords are words that does not carry meaning in natural language. Generally, semantics of nouns is easier to identify and to grasp since nouns have meaning by themselves. Therefore, articles, prepositions, and conjunctions are natural candidates for a list of stopwords.

Elimination of stopwords contributes to reducing the size of the indexing structure considerably. In fact, it is common to obtain a compression in the size of the indexing structure of 40% or more solely with the elimination of stopwords.

Since stopword elimination also provides for compression of the indexing structure, the list of stopwords might be extended to include words other than articles, prepositions, and conjunctions. For instance, some verbs, adverbs, and adjectives could be treated as stopwords. SMART system [78], that is a traditional retrieval system for English documents, provides a list of 571 stopwords[1].

One of the problems in elimination of stopwords is to reduce recall. For instance, let us consider a user who is looking for documents containing the phrase 'to be or not to be.' In this case, elimination of stopwords might leave no terms making it impossible to properly recognize the documents which contain the phrase specified. This is one additional reason for the adoption of a full text index by some Web search engines.

2.2.4 Stemming

A user often specifies a query but only a variant of this word is present in a relevant document. Plurals, gerund forms, and past tense suffixes are examples of syntactical variations which prevent a perfect match between a query word and a respective document word. This problem can be partially overcome with the substitution of the words by their respective stems.

A *stem* is the portion of a word which is left after the removal of its affixes (i.e., prefixes and suffixes). A typical example of a stem is the word *connect* which is the stem for the variants *connected*, *connecting*, *connection,* and *connections*. Stems are thought to be useful for improving retrieval performance because they reduce variants of the same root word to a common concept. Furthermore, stemming has the secondary effect of reducing the size of the indexing structure because the number of distinct

[1] ftp://ftp.cs.cornell.edu/pub/smart/english.stop

index terms is reduced.

While the argument supporting stemming seems sensible, there is controversy in the literature about the benefits of stemming for retrieval performance. In fact, different studies lead to rather conflicting conclusions. Frakes [29] compares eight distinct studies on the potential benefits of stemming and concludes that the results of the eight experimental studies he explored do not reach a satisfactory results although he favors the usage of stemming. As a result of these doubts, many Web search engines do not employ any stemming algorithm whatsoever.

In affix removal, the most important part is suffix removal because most variants of a word are generated by the introduction of suffixes. While the Lovins algorithm [58], the Paice/Husk algorithm [67] is well known suffix removal algorithms, the most popular one is that by Porter [68] because its simplicity and elegance. The program of the Porter algorithm in C, Java, and Perl is also provided[2].

2.3 Retrieval Models

In this section, we show the following three classic information retrieval models:

(1) the Boolean model,

(2) the vector space model,

(3) the probabilistic model.

In the following discussion, let t_i be an index term, d be a document, and $w_{t_i}^d$ be a weight associated with t_i in d. This weight quantifies the imprtance of the index term for describing the document semantic contents. In addition, let m be the number of distinct index terms in the system and t_i be a generic index term. $T = \{t_1, t_2, \cdots, t_m\}$ is the set of all index terms. A weight $w_{t_i}^d$ is associated with each index term t_i of a document d. For an index term which does not appear in the document text, $w_{t_i}^d = 0$. With the document, d is associated with an index term vector $\boldsymbol{w}^d$ represented by $\boldsymbol{w}^d = (w_{t_1}^d, w_{t_2}^d, \cdots, w_{t_m}^d)$. Furthermore, let g_i be a function that returns the weight associated with the index term t_i in any m-dimensional vector (i.e., $g_i(t_i) = w_{t_i}^d$). We assume that index term weights are mutually independent. In other words, knowing

[2]http://www.tartarus.org/˜martin/PorterStemmer/

the weight $w_{t_i}^d$ tells us nothing about the weight $w_{t_{i+1}}^d$. This is a simplification because occurrences of index terms in a document are not uncorrelated.

2.3.1 Boolean Model

The Boolean model is a simple retrieval model based on set theory and Boolean algebra. Since the concept of a set is quite intuitive, the Boolean model provides a framework which is easy to grasp by a common user of an information retrieval system. Furthermore, the queries are specified as Boolean expressions that have precise semantics. Given its inherent simplicity and neat formalism, the Boolean model recieved great attention in past years and was adopted by many of the early commercial bibliographic systems.

However, the Boolean model has the following two drawbacks. First, its retrieval strategy is based on a binary decision criterion (i.e., a document is predicted to be either relevant or non-relevant) without any notion of a ranking, which prevents good retrieval performance. Second, while Boolean expresions have precise semantics, it is not often simple to translate an information need into a Boolean expression. In fact, most users find it difficult and awkward to express their query requests in terms of Boolean expressions. The Boolean expressions actually formulated by users are often quite simple. Despite these drawbacks, the Boolean model is the standard model for current large scale, operational information retrieval systems.

The Boolean model considers that index terms are present or absent in a document. As a result, the index term weights are assumed to be all binary, i.e., $w_{t_i}^d \in \{0, 1\}$. A query Q is composed of index terms linked by three connectives: *not*, *and*, *or*. Thus, a query is essentially a conventional Boolean expression that can be represented as a disjunction of conjunctive vectors (i.e., in disjunctive normal form). For instance, the query [$Q = t_a \wedge (\neg t_b \vee t_c)$] can be written in disjunctive normal form as [$\boldsymbol{Q}_{dnf} = (1, 1, 1) \vee (1, 0, 1) \vee (1, 0, 0)$], where each of the components is a binary weighted vector associated with the tuple (t_a, t_b, t_c). These weighted binary vectors are called the conjunctive components of $\boldsymbol{Q}_{dnf}$. Figure 2.2 illustrates the three conjunctive components for the query Q.

In summary, the Boolean model is defined as Figure 2.3. The Boolean model predicts that each document is either *relevent* or *non-relevant*. There is no notion of a *partial match* to the query conditions. For instance, let d be a document where

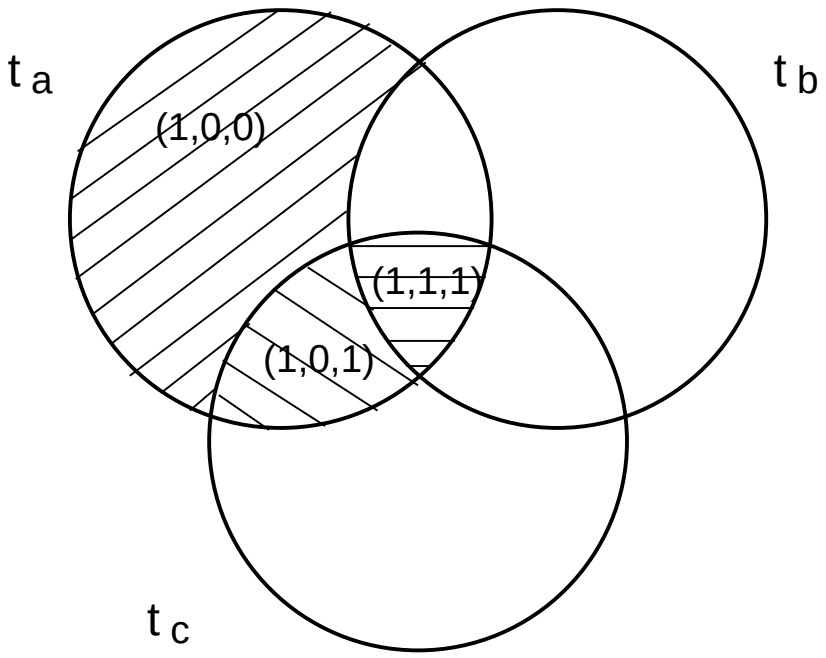


Figure 2.2. The three conjunctive components for the query [$Q = t_a \wedge (\neg t_b \vee t_c)$].

Definition *Boolean model*

For the Boolean model, the index term weight variables are all binary, i.e., $w_{t_i}^d \in \{0, 1\}$. A query Q is a conventional Boolean expression. Let $\boldsymbol{Q}_{dnf}$ be disjunctive normal form for the query Q. In addition, let $\boldsymbol{Q}_{cc}$ be any of the conjunctive components of $\boldsymbol{Q}_{dnf}$. The similarity $sim(d, Q)$ of a document d to the query Q is defined as follows:

$$sim(d, Q) = \begin{cases} 1; & if \exists \boldsymbol{Q}_{cc} | (\boldsymbol{Q}_{cc} \in \boldsymbol{Q}_{dnf}) \vee (\forall t_i, g_i(\boldsymbol{w}^d) = g_i(\boldsymbol{Q}_{cc})) \\ 0; & otherwise \end{cases}$$

If $sim(d, Q) = 1$ then the Boolean model predicts that the document d is relevant to the query Q. Otherwise, the prediction is that the document is not relevant.

Figure 2.3. Definition of the Boolean model.

$\boldsymbol{d} = (0, 1, 0)$. Document d includes the index term t_b but is considered non-relevant to the query [$Q = t_a \wedge (\neg t_b \vee t_c)$].

The main advantages of the Boolean model are the clean formalism behind the model and its simplicity. On the other hand, the main disadvantage is that exact matching may lead to retrieval of too few or too many documents. Nowadays, it is well known that index term weighting can lead to a noticeable improvement in retrieval performance.

2.3.2 Vector Space Model

The vector space model [76] recognizes that the use of binary weights like Boolean model is too limiting and proposes a framework where partial matching is possible. This is accomplished by assigning *non-binary* weights to index terms in queries and in documents. These term weights are utlimately used to compute *similarity* between each document stored in the system and the user query. By sorting the retrieved documents in decreasing order of this similarity, the vector model takes into account documents which match the query terms only partially. The main resultant effect is that the ranked document answer set is a lot more precise in the sense it better matches the user information need than the document answer set retrieved by the Boolean model. The vector space model can be summarized as Figure 2.4. Therefore, a document d and a user query q are represented as m-dimensional vectors as shown in Figure 2.5. The vector model proposes to evaluate similarity of documents d with regard to the query q as the correlation between the vectors $\boldsymbol{w}^d$ and $\boldsymbol{Q}$. This correlation can be quantified, for instance, by the cosine of the angle θ between these two vectors. In other words,

$$
\begin{aligned}
sim(\boldsymbol{w}^d, \boldsymbol{Q}) &= \frac{\boldsymbol{w}^d \cdot \boldsymbol{Q}}{|\boldsymbol{w}^d| \cdot |\boldsymbol{Q}|} \\
&= \frac{\sum_{i=1}^{m} w_{t_i}^d \times q_{t_i}}{\sqrt{\sum_{i=1}^{m} (w_{t_i}^d)^2} \sqrt{\sum_{i=1}^{m} (q_{t_i})^2}},
\end{aligned}
$$

where $|\boldsymbol{w}^d|$ and $|\boldsymbol{Q}|$ are the norms of the document and query vectors. The factor $|\boldsymbol{Q}|$ does not affect the ranking because it is the same for all documents. The factor $|\boldsymbol{w}^d|$ provides a normalization in the space of the documents.

Since $w_{t_i}^d \geq 0$ and $q_{t_i} \geq 0$, $sim(\boldsymbol{w}^d, \boldsymbol{Q})$ varies from 0 to $+1$. Thus, instead of attempting to predict whether a document is relevant or not, the vector model ranks the documents according to their similarity to the query. A document might be retrieved even if it matches the query only *partially*. For instance, one can establish a threshold on $sim(\boldsymbol{w}^d, \boldsymbol{Q})$ and retrieve the documents with similarity above that threshhold. But, in order to compute rankings, we first need to specify how index term weights are obtained. Index term weights can be calculated in many different ways. Salton et al. [78] reviews a variety of term-weighting techniques.

In the vector space model, the raw frequency of a term t_i inside a document d is quantified. Such term frequency is usually referred to as the *TF* factor and provides one measure of how well that term describes the document contents. Furthermore, the inverse of the frequency of a term t_i among the documents in the collection is also quantified. This factor is usually referred to as the *Inverse Document Frequency* or the *IDF* factor. The motivation for usage of an IDF factor is that terms which appear in many documents are not very useful for distinguishing a relevant document from a non-relevant one.

Definition *Vector space model*

For the vector space model, the weight $w_{t_i}^d$ is positive and non-binary. Furthermore, the index terms in the query are also weighted. Let q_{t_i} be the weight of query vector $\boldsymbol{Q}$. Then, the query vector $\boldsymbol{Q}$ is defined as $\boldsymbol{Q} = (q_{t_1}, q_{t_2}, \cdots, q_{t_m})$ where m is the total number of index terms in the system. As before, the vector for a document d is represented by $\boldsymbol{w}^d = (w_{t_1}^d, w_{t_2}^d, \cdots, w_{t_m}^d)$.

Let N be the total number of documents in the system and $df(t_i)$ be the number of documents where the index term t_i appears. Let $tf(t_i, d)$ be the raw frequency of term t_i in the document d (i.e., the number of times the term t_i is mentioned in the text of the document d). Then, the normalized frequency $tf_{nrm}(t_i, d)$ of term t_i in document d is given by,

$$tf_{nrm}(t_i, d) = \frac{tf(t_i, d)}{\sum_{j=1}^{m} tf(t_j, d)}.$$

If the term t_i does not appear in the document d then $tf_{norm}(t_i, d) = 0$. Furthermore, let $idf(t_i)$, inverse document frequency for t_i, be given by,

$$idf(t_i) = \log \frac{N}{df(t_i)}.$$

The best known term-weighting schemes use weights that are given by,

$$\begin{aligned} w_{t_i}^d &= tf_{nrm}(t_i, d) \cdot idf(t_i) \\ &= \frac{tf(t_i, d)}{\sum_{j=1}^{m} tf(t_j, d)} \cdot \log \frac{N}{df(t_i)}, \end{aligned}$$

or variation of this formula. The term weighting strategies described above are called TF-IDF schemes.

Figure 2.4. Definition of the vector space model.

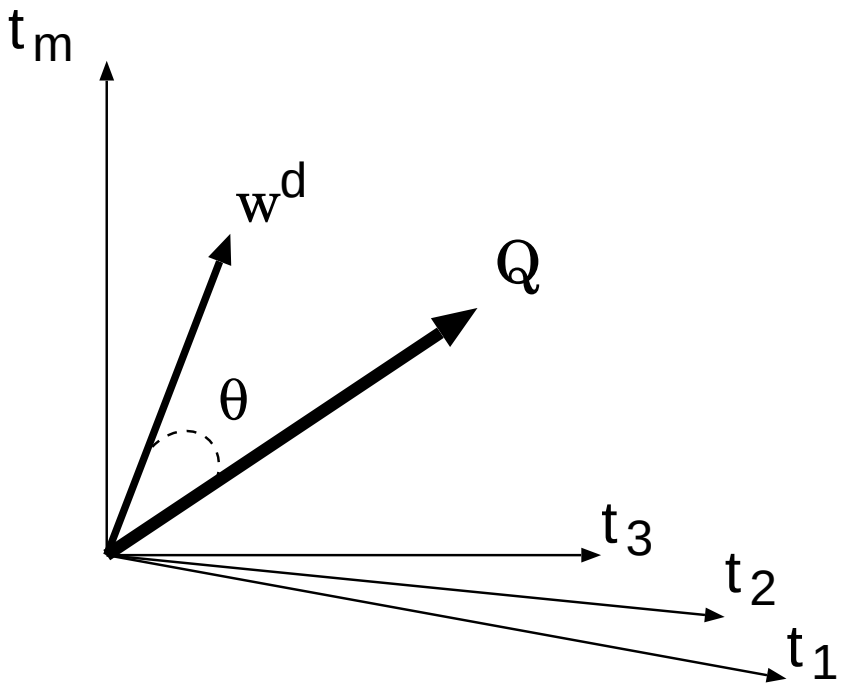


Figure 2.5. Vector Space Model.

Salton et al. [77] examine several variations of the expression described in Figure 2.4 for the weight $w_{t_i}^d$. However, in general, the above expression should provide a good weighting scheme for many collections.

For the query term weights, they suggest,

$$q(t_i) = \left(0.5 + \frac{0.5 \cdot Qf(t_i)}{\sum_{j=1}^{m} Qf(t_j)}\right) \cdot \log \frac{N}{df(t_i)}, \tag{2.1}$$

where $Qf(t_i)$ is the raw frequency of the term t_i in the text of the query q.

The main advantages of the vector space model are: (1) its term-weighting scheme improves retrieval performance; (2) its partial matching strategy allows retrieval of documents that approximate the query conditions; and (3) its cosine ranking formula sorts the documents according to their similarity to the query. Theoretically, the vector model has the disadvantage that index terms are assumed to be mutually independent.

Despite its simplicity, the vector space model is effective ranking strategy with general collections. It yields ranked answer sets which are difficult to improve on without query expansion or relevance feedback (see Section 2.5.1) within the framework of the vecor space model. Although a large variety of alternative ranking methods has been compared with the vector space model, the consensus seems to be that, in general, the vector space model either superior or almost as good as the known alternatives. Furthermore, it is simple and fast. For these reasons, the vector space model is a popular retrieval model nowadays.

2.3.3 Probabilistic Model

Robertson et al. [72] introduced the classic probabilistic model. The model later became known as the Binary Independence Retrieval (BIR) model.

The probabilistic model is based on the following fundamental assumption.

Given a user query q, the probabilistic model assigns to each document d, as a measure of its similarity to the query, the ratio, P (d relevant-to q)/P (d non-relevant-to q) which computes the likelihood of the document d being relevant to the query q. Taking the likelihood of relevance as the rank minimizes the probability of an erroneous judgement [103, 32]. The probabilistic model can be summarized as shown in Figure 2.6

Definition *Probabilistic model*
For the probabilistic model, the index term weight variables are all binary i.e., $w_{t_i}^d \in \{0,1\}$, $w_{q_i} \in \{0,1\}$. A query q is a subset of index terms. Let R be the set of documents known (or initially guessed) to be relavent. Let $\bar{R}$ be the complement of R (i.e., the set of non-relevant documents). Let $P(R|d)$ be the probability that the document d is relevant to the query q and $P(\bar{R}|d)$ be the probability that d is non-relevant to q. The similarity $sim(d,q)$ of the document d to the query q is defined the ratio,

$$sim(d,q) = \frac{P(R|d)}{P(\bar{R}|d)}.$$

Figure 2.6. Definition of the probabilistic model.

Using Bayes' rule, the equation defined in Figure 2.6 can be expressed as follows:

$$sim(d,q) = \frac{P(d|R) \times P(R)}{P(d|\bar{R}) \times P(\bar{R})}.$$

$P(d|R)$ stands for the probability of randomly selecting the document d from the set R of relevant documents. In addition, $P(R)$ stands for the probability that a document randomly selected from the entire collection is relevant. The meanings attached to $P(d|\bar{R})$ and $P(\bar{R})$ are analogous and complementary.

$P(R)$ and $P(\bar{R})$ are the same for all the documents in the collection, in other words, they are constants. Therefore,

$$sim(d,q) \sim \frac{P(d|R)}{P(d|\bar{R})}.$$

Assuming independence of index terms,

$$sim(d,q) \sim \frac{\prod_{g_i(d)=1} P(t_i|R) \times \prod_{g_i(d)=0} P(\bar{t}_i|R)}{\prod_{g_i(d)=1} P(t_i|\bar{R}) \times \prod_{g_i(d)=0} P(\bar{t}_i|\bar{R})}.$$

$P(t_i|R)$ stands for the probability that the index term t_i is present in a document randomly selected from the set R. $P(t_i|\bar{R})$ stands for the probability that the index term

t_i is not present in a document randomly selected from the set R. The probabilities associated with the set $\bar{R}$ have meanings which are analogous to the ones just described. Taking logarithms, recalling that $P(t_i|R) + P(\bar{t}_i|R) = 1$, and ignoring factors which are constant for all documents in the context of the same query, we can finally write,

$$sim(d,q) \sim \sum_{i=1}^{m} q_{t_i} \times w_{t_i}^{d} \times \left(\log \frac{P(t_i|R)}{1 - P(t_i|R)} + \log \frac{1 - P(t_i|\bar{R})}{P(t_i|\bar{R})} \right),$$

which is a key expression for ranking computation in the probabilistic model.

Since we do not know the set R at the beginning, it is necessary to devise a method for initially computing the probabilities $P(t_i|R)$ and $P(t_i|\bar{R})$.

In the very beginning (i.e., immediately after the query specification), there are no retrieved documents. Therefore, one has to make simplifying assumptions such as:
(a) assume that $P(t_i|R)$ is constant for all index terms t_i (typically, eqaul to 0.5), and
(b) assume that the distribution of index terms among the non-relevant documents can be approximated by the distribution of index terms among all documents in the collection. These two assumptions yield,

$$\begin{aligned} P(t_i|R) &= 0.5, \\ P(t_i|\bar{R}) &= \frac{df(t_i)}{N}, \end{aligned}$$

where, as already defined , $df(t_i)$ is the number of documents which contain the index term t_i and N is the total number of documents in the collection. Given this initial guess, we can then retrieve documents which contain query terms and provide an initial probabilistic ranking for them. After that, this initial ranking is improved as follows.

Let V be a subset of a documents initially retrieved and ranked by the probabilistic model. For instance, such a subset can be defined as the top r ranked documents where r is a previously defined threshold. Furthermore, let V_i be the subset of V composed of the documents in V which contain the index term t_i. For simplicity, we also use $|V|$ and $|V_i|$ to denote the number of elements in these sets. For improving the probabilistic ranking, we need to improve our guesses for $P(t_i|R)$ and $P(t_i|\bar{R})$. This can be accomplished with the following assumptions: (a) we can approximate $P(t_i|R)$ by the distribution of the index term t_i among the documents retrieved so far, and
(b) we can approximate $P(t_i|\bar{R})$ by considering that all the non-retrieved documents are non-relevant. With these assumptions, we can write,

$$
\begin{aligned}
P(t_i|R) &= \frac{|V_i|}{|V|}, \\
P(t_i|\bar{R}) &= \frac{df(t_i) - |V_i|}{N - |V|}.
\end{aligned}
$$

This process can then be repeated recursively. By doing so, we are able to improve on our guesses for the probabilities $P(t_i|R)$ and $P(t_i|\bar{R})$ without any assistance from a human subject. However, we can also use assistance from the user for definition of the subset V as originally conceived.

The last formulas for $P(t_i|R)$ and $P(t_i|\bar{R})$ pose problems for small values of V and V_i such as $V = 1$ and $V_i = 0$. To alleviate these problems, an adjustment factor $\frac{df(t_i)}{N}$ is often added as follows:

$$
\begin{aligned}
P(t_i|R) &= \frac{|V_i| + \frac{df(t_i)}{N}}{|V| + 1}, \\
P(t_i|\bar{R}) &= \frac{df(t_i) - |V_i| + \frac{df(t_i)}{N}}{N - |V| + 1}.
\end{aligned}
$$

2.4 Retrieval Evaluation

In an information retrieval system, since the user query request is inherently vague, the retrieved documents are not exact answers. However, they have to be ranked according to the relevance to their query. Such relevance ranking introduces a component that plays a central role in information retrieval. Therefore, information retrieval systems require the evaluation of how precise the answer set is. This type of evaluation is referred to as *retrieval performance evaluation*.

In this section, we discuss retrieval performance evaluation for information retrieval systems. Such an evaluation is usually based on a test reference collection and on an evaluation measure. The test reference collection consists of a collection of documents, a set of example information requests, and a set of relevant documents provided by specialists for each example information request. Given a retrieval strategy S, for each example information request, the evaluation measure quantifies the *similarity* between the set of documents retrieved by S and the set of relevant documents (answer

sets) provided by the specialists. This provides an estimation of the *goodness* of the retrieval strategy S.

In the following discussion, we first explain the two most popular retrieval evaluation measures: recall and precision. We also explain alternative evaluation measures such as the E-measure and the harmonic mean that is often referred to as the F-measure.

2.4.1 Recall and Precision

Let us consider an example information request I and its set R of relevant documents. Let $|R|$ be the number of documents in this set. Furthermore, let $|R_a|$ be the number of documents in the intersection of the sets R and A. Figure 2.7 shows these sets.

The recall and precision measures are defined as follows:

Recall

This measure is the fraction of the relevant documents (the set D_R in Figure 2.7) which have been retrieved, i.e.,

$$Recall = \frac{|D_{R_a}|}{|D_R|}. \tag{2.2}$$

Precision

This measure is the fraction of the retrieved documents (the set D_A in Figure 2.7) which is relevant, i.e.,

$$Precision = \frac{|D_{R_a}|}{|D_A|}. \tag{2.3}$$

Recall and precision, as defined by Equation (2.2) and (2.3), respectively, assume that all documents in the answer set D_A at once. Instead, the documents in D_A are first sorted according to a degree of relevance, in other words, a ranking is generated. The user then examines this ranked list starting from the top document. In this situation, the recall and precision measures vary as the user proceeds with his/her examination of the answer set D_A. Thus, proper evaluation requires plotting a precision versus recall curve as follows.

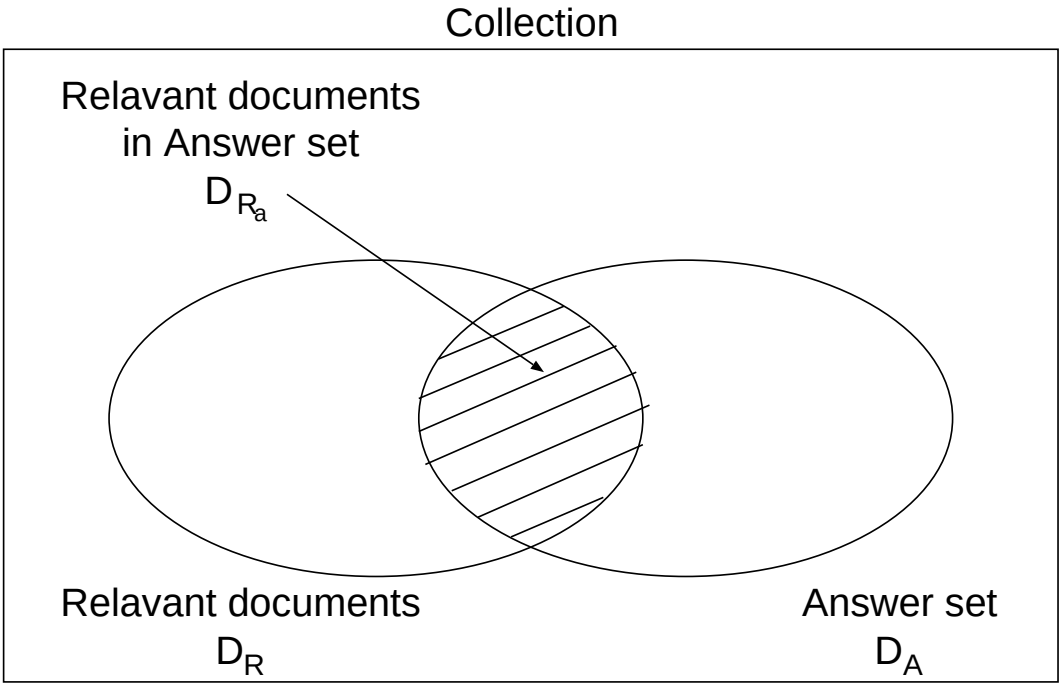


Figure 2.7. Recall and precision for a given example information request.

As before, consider a reference collection and its set of example information requests. Let us focus on a given example information request where a query q is formulated. Assume that a set R_q containing the relevant documents for q has been defined. Without loss of generality, assume further that the set R_q is composed of the following documents,

$$R_q = \{d_2, d_6, d_8, d_{17}, d_{21}, d_{35}, d_{47}, d_{52}, d_{78}, d_{83}\}.$$

Thus, according to a group of specialists, there are ten documents which are relevant to the query q.

Now, let us consider a new algoritm which has just been designed. For the query q, assume that this algorithm returns a ranking of the documents in the answer set as the following.

Ranking for query q:

1. $d_{52}\star$	6. d_{12}	11. d_3	16. $d_{35}\star$	21. $d_{47}\star$	26. d_{28}
2. d_{38}	7. $d_2\star$	12. d_{11}	17. d_{15}	22. d_{37}	27. d_{97}
3. d_{25}	8. d_{48}	13. $d_8\star$	18. d_{107}	23. d_{36}	28. d_{82}
4. $d_{21}\star$	9. d_{66}	14. d_{58}	19. $d_{83}\star$	24. d_{61}	29. $d_{17}\star$
5. d_5	10. $d_{78}\star$	15. d_{72}	20. d_{23}	25. $d_6\star$	30. d_{45}

The documents that are relevant to the query q are marked with a star after the document number. If we examine this ranking, starting from the top document, we can observe the following points. First, the document d_{52} which is ranked as number 1 is relevant. Furthermore, this document corresponds to 10% of all the relevant documents in the set R_q. Thus, we can say that we have a precision of 100% at 10% recall. Second, the document d_{21} which is ranked as number 4 is the next relevant document. At this point, we can say that we have a precision of 50% (two documents out of four are relevant) at 20% recall (two of ten relevant documents have been seen). Third, if we proceed with our examination of the generated ranking, we can plot a curve of precision versus recall as shown in Figure 2.8. This precision versus recall curve is usually based on 11 *standard* recall levels which are 0%, 10%, 20%, $\cdots$,100%.

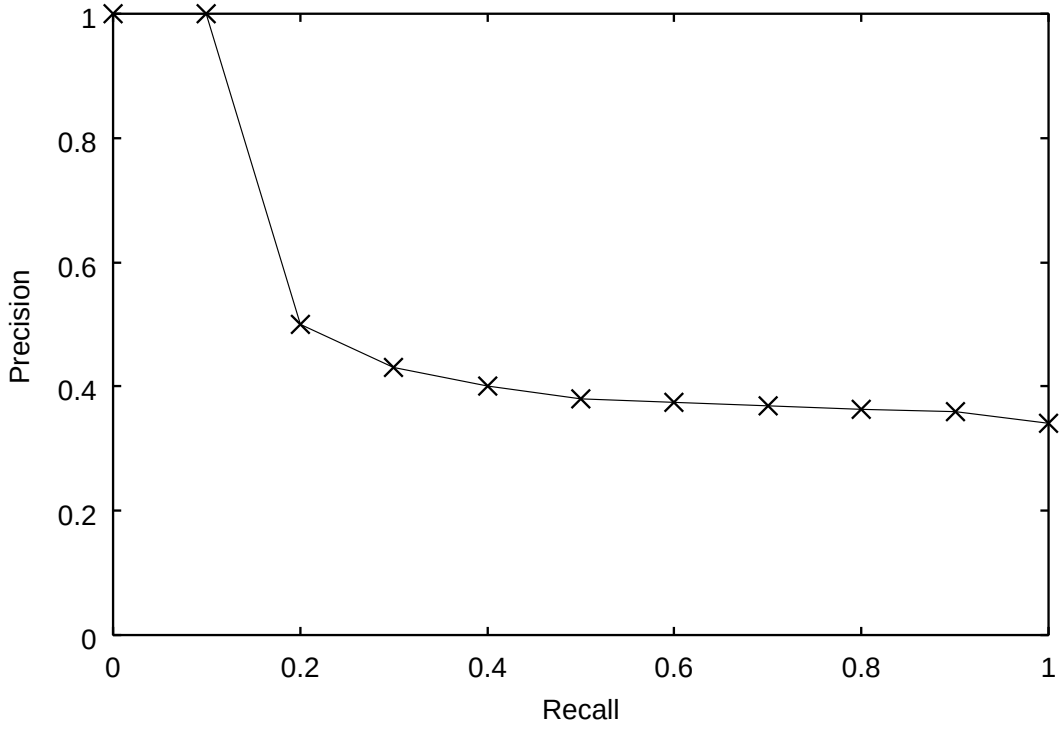


Figure 2.8. Precision at 11 standard recall levels.

In the above example, the precision and recall figures are for a single query. Usually, however, retrieval algorithms are evaluated by running them for several distinct queries. In this case, a distinct precision versus recall curve is generated for each query. To eveluate the retrieval performance of an algorothm over all test queries, we average the precision figures at each recall level as follows:

$$\bar{P}(r) = \sum_{i=1}^{N_q} \frac{P_i(r)}{N_q},$$

where $\bar{P}(r)$ is the average precision at the recall leval r, N_q is the number of used queries, and $P_i(r)$ is the precision at recall level r for the i^{th} query.

The curve of precision versus recall which results from averaging the results for various queries is usually referred to as precision versus recall figures. Such average figures are normally used to compare the retrieval performance of distinct retrieval algorithms. For example, one could compare retrieval performance of a newly proposed retrieval algorithm with the retrieval performance of the classic vector space model. Figure 2.9 illustrates average precision versus recall figures for two distinct retrieval algorithms, (a) and (b). In this case, the algorithm (a) has higher precision at lower recall levels while the algorithm (b) is superior at higher recall levels.

Average precision versus recall figures are now standard evaluation strategy for information retrieval systems and are used widely in the information retrieval literature. They are useful because they allow us to evaluate quantatively both the quality of the overall answer set and the breadth of the retrieval algorithm.

Average precision versus recall figures are useful for comparing the retrieval performance of distinct algorithms over a set of example queries. However, there are situations in which we would like to compare the retrieval performance of our retrieval algorithms for the individual queries. In this case, a single precision value for each query can be used. This single value should be interpreted as a summary of the corresponding precision versus recall curve. Usually, this single value summary is taken as the precision at a specified recall level. For instance, we could evaluate the precision when we observe the first relevant document and take this precision as the single value summary.

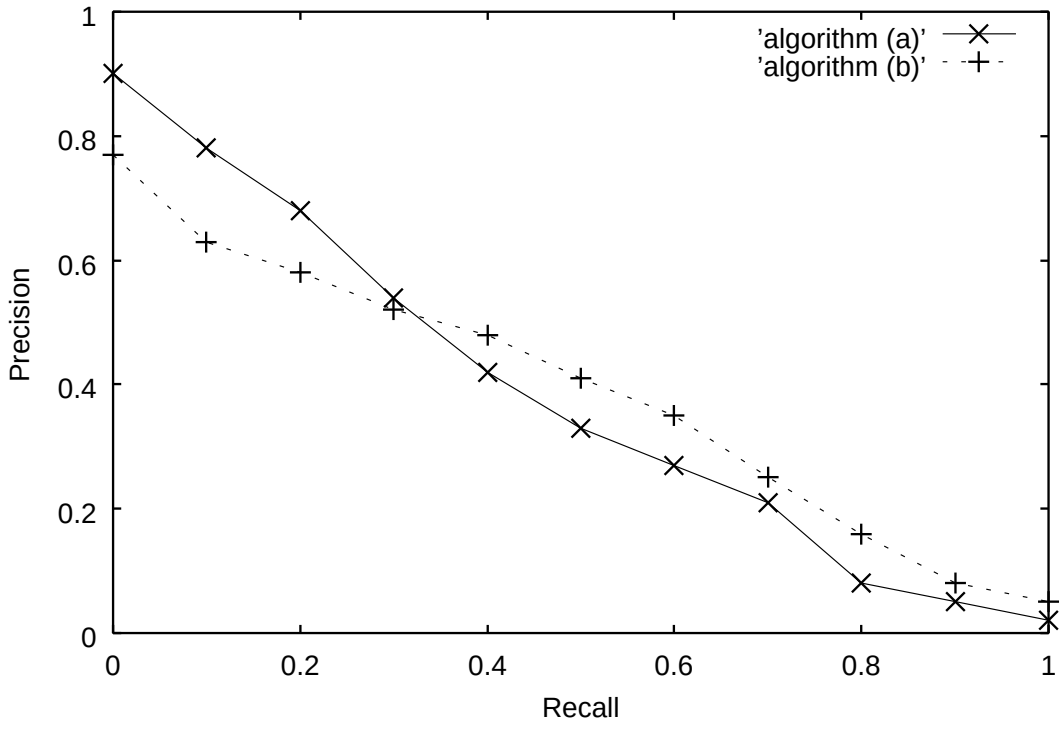


Figure 2.9. Recall and precision for a given example information request.

Average Precision at Seen Relevant Documents

The idea here is to generate a single value summary of the ranking by averaging the precision figures obtained after each new relevant document is observed in the ranking. For instance, consider the example in Figure 2.8. The precision figures after each new relevant document is observed are 1.0, 0.5, 0.43, 0.4, 0.38, 0.375, 0.368, 0.364, 0.36, and 0.34. Thus, the average precision at seen relevant documents is given by (1.0 + 0.5 + $\cdots$ + 0.34)/10 or 0.45. This measure favors systems which retrieve relevant document quickly (i.e., early in the ranking).

R-Precision

The idea here is to generate a single value summary of the ranking by computing the precision at the R^{th} position in the ranking, where R is the total number of relevant documents for the current query (i.e., the number of documents in the set R_q). For instance, consider the examples in Figure 2.8. The value of R-precision is 0.4 for this example because $R = 10$ and there are four relevant documents among the first ten documents in the ranking. The R-precision measure is a useful parameter for observing the behavior of an algorithm for each individual query in an experiment. Additionally, one can also compute an average R-precision figure over all queries.

Precision Histograms

The R-precision measures for several queries can be used to compare the retrieval history of two algorithms as follows. Let $RP_A(i)$ and $RP_B(i)$ be the R-precision values of the retrieval algorithms A and B for the i^{th} query. For instance, define the difference,

$$RP_{A/B}(i) = RP_A(i) - RP_B(i).$$

A value of $RP_{A/B}(i)$ equal to 0 indicates that both algorithms have equivalent performance for the i^{th} query. A positive value of $RP_{A/B}(i)$ indicates a better retrieval performance by algorithm A for the i^{th} query while a negative value indicates a better retrieval performance by algorithm B. Figure 2.10 illustrates the $RP_{A/B}(i)$ values for two hypothetical retrieval algorithms over ten example queries. The algorithm A

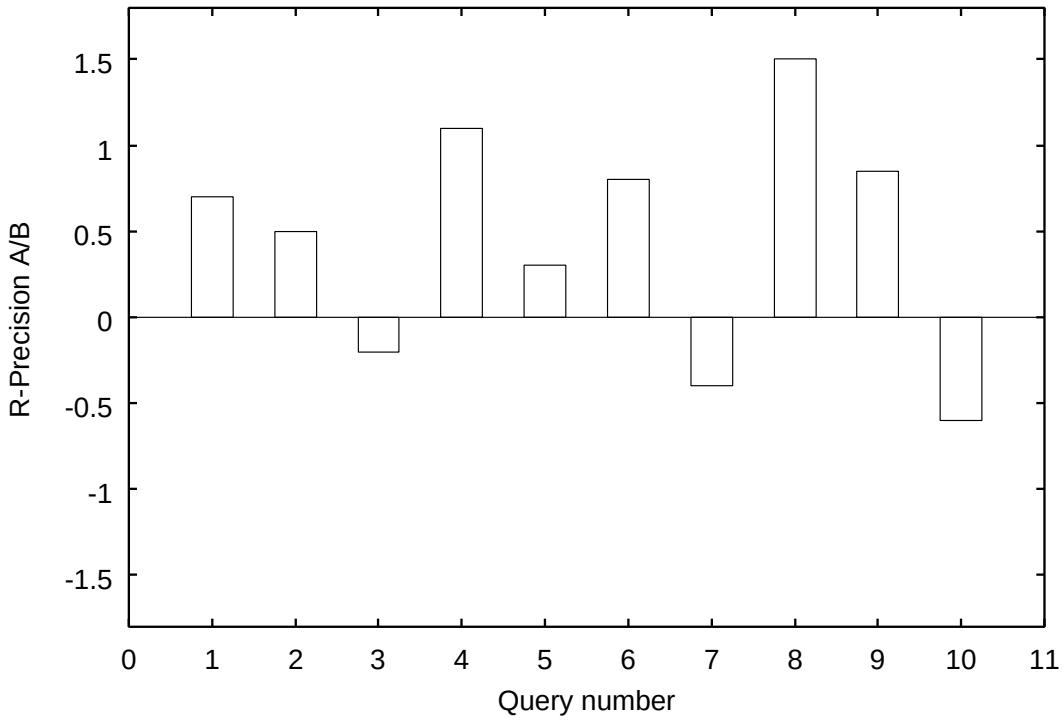


Figure 2.10. A precision histogram.

is superior for seven queries while the algorithm B performs better for the three other queries (numbered 3, 7, and 10). This type of bar graph is called a *precision histogram*.

2.4.2 Alternative Measures

Despite their popularity, since recall and precision are not always the most appropriate measures for evaluating retrieval performance, alternative measures have been proposed over the years. In the following, we briefly review some of them.

The Harmonic Mean (F-Measure)

As discussed above, a single measure which combines recall and precision might be of interest. One such measure is the harmonic mean F of recall and precision [48] which is computed as,

$$\begin{aligned} F(j) &= \frac{2}{\frac{1}{r(j)} + \frac{1}{P(j)}} \\ &= \frac{2r(j)P(j)}{r(j) + P(j)}, \end{aligned}$$

where $r(j)$ is the recall for the j^{th} document in the ranking, $P(j)$ is the precision for the j^{th} document in the ranking, and $F(j)$ is the harmonic mean of $r(j)$ and $P(j)$ (thus, relative to the j^{th} document in the ranking). The function F assumes values in the interval [0,1]. It is 0 when no relevant documents have been retrieved and is 1 when all ranked documents are relevant. Furthermore, the harmonic mean F assumes a high value only when both recall and precision are high. Therefore, determination of the maximium value for F can be interpreted as an attempt to find the best possible compromise between recall and precision. This measure is often referred to as F-measure.

The E-Measure

Another measure which combines recall and precision was proposed by van Rijsbergen [103] and is called the E evaluation measure. The idea is to allow the user to specify whether he/she is more interested in recall or in precision. The E-measure is defined

as follows:

$$E(j) = 1 - \frac{1 + b^2}{\frac{b^2}{r(j)} + \frac{1}{P(j)}},$$

where $r(j)$ is the recall for the j^{th} document in the ranking, $P(j)$ is the precision for the j^{th} document in the ranking, $E(j)$ is the E evaluation measure relative to $r(j)$ and $P(j)$, and b is a user specified parameter which reflects the relative importance of recall and precision. For $b = 1$, the $E(j)$ measure works as the complement of the harmonic mean $F(j)$. Values of b greater than 1 indicate that the user is more interested in precision than in recall while values of b smaller than 1 indicate that the user is more interested in recall than precision.

2.5 Related Techniques of Information Retrieval

2.5.1 Relevance Feedback

Relevance feedback [73] is the most popular query reformulation strategy. In a relevance feedback cycle, the user is presented with a list of the retrieved documents and marks those that are relevant after examining them. In practice, only the top 10 or 20 ranked documents need to be examined. The main idea consists of selecting important terms, or expressions, attached to the documents that have been identified as relevant by the user, and of enhancing the importance of these terms in a new query formulation. The expected effect is that the new query will be moved towards the relevant documents and away from the non-relevant ones.

Early experiments using the SMART system [76] and later experiments using the probabilistic weighting model [72] have shown good improvements in precision for small test collections when relevance feedback is employed. Such improvements come from the use of two basic techniques: query expansion (addition of new terms from relevant documents) and term reweighting (modification of term weights based on the user relevance judgement).

Relevance feedback presents the following main advantages over other query reformulation strategies: (a) it shields the user from the details of the query reformulation process because all the user has to provide is a relevance judgement on documents;
(b) it breaks down the whole searching task into a sequence of small steps which are

easier to grasp; and (c) it provides a controlled process designed to emphasize some terms (relevant ones) and de-emphasize others (non-relevant ones).

In the following, we discuss the usage of user relevance feedback to expand queries with the vector space model.

Query Expansion and Term Reweighting for the Vector Space Model

The application of *relevance feedback* to the vector space model considers that the term-weight vectors of the documents identified as relevant to a given query have similarities among themselves, i.e., relevant documents resemble each other. Further, it is assumed that non-relevant documents have term-weight vectors which are dissimilar from the ones for the relevant documents. The basic idea is to reformulate the query such that it gets closer to the term-weight vector space of the relevant documents.

Let us define some additional terminology regarding the processing of a given query q as follows:

D_r: set of relevant documents, as identified by the user, among the retrieved documents;
D_n: set of non-relevant documents among the retrieved documents;
C_r: set of relevant documents among all documents in the collection;
$|D_r|, |D_n|, |C_r|$: number of documents in the sets D_r, D_n, and C_r, respectively;
α, β, γ: tuning constants.

Consider first the unrealistic situation where the complete set C_r of relevant documents to a given query q is known in advance. In such a situation, it can be demonstrated that the best query vector for distinguishing the relevant documents from the non-relevant documents is given by the following equation:

$$\boldsymbol{Q}^{opt} = \frac{1}{|C_r|} \sum_{\forall \boldsymbol{d} \in C_r} \boldsymbol{d}_j - \frac{1}{N - |C_r|} \sum_{\forall \boldsymbol{d} \notin C_r} \boldsymbol{d}_j \tag{2.4}$$

The problem with this formulation is that the *relevant* documents which compose the set C_r are not known a priori. The natural way to avoid this problem is to formulate an initial query and to incrementally change the initial query vector. This incremental change is accomplished by restricting the computation to the documents *known* to be

relevant (according to the user judgement) at that point. There are three classic and similar ways to calculate the modified query $\boldsymbol{Q}^{mod}$ as follows:

$$
\begin{aligned}
Standard\_Rocchio: \quad & \boldsymbol{Q}^{mod} = \alpha \boldsymbol{Q}^{org} + \frac{\beta}{|D_r|} \sum_{\boldsymbol{d}_j \in D_r} \boldsymbol{d}_j - \frac{\gamma}{|D_n|} \sum_{\boldsymbol{d}_j \in D_n} \boldsymbol{d}_j, \\
Ide\_Regular: \quad & \boldsymbol{Q}^{mod} = \alpha \boldsymbol{Q}^{org} + \beta \sum_{\boldsymbol{d}_j \in D_r} \boldsymbol{d}_j - \gamma \sum_{\boldsymbol{d}_j \in D_n} \boldsymbol{d}_j, \\
Ide\_Dec\_Hi: \quad & \boldsymbol{Q}^{mod} = \alpha \boldsymbol{Q}^{org} + \beta \sum_{\boldsymbol{d}_j \in D_r} \boldsymbol{d}_j - \gamma \ max_{non-relevant}(\boldsymbol{d}_j)
\end{aligned}
$$

where $max_{non-relevant}(\boldsymbol{d}_j)$ is a reference to the highest ranked non-relevant document. Notice that now D_r and D_n stand for the sets of relevant and non-relevant documents (among the retrieved ones) according to the user judgement, respectively. In the original formulations, Rocchio [73] fixed $\alpha = 1$ and Ide [44] fixed $\alpha = \beta = \gamma = 1$. The expressions above are modern variants. The current understanding is that the three techniques yield similar results (in the past, Ide Dec-Hi was considered slightly better).

The Rocchio formulation is basically a direct adaptation of Equation (2.4) where the original query are added. The motivation is that the original query q may practically contain important information. Usually, the information contained in relevant documents is more important than the information provided by non-relevant documents [78]. This suggests making the constant γ smaller than the constnat β. An alternative approach is to set γ to 0 which yields a *positive* feedback strategy.

The main advantages of the above relevance feedback techniques are simplicity and good results. The simplicity is due to the fact that the modified term weights are computed directly from the set of retrieved documents. The good results are observed experimentally and are due to the fact that the modified query vector does reflect a portion of the intended query semantics. The main disadvantage is that *no* optimality criterion is adopted.

2.5.2 Information Extraction

While information retrieval systems can collect useful material such as texts from vast fields of raw material, an information extraction system starts with a collection of such texts, then transforms them into information that is more readily digested and analyzed (see Figure 2.11). It isolates relevant text fragments, extracts relevant information from the fragments, and then pieces together the targeted information in a coherent framework. For example, an article may discuss chemical gases, temperatures, and material specifications, but only one or two of these items may be of interest to the human analyst. The aim of information extraction research is to build systems that find and link relevant information while ignoring extraneous and irrelevant information. Figure 2.12 shows the basic components of a typical information extraction system. Most information extraction systems depend on off-line components to produce the data and rules. Two aspects of preprocessing are common to most of information extraction systems:

- Part-of-speech tagging programs to allow preliminary recognition of phrasal units in sentences,
- Special-purpose rules to recognize the semantic classes of phrasal units, including company names, places, people's names, and equipment names.

Information extraction has numerous potential applications. For example, information available in unstructured text can be translated into traditional databases user can probe through standard queries. Suppose we want to track the profits of Japan car manufacturers and compare them with those of European car manufacturers. Relevant information includes company name, company nationality, the fact that the company is in the car industry, and the amount the and currency of its profits. An information extraction system that tracks news release in this area, updating a database as the information becomes available, can help detect trends as soon as public announcements are made. Other information extraction systems might process news events, including meetings of important people, formation of new companies, and announcements of new products. Meanwhile, information retrieval systems could benefit from the extracted information to construct sensitive indices more closely linked to the actual meaning of a particular text.

Early information extraction system development is exemplified by the following systems:

- One of the earliest information extraction systems, which operated on texts of unrestricted topics, was implemented by DeJong [25, 26]. Using a newswire network as its data source, DeJong's program, called FRUMP sought to match each new story with a relevant script on the basis of keywords and conceptual sentence analysis. FRUMP was a semantically oriented system that used domain-specific expectations to instantiate event descriptions based on scriptal knowledge.

- An even earlier project – before 1970 – for extracting useful information from text was directed by Sager of the Linguistic String Project group at New York University [75]. Sponsored by the American Medical Association, this work sought to convert patient discharge summaries to form suitable for use as input to a traditional Conference on Data Systems Languages (CODASYL) database management system.

- In 1980, Silva et al. [88] extracted satellite-flight information from reports produced by monitors around the world, but the system was restricted to single sentences and lacked a methodology for extracting complete event descriptions.

- Zarri [106] started working on information extraction systems in the early 1980s. The texts he used described the activities of various French historical figures. The system sought to extract information on the relationships and meetings between these people.

- In 1981, Cowie [22] developed an information extraction system that extracted canonical structures from field-guide descriptions of plants and animals. The system used simple information to populate a fixed-record structure.

In addition, the following early commercial information extraction systems were also developed. This signs viability of information extraction systems.

- ATRANS was the earliest information extraction system to be deployed as a commercial product [59]. That was designed to handle money-transfer telexes in international banking. It used sentence analysis techniques similar to those

in FRUMP described above and exploited the fact that the content of money-transfer telexes is highly predictable. ATRANS demonstrated that relatively simple language processing techniques are adequate for information extraction applications narrow in scope and utility.

- JASPER is an information extraction system that extracts information from reports on corporate earnings [1], achieving robust natural language processing capabilities through template-driven techniques for language analysis operating on small sentence fragments. JASPER development depends on a manual inspection of the system in action, along with ongoing system evaluations using representative test sets.

- SCISOR is a prototype incorporating information extraction into an integrated system [70] that uses partial analysis of the text to carry out its processing. A filtering process selects only articles on corporate mergers and acquisitions, extracting information on target and suitor companies and dollar-per-share amounts. These items are stored in a knowledge base that handles queries.

Recently, in the field of life science, knowledge described in literature is actively used with regard to protein-protein interactions. However, in the case of utilizing such knowledge, the main problems are the following: (1) findings described in the form of language can be understood only when researchers read each paper one by one; (2) even if findings in a paper relate to findings in other papers, their correlation can be identified only by researchers who read these papers; (3) if the number of related findings reaches the tens of thousands, the work of indentifying correlative findings sprinkled in papers and organizing them as a knowledge system becomes enormous Therefore, in order to alleviate these problems, we can refer to several approaches that extract information on protein-protein interactions from biological literature. For example, the approaches based on using (1) surface expressions of the sentences within biological literature [9], (2) a full parser [85, 30, 105], and (3) templates [42, 100]. However, in the surface expression approach, it is extremely difficult to define manually comprehensive rules of extracting information on protein-protein interactions. In the full parser approach, it has often been pointed out that the processing speed is slow and that the results have ambiguity. In addition, the template approach can suffer

from the same problems that arise in the surface expression and full parser approaches. Therefore, it is necessary to automatically acquire the rules of extracting information on protein-protein interactions without defining the rules manually and without using a full parser. Moreover, biological literature commonly contains (1) many unknown words such as compound names, as well as original proper nouns coined by the paper's authors; (2) non-alphabetical characters such as "()", "-" and so on; and (3) a variety of verbal forms such as indicative, passive, gerundive, and so on. Furthermore, Ding et al. [27] found that it is effective to analyze a sentence as a unit for extracting information on the interactions of two objects. On the basis of these points, in order to identify sentences describing information on protein-protein interactions, we proposed an approach of automatically constructing the classifier by several machine learning approaches trained by the features that express the characteristics of biological literature [94]. It is expected that these researches would highly useful to support biologists' research.

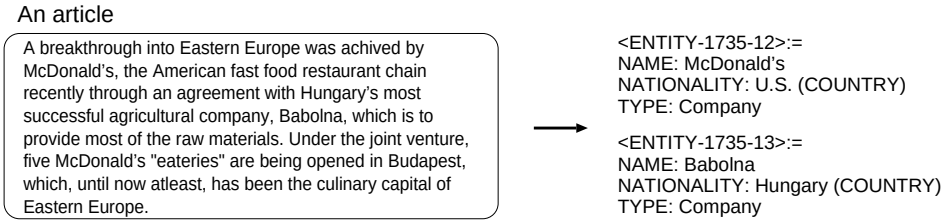


Figure 2.11. Information Extraction transforming a text's structures.

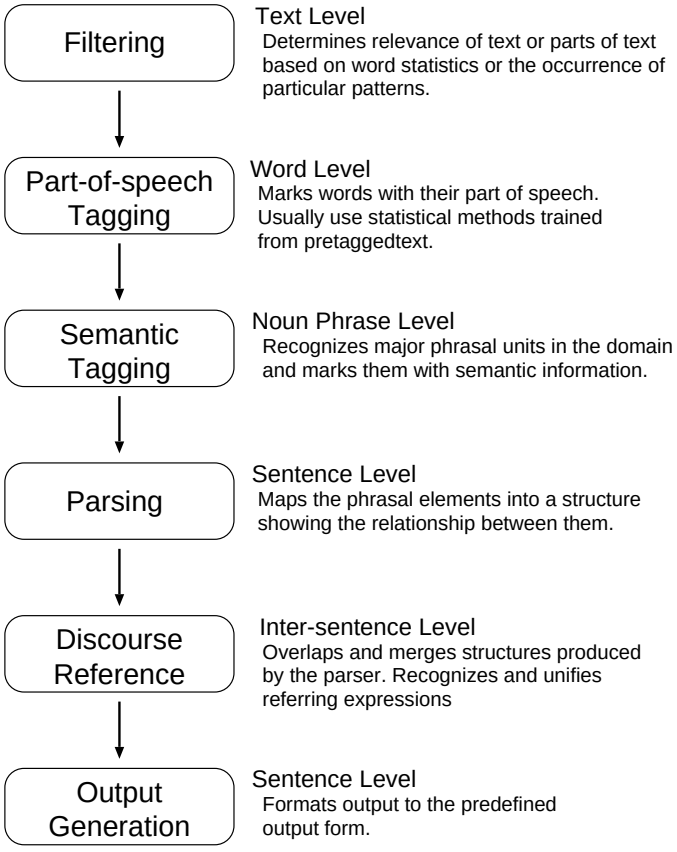


Figure 2.12. Typical information extraction system components.

2.5.3 Information Filtering

Information filtering is a name used to describe a variety of processes involving the delivery of information to people who need it. In a filtering task, a *user profile* describing the user's preferences is constructed. Such a profile is then compared with the incoming documents in an attempt to determine those which might be of interest to this particular user. For instance, this approach can be used to select a news article among thousands of articles which are broadcast each day. Other potential scenarios for the application of filtering include the selection of preferred judicial decisions, or the selection of articles from daily newspapers, etc. Information filtering has the following characteristics or features:

- An information filtering system is an information system designed for unstructured or semistructured data. This contracts with a typical database application that involves very structured data, such as employee records. E-mail messages are an example of semistructured data in that they have well-defined header fields and unstructured text body.

- Information filtering systems deal primarily with textual information. In fact, unstructured data is often used as a synonym for textual data. It is, however, more general than that, should include other types of data such as images, voice, and video that are part of multimedia information systems.

- Information filtering systems involve large amount of data such as gigabytes of text, or much larger amounts of other media.

- Filtering applications typically involve large amount of data, either being broadcast by remote sources such as newswire services, or sent directly by other sources like email.

- Filtering is based on descriptions of individual or group information preferences, often called profiles. Such profiles typically represent long-term interests.

- Filtering is often meant to imply the removal of data from an incoming stream, rather than finding data in that stream. In the former case, the users of the system see what is left after the data is removed; in the latter case, they see the data that is extracted.

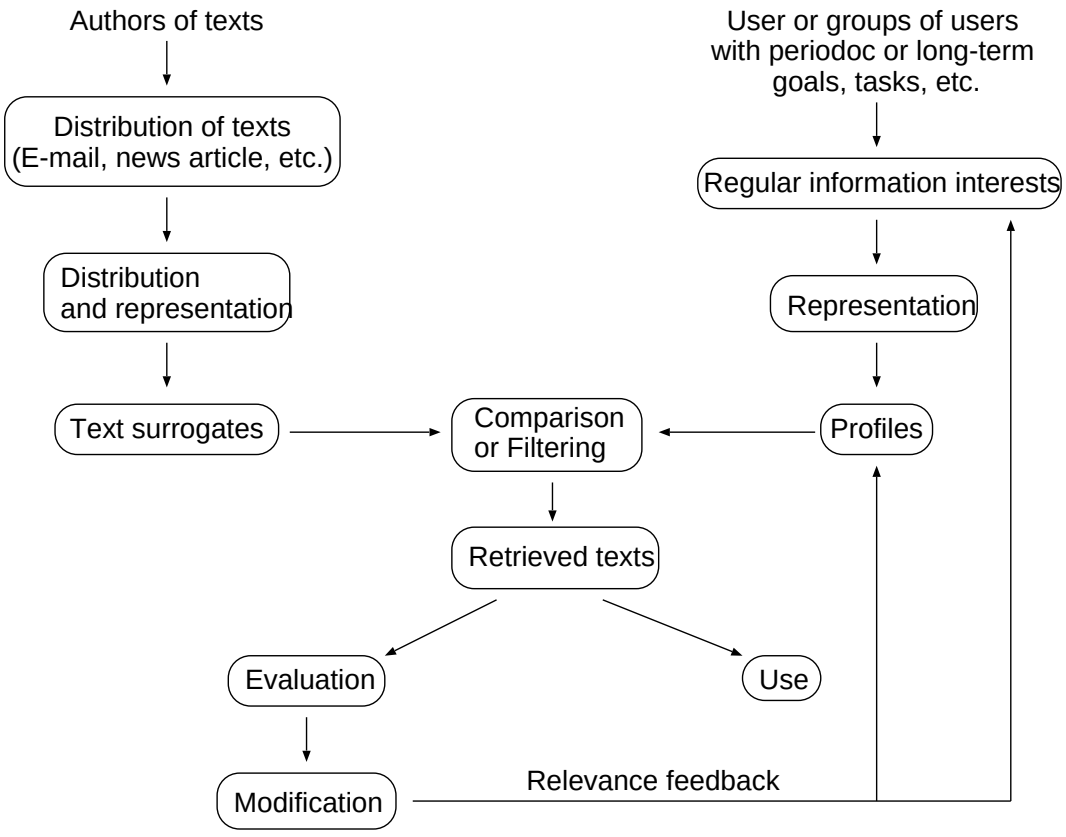


Figure 2.13. A general model of information filtering.

Based on the general model of information retrieval in Figure 2.1, and the description of information filtering features described above, a model of information filtering that appears to describe the major entities and process involved is illustrated in Figure 2.13. In this model, information filtering begins with people (the *users of the filtering system*) who have relatively stable, long-term, or periodic *goals* or *desires* (e.g., accomplishing a work task, or being entertained). Groups, as well as individuals, can be characterized by such goals. These then lead to *regular information interests* (e.g., keeping up-to-date on a topic) that may change slowly over time as conditions, goals, and knowledge change. Such information interests lead the people to engage in relatively passive forms of information-seeking behavior, such as having texts brought to their attention. This is accomplished by *representation* of the information interests as *profiles* or *queries* that can be put to the filtering system. Such profiles have generally been constructed as good specifications of the information interests.

On the left side of Figure 2.13, the focus is on *authors* of *texts*, who are often institutions, such as newspapers, as well as individuals. These institutions, or others, such as newsgroups, undertake to *distribute* the texts as they are generated, so they can be brought to user's attention. To accomplish this, the texts are *represented* and *compared* to the profiles. The comparison results in some of the texts being brought to the user's attention (being retrieved). These texts are *used* (or not) and are *evaluated* in terms of how well they respond to the information interests and their motivating goals. The evaluation may lead to *modification* of the profiles and information interests. The modified entities are used in subsequent comparison process.

In comparing and discussing Figures 2.1 and 2.13, we note that, at this rather abstract level, the entities and processes relevant to information filtering are almost identical to those that are relevant to information retrieval. The major differences appear to be:

- Where information retrieval is typically concerned with single uses of the system, by a person with a one-time goal and one-time query, information filtering is concerned with repeated uses of the system, by a person or persons with long-term goals or interests.

- Where information retrieval recognizes inherent problems in the adequacy of queries as representations of information needs, information filtering assumes that profiles can be correct specifications of information interests.

- Where information retrieval is concerned with the collection and organization of texts, information filtering is concerned with the distribution of texts to groups or individuals.
- Where information retrieval is typically concerned with the selection of texts from a relatively static databases, information filtering is mainly concerned with selection or elimination of texts from a dynamic datastream.
- Where information retrieval is concerned with responding to the user's interaction with texts within a single information-seeking episode, information filtering is concerned with long-term changes over a series of information-seeking episodes.

Chapter 3

Related Work

3.1 Related Work on Web Search Based on its Hyperlink Structures

Hyperlink structures are one of the features of Web pages. Users can navigate the huge Web space easily through this hyperlink structures; therefore, many researches on Web information retrieval have been focusing on the Web's hyperlink structures. In this section, we review related work of information retrieval systems using the Web's hyperlink structures, especially the systems based on the concept of "optimal document granularity" and the two most popular Web page weighting algorithms, *HITS* [50] and *PageRank* [66].

3.1.1 Information Retrieval Systems based on the Concept of "Optimal Document Granularity"

With respect to this research area, we refer to the following works. Tajima et al. [97] presented a technique which uses "*cuts*" (results of Web structure analysis) as retrieval units for the Web. Moreover, they extended to rank search results involved multiple keywords by (1) finding minimal subgraphs of links and Web pages including all keywords; and (2) computing the score of each subgraph based on locality of the keywords within it [96]. Following these works, Li et al. [55] introduced the concept of an "*information unit*," which can be regarded as a logical document consisiting of

multiple physical Web pages as one atomic retrieval unit, and proposed a novel framework for document retrieval by information units. However, these approaches require considerable processing time to analyze hyperlink structures and to discover the semantics of Web pages. In addition, while these systems can find retrieval units exactly, it often arises that they find retrieval units irrelevant to the user specified query terms. As for these works, we do not believe that users could understand the search results intuitively, because these systems return the search results where the multiple query keywords disperse in several hyperlinked Web pages.

3.1.2 HITS Algorithm

Kleinberg [50] originally developed the HITS algorithm which was applied to the Web search engine in the CLEVER project [43]. This algorithm depends on the query and considers the set of pages S that point to, or are redirected to, pages in the answer. Web pages that have many links pointing to them in S are called *authorities*, while Web pages that have many outgoing links are called *hubs*. That is to say, better authorities come from incoming edges from good hubs and better hubs come from outgoing edges to good authorities. Let $H(p)$ and $A(p)$ be the hub and authority score of page p. These scores are defined such that the following equations are satisfied for all pages p:

$$H(p) = \sum_{u \in S | p \rightarrow u} A(u),$$
$$A(p) = \sum_{v \in S | v \rightarrow p} H(v),$$

where $H(p)$ and $A(p)$ are normalized for all Web pages. These scores can be determined through an iterative algorithm, and they converge to the principal eigenvector of the link matrix of S.

Several researchers have extended the above original HITS algorithm. For instance, the *ARC* algorithm of Chakrabarti et al. [16] enhanced the HITS algorithm with textual analysis. ARC computes a distance-2 neighborhood graph and weights edges. The weight of each edge is based on the match between the query terms and the text surrounding the hyperlink in the source document. In [7], Bharat et al. introduced additional heuristics to HITS algorithm by giving a document an authority weight of $1/k$ if the document is in a group of k documents on a first host which link to a single document on a second host, and a hub weight of $1/l$ if there are l links from the

document on a first host to a set of documents on a second host. However, the major problem in this technique is that the Web pages that the root document point to get the largest authoritiy scores because the hub score of the root page dominates all the others when a Web page has few in-links but a large number of out-links, most of which are not very relevant to the query. In order to solve this problem, Li et al. [54] proposed a new weighted HITS-based method that assigns appropriate weights to in-links of root Web pages and combined content analysis with HITS-based algorithm. In addition, Chakrabarti et al. [15, 18] considered not merely the text of Web page but also the Document Object Model (DOM) within the HTML. This improves on intermediate work in the CLEVER project [43] that broke hubs into pieces at logical HTML boundaries.

3.1.3 PageRank Algorithm

PageRank [66] simulates a user navigating randomly in the Web who jumps to a random page with probability d, or follows a random hyperlink with probability $1 - d$. It is further assumed that this user never returns to a previously visited page following an already traversed hyperlink backwards. This process can be modeled with a Markov chain, from which the stationary probability of being in each Web page can be computed. This value is then used as a part of the ranking mechanism employed by Google [11]. Let $C(a)$ be the number of outgoing links from Web page a and suppose that Web pages p_1 to p_n point to the Web page a. Then, the PageRank, $PR(a)$ of a is defined as:

$$PR(a) = d + (1 - d) \sum_{i=1}^{n} \frac{PR(p_i)}{C(p_i)},$$

where the value of d is empirically set to about 0.15-0.2 by the system. The weights of other Web pages are normalized by the number of links in the Web page. PageRank can be computed using an iterative algorithm, and corresponds to the principal eigenvector of the normalized link matrix of the Web. The major problems of this algorithm are that (1) the contents of Web pages are not analyzed, so the "importance" of a given Web page is independent of the query; and (2) specific famous Web sites tend to be ranked more highly. To yield more accurate search results, Rafiei et al. [69] proposed using the set of Web pages that contain some terms as a bias set for influencing the

PageRank computation, with the goal of returning terms for which a given page has a high reputation. Furthermore, Haveliwala [36] proposed computing a set of PageRank vectors to capture more accurately the notion of importance with respect to a particular topic. In the next section, we describe his work in more detail.

3.2 Related Work on Web Search Based on User's Preferences

There are several types of search systems that provide users with information more relevant to their individual needs. For example, we review related work of hyperlink-based personalized Web search, personalized Web sites, recommender systems, and personalized multimedia systems.

3.2.1 Hyperlink-Based Personalized Web Search

The field of Web information retrieval focuses on hyperlink structures of the Web, for example with Web search engines such as Google[1] [11] and the CLEVER project [43]. To address several problems with these engines, i.e., (1) the weight for a Web page is merely defined, and (2) the relativity of contents among hyperlinked Web pages is not considered, we proposed several approaches to refining the TF-IDF scheme for Web pages using their hyperlinked neighboring pages [92, 93, 95]. In personalized Web searches, the hyperlink structures of the Web are also becoming important. The use of personalized PageRank to enable personalized Web searches was first proposed in [66], where it was suggested as a modification of the global PageRank algorithm, which computes a universal notion of importance. The computation of (personalized) PageRank scores was not addressed beyond the original algorithm. Haveliwala [36] used personalized PageRank scores to enable "topic sensitive" Web searches as briefly mentioned in the previous section. Specifically, precomputed hub vectors corresponding to broad categories in ODP (Open Directory Project[2]) were used to bias importance scores, where the vectors and weights were selected according to the text query. Experiments in this work concluded that the use of personalized PageRank scores can

[1] http://www.google.com/
[2] http://dmoz.org/

improve a Web search. However, experiments were conducted based only on context associated with a query itself, and no experiments based on a user's context such as browsing patterns, bookmarks, and so on were conducted. Therefore, it is not clear if user-based personalization using this approach is effective. In addition, the number of hub vectors used was limited to 16 due to the computational requirements, which were not addressed in that work. In order to address this problem, Jeh et al. [46] proposed an approach that can scale well with the large size of hub vectors to realize personalized Web searches. On the other hand, Chang et al. [19] proposed algorithms for creating "personally customized *authority* documents" to correspond more closely to the user's internal model following the conventions of Kleinberg's HITS algorithm [50].

3.2.2 Personalized Web Sites

Link topology and the structure and contents of Web pages are often used in the construction of personalized Web sites. In this section, we review the framework of these systems with regard to "Link Personalization," "Content Personalization," and "Design-Oriented Personalization."

(a) Link Personalization

This scheme involves selecting the links that are more relevant to the user and changing the original navigation space by reducing or improving the relationships between Web pages. E-commerce applications use link personalization to recommend items based on the buying history of clients or some categorization of clients based on ratings and opinions. Users who give similar ratings to similar objects are presumed to have similar preferences, so when a user seeks recommendations about a certain product, the site suggests those recommendations that are most popular for his/her class or those that best correlate with the given product for that class. At the E-commerce site for Amazon.com[3] (the largest bookstore on the Internet), link personalization is widely used to link the homepage with recommendations, new releases, shopping groups, etc. that are personalized. At Amazon.com, this approach has been taken to an extreme by constructing a "New for you" home page and presenting it to each user, with new products that the user may be interested in. Additionally, Amazon.com uses implicit recommen-

[3] http://www.amazon.com/

dations via purchase history and/or explicit recommendations via "rate it" features to generate recommendations of products to purchase. In a recent study, Tsandilas et al. [102] proposed a system that automatically adapts links in the browsed pages based on their relevance to the weighted topics specified by sliders that users can manipulate.

(b) Content Personalization

In general, content personalization is done when pages present different information to different users. The difference between this and the link personalization described in Section 3.2.2(a) is subtle because part of the contents (i.e., the link anchors) presents different information when links are personalized. However, content personalization is referred to when substantial information in a Web page is personalized, unlike link anchors. For example, My Yahoo![4] [61] or My Netscape[5] filters the information that is relevant to the user, showing only sections and details in which the user may be interested. The user may explicitly indicate his/her preferences, or preferences may be inferred (semi-) automatically from his/her profile or from his/her navigation activity. At these sites, users choose a set of "modules" from a large set including weather, news, music and so on, and further personalize these modules by choosing a set of attributes of the module to be perceived. Some "automatic" customization may occur if the users input their zip code, for instance, when selecting a sport event they may be interested in. The approach followed in these applications is that the users should be able to "construct" their own pages and even the layout may be customized. However, user preferences or demographic information are acquired based on the prior questionnaire. Therefore, these sites have two problems: (1) the users' loads become high because these systems heavily rely on the users' inputs; (2) these sites cannot adapt to the changes in users' preferences unless the users change their previously registered preferences by themselves.

(c) Design-Oriented Personalization

Although design issues with regard to personalization are just now being introduced to the Web community, we can refer to the following approaches in this context of Web site personalization.

[4]http://www.my.yahoo.com/
[5]http://my.netscape.com/

The Web Site Design Method (WSDM) [101] focuses on the construction of "Audience-driven" Web applications. Requirements for each potential user profile are systematically gathered and a class diagram of user profiles is constructed. Different navigation tracks are specified for each audience though there is no notation for expressing individual differences. Web Modeling Language (WebML) [14] is a notation for specifying complex Web sites at the conceptual level. Using this WebML modeling language and its supporting software, it is possible to specify the structure of the application's information base (structural model), the structure of nodes (composition model), the topology of links between pages (navigation model), the layout and graphic requirements for page rendering (presentation model), and the customization features for one-to-one content delivery (personalization model). WebML allows well-known Web patterns and supports data derivation and user modeling. Personalization in WebML is expressed using event-condition-action rules. While WebML is a data-oriented approach, Rossi et al. [74] developed an object-oriented method for personalization based on the idea that Web applications are hypermedia applications because users navigate a hypermedia information space composed of Web pages connected by hyperlinks. They presented Object-Oriented Hypermedia Design Method (OOHDM) approaches to constructing personalized Web applications focusing on a design-oriented discussion of personalization. In their approach, different Web applications for different profiles can be constructed by simply reusing a conceptual schema. In addition, they showed that their notation and the underlying design framework bring concise specifications by reusing existing ones. However, these methods also do not consider the dynamic changes in users' preferences.

3.2.3 Recommender Systems

It has become increasingly difficult to search for useful information on the Web because the amount of information on the Web continues to grow. Therefore, we get the feeling of being overwhelmed by the number of choices. This situation is often referred to as "information overload." As one of the most promising approaches to alleviate this overload, recommender systems have emerged in domains such as E-commerce, digital libraries, and knowledge management. These systems provide personalized suggestions based on user preferences. Recommender systems collect user feedback in the form of ratings for items in a given domain and exploit similari-

ties and differences among profiles of several users in determining how to recommend an item. There are two prevalent approaches to constructing recommender systems – collaborative filtering-based and content-based recommendation.

(a) Collaborative Filtering-Based Recommendation

Collaborative filtering-based recommendation is the most successful recommendation technique to date. The term collaborative filtering was coined by Goldberg et al. [34]. Collaborative filtering means that people collaborate to help one another perform filtering by recording their reactions to documents they read. Based on this concept, Goldberg et al. developed a system called Tapestry that is one of the earliest implementations of collaborative filtering-based recommendation. This system is used to filter email and it allows users to annotate messages. Annotations became accessible as virtual fields of the messages, and users can formulate filtering queries which access these fields. Users then create queries such as "show me all office memos that Bill thought were important." The collaborative filtering provided by Tapestry was not automated, and users were required to formulate complex queries in a special query language designed for the task. In addition, this system relied on explicit opinions of people from a close-knit community, such as a group of office workers. However, recommender systems for large communities generally cannot depend on everyone knowing each other. Therefore, the framework in Tapestry is not appropriate to systems for large communities.

Rating-based automated collaborative filtering is quickly becoming a popular approach to reducing information overload by providing personalized recommendations for information, products or services. For example, the k-nearest neighbor collaborative filtering-based systems are achieving widespread success on the Web. The GroupLens research system [71, 51], which filters Usenet news, first introduced an automated collaborative filtering system using the k-nearest neighbor-based algorithm. In this algorithm, a subset of appropriate k users is chosen based on their similarity to the active user, and a weighted aggregate of their rating is used to generate predictions for the active user. GroupLens then recommends Usenet news articles to these active users.

While the Tapestry and GroupLens rely on explicit ratings, some systems rely on implicit ratings. For example, Morita et al. [65] exploit "time-spent-reading" as a

measure of implicit ratings. They observed strong positive correlations between the time users spent reading messages and the personal interest ratings of those messages. Their work suggests that it might be possible for time-spent-reading measures to stand in for ratings, further reducing user tasks. PHOAKS (People Helping One Another Know Stuff) [98] also uses implicit ratings to construct a recommender system by examining Usenet news postings to find "endorsements" of Web sites. It then creates a listing of the top Web sites endorsed in each newsgroup. Some recommender systems also explore user preferences transparently without any extra effort from the users like the recommender systems relying on implicit ratings described above. For example, Letizia [56, 57] and WebWatcher [47] infer user preferences by observing user-browsing behavior. However, the main shortcomings in Letizia and WebWatcher are that they maintain persistent and slowly-changing user models and overlook the fact that different browsing sessions by the same user or even a single session may involve different user interests and goals. Moreover, Kelly et al. [49] have published a nice summary with regard to the systems using implicit measures.

Furthermore, at the E-commerce sites such as Amazon.com, CDnow.com (the largest CD store on the Web) and MovieFinder.com (one of the most visited movie sites), automated collaborative filtering systems have been used with considerable success. Their widespread use, however, has exposed some of their limitations such as (1) sparsity of the user-item rating matrix, (2) scalability with the growth of the number of users and the number of item, and (3) cold-start problems where recommendations are required for items that no one has yet rated. The problems of sparsity are addressed in [81, 35] by incorporating semi-intelligent filtering agents into the system. The problems of scalability and high dimensionality in recommender systems are discussed in [8, 41]. Both sparsity and scalability issues are addressed simultaneously in [79]. Moreover, the cold-start problem is dealt with in a probabilistic model that combines content and collaborative information by using expectation maximization learning in [84].

(b) Content-Based Recommendation

A content-based approach provides recommendations by comparing representations of content contained in an item with representations of content that the user is interested in. In this approach, a model of user ratings is first developed. Algorithms in this cat-

egory use probabilities and envision the collaborative filtering process by computing the expected value of a user prediction given the user's ratings on other items. The model building process is performed by three different machine learning algorithms: (1) Bayesian network, (2) clustering, and (3) rule-based models. The Bayesian network model [10] constructs a probabilistic model for a collaborative filtering problem. Clustering model addresses collaborative filtering as a classification problem [6, 10]. It works by clustering similar users in the same class and estimating the probability that a particular user is in a particular class; from there it computes the conditional probability of ratings. The rule-based model applies association rule discovery algorithms to find associations between co-purchased items. It then generates item recommendations based on the strength of the association between items [80].

The systems described in Section 3.2.3(a) only provide recommendations based on collaborative filtering. However, some systems provide better recommendations by combining collaborative filtering with content information. Fab [5] uses relevance feedback to simultaneously construct a personal filter along with a communal "topic" filter. Web pages are initially ranked by the topic filter and then sent to user's personal filters. The user then provides relevance feedback for that Web page, and this feedback is used to modify both the personal filter and the originating topic filter. Basu et al. [6] integrate content and collaboration in a framework where they treat recommendation as a classification task. Melville et al. [63] overcome drawbacks of collaborative filtering systems in their recommender system by exploiting content information of items already rated. In recent study on recommender systems, Schafer et al. [83] introduce a new class of recommender system that provides users with personalized control over the generation of a single recommendation list formed from a combination of rich data using multiple information resources and recommendation techniques.

3.2.4 Personalized Multimedia Systems

Systems related to personalization on the Web seems to be mainly based on text retrieval. However, personalized systems in the field of multimedia are also being developed. In music, Ringo [87] uses collaborative filtering techniques to provide users with recommendations for music albums and artists. In addition, Ringo has support for message boards (independent of the recommender system), where users can discuss their music preferences. Cutomized Internet Radio (CIR) [53] is an application

that schedules content retrieval from multiple Web radio stations based on a schedule that a user configures. Field et al. [28] developed a system for personalized audio called *Personal DJ*. Bellcore Video recommender systems [39] are email and Web-based and they generate recommendations on movies. Merialdo et al. [64] constructed a system that personalizes TV news program to optimize the content value for a specific user based on manual categorization and automatic keyword extraction. PTV [90] is a content personalization system that provides personalized TV listings to users. In [40], an application for providing and managing personalized, interactive video on the Web using Synchronized Multimedia Integration Language (SMIL) [99] is described.

Chapter 4

Web Search Based on its Hyperlink Structures

In information retrieval systems based on the vector space model, the TF-IDF scheme is widely used to characterize documents. However, in the case of documents with hyperlink structures such as Web pages, it is necessary to develop a technique for representing the contents of Web pages more accurately by exploiting the contents of their hyperlinked neighboring pages. In this chapter, we first propose several approaches to refining the TF-IDF scheme for a target Web page by using the contents of its hyperlinked neighboring pages, and then compare the retrieval accuracy of our proposed approaches. Experimental results show that, generally, more accurate feature vectors of a target Web page can be generated in the case of utilizing the contents of its hyperlinked neighboring pages at levels up to second in the backward direction from the target page.

4.1 Introduction

The World Wide Web (WWW) is a useful resource for users to obtain a great variety of information. Three billion Web pages are the lower bound that comes from the coverage of search engines [4], and it is obvious that the number of Web pages continues to grow. Therefore, it is getting more and more difficult for users to find relevant information on the WWW. Under these circumstances, Web search engines are one of the most popular methods for finding valuable information effectively, and they are classified

into two generations based on their indexing techniques [12]. In the first-generation search engines developed in the early stages of the Web, only terms included in Web pages were utilized as indices. Therefore, the traditional document retrieval technique was merely applied to Web page searches. However, Web pages have peculiar features such as hyperlink structures or are in numbers too large to search effectively. Consequently, users are not satisfied with the ease of use and retrieval accuracy of the search engines because such features of Web pages are not exploited in the first-generation search engines.

To deal with these problems, in the second-generation search engines, the hyperlink structures of Web pages are taken into account. For example, the approaches called HITS (Hypertext Induced Topic Search) [50] (see Section 3.1.2) and PageRank [66] (see Section 3.1.3) are applied to the search engine of the CLEVER project [43] and Google[1] [11], respectively. In these algorithms, weighting Web pages based on hyperlink structures achieves higher retrieval accuracy compared with the first-generation search engines. However, these algorithms have shortcomings in that (1) the importance for a Web page is merely defined; and (2) the relativity of contents among hyperlinked Web pages is not considered. Taking these points into account, Davison [24] concentrated on textual content and showed that Web pages are siginificantly more likely to be related topically to pages to which they are linked. Based on this finding, his research group has released the search engine "Teoma[2]" that does context-sensitive HITS on the lines of the CLEVER project. This search engine uses the concept of "Subject-Specific Popularity [2]," which ranks a site based on the number of same-subject pages that reference it, not just general popularity, to determine a site's level of authority. However, the problem of Web pages irrelevant to a user's query often being ranked highly still remains. Hence, in order to provide users with relevant Web pages, it is necessary to develop a technique for representing the contents of Web pages more accurately. In order to achieve this purpose, we have proposed some methods for improving a feature vector for a target Web page [93]. Our proposed methods, however, also have a problem in that only Web pages out-linked from a target Web page are used in order to generate the feature vector of the target Web page. Since Web pages usually have in- and out- linked pages, in the case of generating a more accurate feature vector

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

[2] http://www.teoma.com/

of a Web page, it is necessary to use both its in- and out- linked pages. Therefore, in this chapter, we first propose three approaches to refining the TF-IDF scheme [78] for a target Web page using both its in- and out-linked pages in order to represent the contents of the target Web page more accurately. Then, we compare retrieval accuracy of our proposed approaches using the refined feature vector. Our approach is novel in refining TF-IDF based feature vectors of target Web pages by reflecting the contents of their hyperlinked neighboring Web pages.

4.2 Proposed Methods

As we described in Section 3.1.1, the information retrieval systems based on the concept of "optimal document granularity" have a problem, in that the search results are incomprehensible for users. Moreover, HITS [50] and PageRank [11] also have problems: (1) the weight for a Web page is merely defined; and (2) the relativity of contents among hyperlinked Web pages is not considered.

On the basis of these problems, the feature vector of a Web page should be generated by using the contents of its hyperlinked neighboring pages in order to represent the contents of Web pages more accurately. We, therefore, propose refining the TF-IDF scheme for a target Web page by using the contents of its hyperlinked neighboring pages. Unlike researches described in Section 3.1, our approach is novel in refining the TF-IDF based feature vector of a target Web page by reflecting the contents of its hyperlinked neighboring Web pages. Our approach is query-independent, and link-based computations are performed offline like PageRank algorithm. At query time, we only compute the similarity between the refined feature vector and query vector formulated by user specified query. Therefore, the query-time costs are not much greater than the HITS algorithm, whose query-time costs depends on the query.

In the following discussion, let a target page be p_{tgt}. Then, we define i as the length of the shortest directed path from p_{tgt} to its hyperlinked neighboring pages. Let us assume that there are N_i Web pages $(p_{i_1}, p_{i_2}, \cdots, p_{i_{N_i}})$ in the i^{th} level from p_{tgt}. Moreover, we denote the feature vector $\boldsymbol{w}^{p_{tgt}}$ of p_{tgt} as follows:

$$\boldsymbol{w}^{p_{tgt}} = (w_{t_1}^{p_{tgt}}, w_{t_2}^{p_{tgt}}, \cdots, w_{t_m}^{p_{tgt}}), \tag{4.1}$$

where m is the number of distinct terms in the Web page collection, and t_k $(k =$

$1, 2, \cdots, m)$ denotes each term. Using the TF-IDF scheme, we also define each element $w_{t_k}^{p_{tgt}}$ of $\boldsymbol{w}^{p_{tgt}}$ as follows:

$$w_{t_k}^{p_{tgt}} = \frac{tf(t_k, p_{tgt})}{\sum_{s=1}^{m} tf(t_s, p_{tgt})} \cdot \log \frac{N_{web}}{df(t_k)}, \tag{4.2}$$

where $tf(t_k, p_{tgt})$ is the frequency of term t_k in the target page p_{tgt}, N_{web} is the total number of Web pages in the collection, and $df(t_k)$ is the number of Web pages in which term t_k appears. Below, we refer to $\boldsymbol{w}^{p_{tgt}}$ as the "*initial feature vector*." Subsequently, we denote the refined feature vector $\boldsymbol{w}'^{p_{tgt}}$ as follows:

$$\boldsymbol{w}'^{p_{tgt}} = (w_{t_1}'^{p_{tgt}}, w_{t_2}'^{p_{tgt}}, \cdots, w_{t_m}'^{p_{tgt}}),$$

and refer to this $\boldsymbol{w}'^{p_{tgt}}$ as the "*refined feature vector*." In this chapter, we propose three approaches to refining the "initial feature vector" based on the TF-IDF scheme defined by Equation (4.2) as follows:

Method I
the approach relies on the contents of all Web pages at levels up to ${L_{(in)}}^{th}$ in the backward direction and levels up to ${L_{(out)}}^{th}$ in the forward direction from the target page p_{tgt},

Method II
the approach relies on the centroid vectors of clusters generated from Web page groups created at each level up to ${L_{(in)}}^{th}$ in the backward direction and each level up to ${L_{(out)}}^{th}$ in the forward direction from the target page p_{tgt},

Method II
the approach relies on the centroid vectors of clusters generated from Web page groups created at levels up to ${L_{(in)}}^{th}$ in the backward direction and levels up to ${L_{(out)}}^{th}$ in the forward direction from the target page p_{tgt}.

4.2.1 Method I

In this approach, we reflect the contents of all Web pages at levels up to ${L_{(in)}}^{th}$ in the backward direction and levels up to ${L_{(out)}}^{th}$ in the forward direction from the target page p_{tgt}. Based on the ideas that (1) there are Web pages similar to the contents of p_{tgt} in the neighborhood Web pages of p_{tgt}; and (2) since on one hand such Web pages exist right near p_{tgt}, on the other hand they might exist far removed from p_{tgt} in the vector space, we reflect the following two factors on each element of the initial feature vector $\boldsymbol{w}^{p_{tgt}}$:

Method I–i
the length of the shortest directed path in the backward or forward direction from p_{tgt} to its hyperlinked neighboring pages in the Web space,

Method I–ii
the distance between $\boldsymbol{w}^{p_{tgt}}$ and feature vector of in- and out-linked pages of p_{tgt} in the vector space.

For example, Figure 4.1(a) shows that $\boldsymbol{w}'^{p_{tgt}}$ is generated by reflecting the contents of all Web pages at levels up to second in the backward and forward directions from p_{tgt} on $\boldsymbol{w}^{p_{tgt}}$. In Figure 4.1(a), $p_{ij_{(in)}}$ and $p_{ij_{(out)}}$ correspond to the j^{th} page in the i^{th} level in the backward and forward directions from p_{tgt}, respectively. In addition, Figure 4.1(b) shows that refined feature vector $\boldsymbol{w}'^{p_{tgt}}$ is generated by reflecting each feature vector of in- and out-linked pages of p_{tgt} on the initial feature vector $\boldsymbol{w}^{p_{tgt}}$.

In Method I–i, each element $w_{t_k}'^{p_{tgt}}$ of $\boldsymbol{w}'^{p_{tgt}}$ is defined as follows:

$$
\begin{aligned}
w_{t_k}'^{p_{tgt}} &= w_{t_k}^{p_{tgt}} \\
&+ \left(\sum_{i=1}^{L_{(in)}} \sum_{j=1}^{N_{i_{(in)}}} \frac{w_{t_k}^{p_{ij_{(in)}}}}{N_{i_{(in)}} \cdot i} \right) \\
&+ \left(\sum_{i=1}^{L_{(out)}} \sum_{j=1}^{N_{i_{(out)}}} \frac{w_{t_k}^{p_{ij_{(out)}}}}{N_{i_{(out)}} \cdot i} \right). \qquad (4.3)
\end{aligned}
$$

Equation (4.3) shows that the product of $w_{t_k}^{p_{ij_{(in)}}}$ (weight of term t_k in Web page $p_{ij_{(in)}}$) and the reciprocal of i (the length of the shortest directed path in the backward direction

from p_{tgt} to its hyperlinked neighboring pages), and similarly, the product of $w_{t_k}^{p_{ij_{(out)}}}$ (weight of term t_k in Web page $p_{ij_{(out)}}$) and the reciprocal of i (the length of the shortest directed path in the forward direction from p_{tgt} to its hyperlinked neighboring pages) is added to $w_{t_k}^{p_{tgt}}$ (weight of term t_k in p_{tgt}, computed by Equation (4.2)) with respect to all Web pages at levels up to ${L_{(in)}}^{th}$ in the backward direction and levels up to ${L_{(out)}}^{th}$ in the forward direction from p_{tgt}.

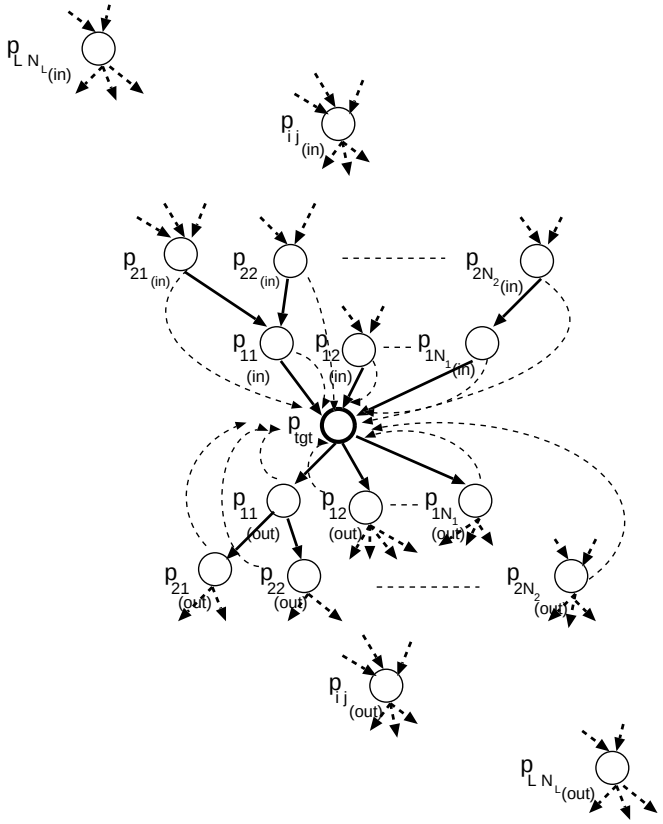
In Method I-ii, each element $w_{t_k}^{\prime p_{tgt}}$ of $\boldsymbol{w}^{\prime p_{tgt}}$ is defined as follows:

$$\begin{aligned} w_{t_k}^{\prime p_{tgt}} &= w_{t_k}^{p_{tgt}} \\ &+ \frac{1}{Dim}\left(\sum_{i=1}^{L_{(in)}}\sum_{j=1}^{N_{i_{(in)}}}\frac{w_{t_k}^{p_{ij_{(in)}}}}{N_{i_{(in)}}\cdot dis(\boldsymbol{w}^{p_{tgt}},\boldsymbol{w}^{p_{ij_{(in)}}})}\right) \\ &+ \frac{1}{Dim}\left(\sum_{i=1}^{L_{(out)}}\sum_{j=1}^{N_{i_{(out)}}}\frac{w_{t_k}^{p_{ij_{(out)}}}}{N_{i_{(out)}}\cdot dis(\boldsymbol{w}^{p_{tgt}},\boldsymbol{w}^{p_{ij_{(out)}}})}\right). \end{aligned} \tag{4.4}$$

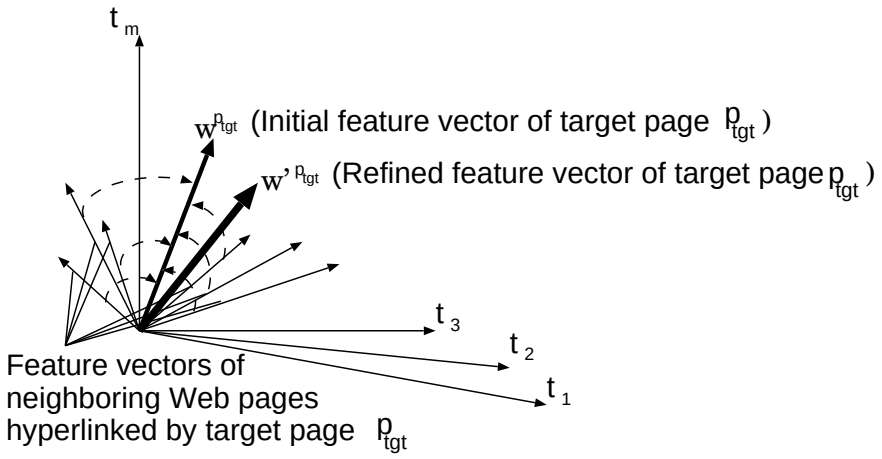
Equation (4.4) shows that the product of $w_{t_k}^{p_{ij_{(in)}}}$ (weight of term t_k in Web page $p_{ij_{(in)}}$) and the reciprocal of $dis(\boldsymbol{w}^{p_{tgt}},\boldsymbol{w}^{p_{ij_{(in)}}})$ (the distance between $\boldsymbol{w}^{p_{tgt}}$ and $\boldsymbol{w}^{p_{ij_{(in)}}}$ in the vector space), and similarly, the product of $w_{t_k}^{p_{ij_{(out)}}}$ (weight of term t_k in Web page $p_{ij_{(out)}}$) and the reciprocal of $dis(\boldsymbol{w}^{p_{tgt}},\boldsymbol{w}^{p_{ij_{(out)}}})$ (the distance between $\boldsymbol{w}^{p_{tgt}}$ and $\boldsymbol{w}^{p_{ij_{(out)}}}$ in the vector space) is added to $w_{t_k}^{p_{tgt}}$ (weight of term t_k in p_{tgt}, computed by Equation (4.2)) with respect to all Web pages at levels up to ${L_{(in)}}^{th}$ in the backward direction and levels up to ${L_{(out)}}^{th}$ in the forward direction from p_{tgt}. If the distance between $\boldsymbol{w}^{p_{tgt}}$ and $\boldsymbol{w}^{p_{ij_{(in)}}}, \boldsymbol{w}^{p_{ij_{(out)}}}$ in the vector space is very close, the values of the second and third terms of Equation (4.4) can be dominant compared with the first term $w_{t_k}^{p_{tgt}}$. Therefore, in order to prevent this phenomenon, we also define Dim, which denotes the number of distinct terms in the Web page collection. $dis(\boldsymbol{w}^{p_{tgt}},\boldsymbol{w}^{p_{ij_{(in)}}})$ and $dis(\boldsymbol{w}^{p_{tgt}},\boldsymbol{w}^{p_{ij_{(out)}}})$ are defined by the following equations, respectively:

$$dis(\boldsymbol{w}^{p_{tgt}},\boldsymbol{w}^{p_{ij_{(in)}}}) = \sqrt{\sum_{k=1}^{m}(w_{t_k}^{p_{tgt}} - w_{t_k}^{p_{ij_{(in)}}})^2},$$

$$dis(\boldsymbol{w}^{p_{tgt}},\boldsymbol{w}^{p_{ij_{(out)}}}) = \sqrt{\sum_{k=1}^{m}(w_{t_k}^{p_{tgt}} - w_{t_k}^{p_{ij_{(out)}}})^2}.$$



(a)



(b)

Figure 4.1. The refinement of a feature vector as performed by Method I [(a) in the Web space, (b) in the vector space].

4.2.2 Method II

In this approach, we first construct Web page groups $G_{i_{(in)}}$ at each level up to ${L_{(in)}}^{th}$ in the backward direction, and $G_{i_{(out)}}$ at each level up to ${L_{(out)}}^{th}$ in the forward direction from the target page p_{tgt}. Then, we generate $\boldsymbol{w}'^{p_{tgt}}$ by reflecting centroid vectors of clusters generated from $G_{i_{(in)}}$ and $G_{i_{(out)}}$ on the initial feature vector $\boldsymbol{w}^{p_{tgt}}$. This approach is based on the idea that Web pages at each level in the backward and forward directions from p_{tgt} are classified into some topics in each level. In addition, we reflect the following two factors on each element of the initial feature vector $\boldsymbol{w}^{p_{tgt}}$:

Method II–i
the length of the shortest directed path in the backward or forward direction from p_{tgt} to its hyperlinked neighboring pages in the Web space,

Method II–ii
the distance between $\boldsymbol{w}^{p_{tgt}}$ and the centroid vectors of the clusters in the vector space.

In other words, we first create Web page groups $G_{i_{(in)}}$ and $G_{i_{(out)}}$ which are defined by Equation (4.5) and (4.6) as follows:

$$G_{i_{(in)}} = \{p_{i1_{(in)}}, p_{i2_{(in)}}, \cdots, p_{iN_{i_{(in)}}}\}, \tag{4.5}$$

$$G_{i_{(out)}} = \{p_{i1_{(out)}}, p_{i2_{(out)}}, \cdots, p_{iN_{i_{(out)}}}\}, \tag{4.6}$$

$$(i = 1, 2, \cdots, L),$$

and then produce K clusters in each Web page group $G_{i_{(in)}}$ and $G_{i_{(out)}}$ by means of the K-means algorithm [60]. The centroid vectors $\boldsymbol{w}^{g_{ic_{(in)}}}$ and $\boldsymbol{w}^{g_{ic_{(out)}}}(c = 1, 2, \cdots, K)$ are produced in $G_{i_{(in)}}$ and $G_{i_{(out)}}$, respectively. We generate a refined feature vector $\boldsymbol{w}'^{p_{tgt}}$ by reflecting the two factors described above on each element of the initial feature vector $\boldsymbol{w}^{p_{tgt}}$. For instance, Figure 4.2(a) shows that we construct Web page groups $G_{1_{(in)}}, G_{2_{(in)}}, G_{1_{(out)}}$, and $G_{2_{(out)}}$ at each level up to the second in the backward and forward directions from p_{tgt}, and generate a refined feature vector $\boldsymbol{w}'^{p_{tgt}}$ by reflecting the centroid vectors of each cluster produced in each Web page group $G_{1_{(in)}}, G_{2_{(in)}}$, $G_{1_{(out)}}$, and $G_{2_{(out)}}$, on $\boldsymbol{w}^{p_{tgt}}$. Moreover, Figure 4.2(b) shows that the refined feature vector $\boldsymbol{w}'^{p_{tgt}}$ is generated by reflecting the centroid vectors of each cluster on $\boldsymbol{w}^{p_{tgt}}$.

In Method II–i, we define each element $w_{t_k}^{\prime p_{tgt}}$ of $\boldsymbol{w}^{\prime p_{tgt}}$ as follows:

$$\begin{aligned} w_{t_k}^{\prime p_{tgt}} &= w_{t_k}^{p_{tgt}} \\ &+ \left(\sum_{i=1}^{L_{(in)}} \sum_{c=1}^{K} \frac{w_{t_k}^{g_{ic_{(in)}}}}{i} \right) \\ &+ \left(\sum_{i=1}^{L_{(out)}} \sum_{c=1}^{K} \frac{w_{t_k}^{g_{ic_{(out)}}}}{i} \right). \end{aligned} \tag{4.7}$$

Equation (4.7) shows that the product of $w_{t_k}^{g_{ic_{(in)}}}$ (weight of term t_k in centroid vector $\boldsymbol{w}^{g_{ic_{(in)}}}$ of cluster c constructed from $G_{i_{(in)}}$) and the reciprocal of i (the length of the shortest directed path in the backward direction from p_{tgt} to its hyperlinked neighboring pages), and similarly, the product of $w_{t_k}^{g_{ic_{(out)}}}$ (weight of term t_k in centroid vector $\boldsymbol{w}^{g_{ic_{(out)}}}$ of cluster c constructed from $G_{i_{(out)}}$) and the reciprocal of i (the length of the shortest directed path in the forward direction from p_{tgt} to its hyperlinked neighboring pages) are added to $w_{t_k}^{p_{tgt}}$ (weight of term t_k in p_{tgt}, computed by Equation (4.2)) with respect to all centroid vectors constructed at each level up to ${L_{(in)}}^{th}$ in the backward direction and each level up to ${L_{(out)}}^{th}$ in the forward direction from p_{tgt}.

In Method II–ii, we define each element $w_{t_k}^{\prime p_{tgt}}$ of $\boldsymbol{w}^{\prime p_{tgt}}$ as follows:

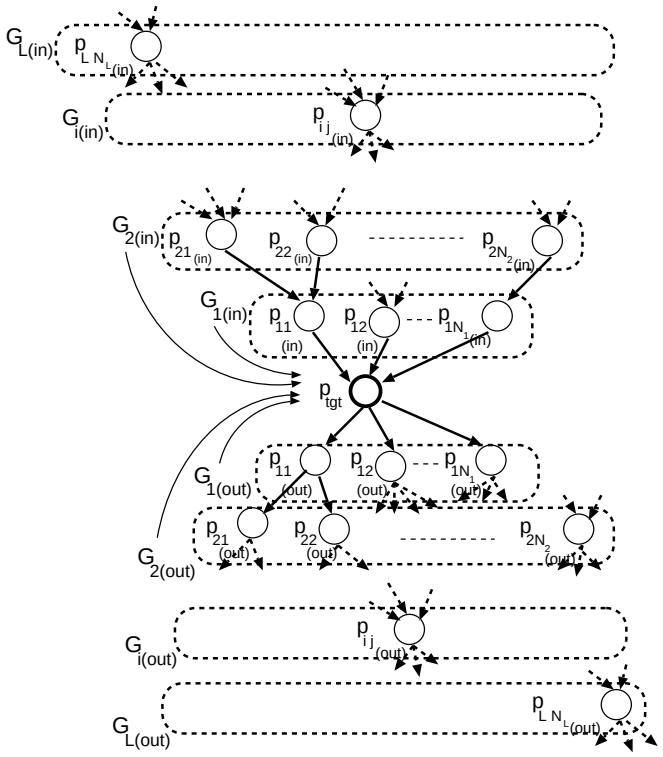
$$\begin{aligned} w_{t_k}^{\prime p_{tgt}} &= w_{t_k}^{p_{tgt}} \\ &+ \frac{1}{Dim} \left(\sum_{i=1}^{L_{(in)}} \sum_{c=1}^{K} \frac{w_{t_k}^{g_{ic_{(in)}}}}{dis(\boldsymbol{w}^{p_{tgt}}, \boldsymbol{w}^{g_{ic_{(in)}}})} \right) \\ &+ \frac{1}{Dim} \left(\sum_{i=1}^{L_{(out)}} \sum_{c=1}^{K} \frac{w_{t_k}^{g_{ic_{(out)}}}}{dis(\boldsymbol{w}^{p_{tgt}}, \boldsymbol{w}^{g_{ic_{(out)}}})} \right). \end{aligned} \tag{4.8}$$

Equation (4.8) shows that the product of $w_{t_k}^{g_{ic_{(in)}}}$ (weight of term t_k in centroid vector $\boldsymbol{w}^{g_{ic_{(in)}}}$ of cluster c constructed from $G_{i_{(in)}}$) and the reciprocal of $dis(\boldsymbol{w}^{p_{tgt}}, \boldsymbol{w}^{g_{ic_{(in)}}})$ (the distance between $\boldsymbol{w}^{p_{tgt}}$ and $\boldsymbol{w}^{g_{ic_{(in)}}}$ in the vector space), and similarly, the product of $w_{t_k}^{g_{ic_{(out)}}}$ (weight of term t_k in centroid vector $\boldsymbol{w}^{g_{ic_{(out)}}}$ of cluster c constructed from $G_{i_{(out)}}$) and the reciprocal of $dis(\boldsymbol{w}^{p_{tgt}}, \boldsymbol{w}^{g_{ic_{(out)}}})$ (the distance between $\boldsymbol{w}^{p_{tgt}}$ and $\boldsymbol{w}^{g_{ic_{(out)}}}$ in the vector space) are added to $w_{t_k}^{p_{tgt}}$ (weight of term t_k in p_{tgt}, computed by Equation (4.2)) with respect to all centroid vectors constructed at each level up to ${L_{(in)}}^{th}$ in the backward direction and each level up to ${L_{(out)}}^{th}$ in the forward direction

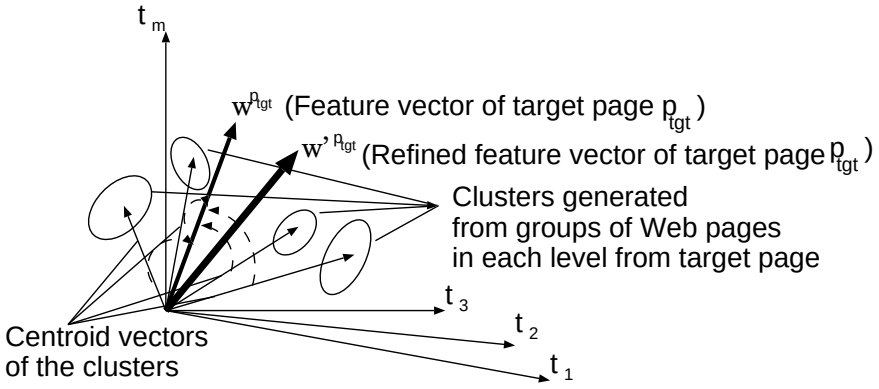
from p_{tgt}. In addition, we introduce Dim for the purpose of preventing the values of the second and third terms from dominating compared with the first term in Equation (4.8). $dis(\boldsymbol{w}^{p_{tgt}}, \boldsymbol{w}^{g_{ic_{(in)}}})$ and $dis(\boldsymbol{w}^{p_{tgt}}, \boldsymbol{w}^{g_{ic_{(out)}}})$ are defined as follows:

$$dis(\boldsymbol{w}^{p_{tgt}}, \boldsymbol{w}^{g_{ic_{(in)}}}) = \sqrt{\sum_{k=1}^{m} (w_{t_k}^{p_{tgt}} - w_{t_k}^{g_{ic_{(in)}}})^2},$$

$$dis(\boldsymbol{w}^{p_{tgt}}, \boldsymbol{w}^{g_{ic_{(out)}}}) = \sqrt{\sum_{k=1}^{m} (w_{t_k}^{p_{tgt}} - w_{t_k}^{g_{ic_{(out)}}})^2}.$$



(a)



(b)

Figure 4.2. The refinement of a feature vector as performed by Method II [(a) in the Web space, (b) in the vector space].

4.2.3 Method III

This approach is based on the idea that Web pages at levels up to ${L_{(in)}}^{th}$ in the backward direction and levels up to ${L_{(out)}}^{th}$ in the forward direction from target page p_{tgt} is composed of some topics. According to this idea, we cluster the set of all Web pages at levels up to ${L_{(in)}}^{th}$ in the backward direction and levels up to ${L_{(out)}}^{th}$ in the forward direction from p_{tgt}, and generate $\boldsymbol{w}'^{p_{tgt}}$ by reflecting centroid vectors of the clusters on the initial feature vector $\boldsymbol{w}^{p_{tgt}}$. Furthermore, we reflect the following two factors on each element of the initial feature vector $\boldsymbol{w}^{p_{tgt}}$:

Method III–i
the length of the shortest directed path in the backward or forward direction from p_{tgt} to its hyperlinked neighboring pages in the Web space,

Method III–ii
the distance between $\boldsymbol{w}^{p_{tgt}}$ and the centroid vector of the cluster in the vector space.

In other words, we create Web page groups $G_{i_{(in)}}$ and $G_{i_{(out)}}$ as defined by Equation (4.9) and (4.10), respectively,

$$\begin{aligned} G_{i_{(in)}} = \{&p_{11_{(in)}}, p_{12_{(in)}}, \cdots, p_{1N_{1_{(in)}}}, \\ &p_{21_{(in)}}, p_{22_{(in)}}, \cdots, p_{2N_{2_{(in)}}}, \\ &p_{i1_{(in)}}, p_{i2_{(in)}}, \cdots, p_{iN_{i_{(in)}}}\}, \end{aligned} \tag{4.9}$$

$$\begin{aligned} G_{i_{(out)}} = \{&p_{11_{(out)}}, p_{12_{(out)}}, \cdots, p_{1N_{1_{(out)}}}, \\ &p_{21_{(out)}}, p_{22_{(out)}}, \cdots, p_{2N_{2_{(out)}}}, \\ &p_{i1_{(out)}}, p_{i2_{(out)}}, \cdots, p_{iN_{i_{(out)}}}\}, \\ &(i = 1, 2, \cdots, L), \end{aligned} \tag{4.10}$$

and produce K clusters in $G_{i_{(in)}}$ and $G_{i_{(out)}}$ by means of the K-means algorithm [60]. The centroid vectors $\boldsymbol{w}^{g_{c_{(in)}}}$ and $\boldsymbol{w}^{g_{c_{(out)}}}(c = 1, 2, \cdots K)$ are produced in $G_{i_{(in)}}$ and $G_{i_{(out)}}$, respectively. Then, we generate the refined feature vector $\boldsymbol{w}'^{p_{tgt}}$ by reflecting the two factors described above on each element of the initial feature vector $\boldsymbol{w}^{p_{tgt}}$. For instance, Figure 4.3(a) shows that we construct Web page groups $G_{2_{(in)}}$ and $G_{2_{(out)}}$ at levels up to second in the backward and forward directions from p_{tgt}, and generate

refined feature vector $\boldsymbol{w}'^{p_{tgt}}$ by reflecting the centroid vectors of clusters produced in Web page group $G_{2_{(in)}}$ and $G_{2_{(out)}}$ on the initial feature vector $\boldsymbol{w}^{p_{tgt}}$. Furthermore, Figure 4.3(b) shows that refined feature vector $\boldsymbol{w}'^{p_{tgt}}$ is generated by reflecting centroid vectors of each cluster on the initial feature vector $\boldsymbol{w}^{p_{tgt}}$.

In Method III–i, each element $w_{t_k}'^{p_{tgt}}$ of $\boldsymbol{w}'^{p_{tgt}}$ is defined as follows:

$$\begin{aligned} w_{t_k}'^{p_{tgt}} &= w_{t_k}^{p_{tgt}} \\ &+ \left(\sum_{c=1}^{K} \frac{w_{t_k}^{g_{c_{(in)}}}}{i} \right) \\ &+ \left(\sum_{c=1}^{K} \frac{w_{t_k}^{g_{c_{(out)}}}}{i} \right). \end{aligned} \tag{4.11}$$

Equation (4.11) shows that the product of $w_{t_k}^{g_{c_{(in)}}}$ (weight of term t_k in centroid vector $\boldsymbol{w}^{g_{c_{(in)}}}$ of cluster c generated from $G_{i_{(in)}}$) and the reciprocal of i (the length of the shortest directed path in the backward direction from p_{tgt} to its hyperlinked neighboring pages), and similarly the product of element $w_{t_k}^{g_{c_{(out)}}}$ (weight of term t_k in centroid vector $\boldsymbol{w}^{g_{c_{(out)}}}$ of cluster c generated from $G_{i_{(out)}}$) and the reciprocal of i (the length of the shortest directed path in the forward direction from p_{tgt} to its hyperlinked neighboring pages) are added to $w_{t_k}^{p_{tgt}}$ (weight of term t_k in p_{tgt} computed by Equation (4.2)), with respect to the number of clusters K.

In Method III–ii, each element $w_{t_k}'^{p_{tgt}}$ of $\boldsymbol{w}'^{p_{tgt}}$ is defined as follows:

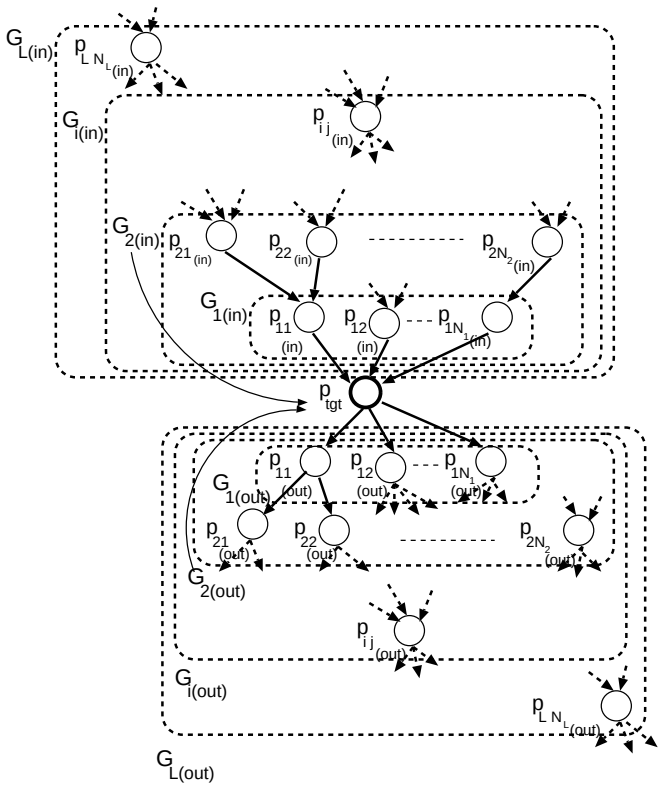
$$\begin{aligned} w_{t_k}'^{p_{tgt}} &= w_{t_k}^{p_{tgt}} \\ &+ \frac{1}{Dim} \left(\sum_{c=1}^{K} \frac{w_{t_k}^{g_{c_{(in)}}}}{dis(\boldsymbol{w}^{p_{tgt}}, \boldsymbol{w}^{g_{c_{(in)}}})} \right) \\ &+ \frac{1}{Dim} \left(\sum_{c=1}^{K} \frac{w_{t_k}^{g_{c_{(out)}}}}{dis(\boldsymbol{w}^{p_{tgt}}, \boldsymbol{w}^{g_{c_{(out)}}})} \right). \end{aligned} \tag{4.12}$$

Equation (4.12) shows that the product of $w_{t_k}^{g_{c_{(in)}}}$ (weight of term t_k in centroid vector $\boldsymbol{w}^{g_{c_{(in)}}}$ of cluster c generated from $G_{i_{(in)}}$) and the reciprocal of $dis(\boldsymbol{w}^{p_{tgt}}, \boldsymbol{w}^{g_{c_{(in)}}})$ (the distance between $\boldsymbol{w}^{p_{tgt}}$ and $\boldsymbol{w}^{g_{c_{(in)}}}$ in the vector space), and similarly the product of element $w_{t_k}^{g_{c_{(out)}}}$ (weight of term t_k in centroid vector $\boldsymbol{w}^{g_{c_{(out)}}}$ of cluster c generated from $G_{i_{(out)}}$) and the reciprocal of $dis(\boldsymbol{w}^{p_{tgt}}, \boldsymbol{w}^{g_{c_{(out)}}})$ (the distance between $\boldsymbol{w}^{p_{tgt}}$ and

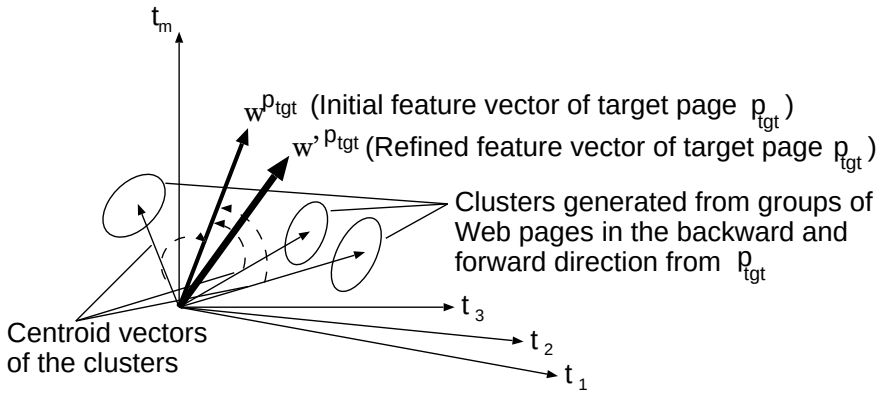
$\boldsymbol{w}^{g_{c_{(out)}}}$ in the vector space) are added to $w_{t_k}^{p_{tgt}}$ (weight of term t_k in p_{tgt} computed by Equation (4.2)), with respect to the number of clusters K. As mentioned in Method I and II, in order to prevent the value of the second and third terms of equation (4.12) from becoming dominant compared with the original term weight $w_{t_k}^{p_{tgt}}$, we introduce Dim, which denotes the number of distinct terms in the Web page collection. We also define $dis(\boldsymbol{w}^{p_{tgt}}, \boldsymbol{w}^{g_{c_{(in)}}})$ and $dis(\boldsymbol{w}^{p_{tgt}}, \boldsymbol{w}^{g_{c_{(out)}}})$ as follows:

$$dis(\boldsymbol{w}^{p_{tgt}}, \boldsymbol{w}^{g_{c_{(in)}}}) = \sqrt{\sum_{k=1}^{m}(w_{t_k}^{p_{tgt}} - w_{t_k}^{g_{c_{(in)}}})^2},$$

$$dis(\boldsymbol{w}^{p_{tgt}}, \boldsymbol{w}^{g_{c_{(out)}}}) = \sqrt{\sum_{k=1}^{m}(w_{t_k}^{p_{tgt}} - w_{t_k}^{g_{c_{(out)}}})^2}.$$



(a)



(b)

Figure 4.3. The refinement of a feature vector as performed by Method III [(a) in the Web space, (b) in the vector space].

4.3 Experiments

4.3.1 Experimental Setup

We conducted experiments in order to verify the retrieval accuracy of our three approaches described in Section 4.2. They were implemented using Perl on a workstation (CPU: UltraSparc-II 480 MHz×4, Memory: 2 GBytes, OS: Solaris 8), and the experiments were conducted using the TREC WT10g test collection [37], which contains about 1.69 million Web pages. Stop words were eliminated from all Web pages in the collection based on the stopword list[3] and stemming was performed using Porter Stemmer[4] [68]. We formulated query vector $\boldsymbol{Q}$ using the terms included in the "title" field in each Topic from 451 to 500 at the TREC WT10g test collection. This query vector $\boldsymbol{Q}$ is denoted as follows:

$$\boldsymbol{Q} = (q_{t_1}, q_{t_2}, \cdots, q_{t_m}),$$

where m is the number of distinct terms in the Web page collection, and t_k $(k = 1, 2, \cdots, m)$ denotes the each term. Each element q_{t_k} of $\boldsymbol{Q}$ is defined as follows:

$$q_{t_k} = \left(0.5 + \frac{0.5 \cdot Qf(t_k)}{\sum_{s=1}^{m} Qf(t_s)}\right) \cdot \log \frac{N_{web}}{df(t_k)}, \quad (k = 1, 2, \cdots, m) \tag{4.13}$$

where $Qf(t_k), N_{web}$, and $df(t_k)$ is the number of index terms t_k, the total number of Web pages in the test collection, and the number of Web pages in which the term t_k appears, respectively. As reported in [77], Equation (4.13) is the element of a query vector that brings the best search result (see Equation (2.1) in Section 2.3.2). We then compute the similarity $sim(\boldsymbol{w}'^{p_{tgt}}, \boldsymbol{Q})$ between refined feature vector $\boldsymbol{w}'^{p_{tgt}}$ and query vector $\boldsymbol{Q}$. The $sim(\boldsymbol{w}'^{p_{tgt}}, \boldsymbol{Q})$ is defined as follows:

$$sim(\boldsymbol{w}'^{p_{tgt}}, \boldsymbol{Q}) = \frac{\boldsymbol{w}'^{p_{tgt}} \cdot \boldsymbol{Q}}{|\boldsymbol{w}'^{p_{tgt}}| \cdot |\boldsymbol{Q}|}. \tag{4.14}$$

In addition, we compute average R-precision $\bar{P}$ based on the following equation:

$$\bar{P} = \frac{1}{N_q} \sum_{n=1}^{N_q} \frac{Rel_{q_n}}{R_{q_n}}, \tag{4.15}$$

[3] ftp://ftp.cs.cornell.edu/pub/smart/english.stop

[4] http://www.tartarus.org/~martin/PorterStemmer/

where q_n is the n^{th} $(n = 1, 2, \cdots, N_q)$ query, R_{q_n} is the number of relevant documents for q_n, and Rel_{q_n} is the number of top R_{q_n} relevant documents that system returns (see Section 2.4.1). With regard to the set of 50 retrieval tasks ($N_q = 50$), we apply Equation (4.15) to evaluate the top 1000 Web pages that our proposed methods return.

In order to verify the effectiveness of the three proposed methods described in Section 4.2, we generated the refined feature vector $\boldsymbol{w}'^{p_{tgt}}$ for initial feature vector $\boldsymbol{w}^{p_{tgt}}$ of target page p_{tgt} with respect to the following cases:

Method I [(Method I–i), (Method I–ii)]

(a) where the contents of all Web pages at levels up to ${L_{(in)}}^{th}$ in the backward direction from p_{tgt} reflect on the initial feature vector $\boldsymbol{w}^{p_{tgt}}$,

(b) where the contents of all Web pages at levels up to ${L_{(out)}}^{th}$ in the forward direction from p_{tgt} reflect on the initial feature vector $\boldsymbol{w}^{p_{tgt}}$,

(c) where the contents of all Web pages at levels both up to ${L_{(in)}}^{th}$ in the backward direction and up to ${L_{(out)}}^{th}$ in the forward direction from p_{tgt} reflect on the initial feature vector $\boldsymbol{w}^{p_{tgt}}$,

Method II [(Method II–i), (Method II–ii)]

(a) where the centroid vectors of clusters generated by the group of Web pages at each level up to ${L_{(in)}}^{th}$ in the backward direction from p_{tgt} reflect on the initial feature vector $\boldsymbol{w}^{p_{tgt}}$,

(b) where the centroid vectors of clusters generated by the group of Web pages at each level up to ${L_{(out)}}^{th}$ in the forward direction from p_{tgt} reflect on the initial feature vector $\boldsymbol{w}^{p_{tgt}}$,

(c) where the centroid vectors of clusters generated by the group of Web pages at each level both up to ${L_{(in)}}^{th}$ in the backward direction and up to ${L_{(out)}}^{th}$ in the forward direction from p_{tgt} reflect on the initial feature vector $\boldsymbol{w}^{p_{tgt}}$,

Method III [(Method III–i), (Method III–ii)]

(a) where the centroid vectors of clusters generated by the group of all Web pages at levels up to ${L_{(in)}}^{th}$ in the backward direction from p_{tgt} reflect on the initial feature vector $\boldsymbol{w}^{p_{tgt}}$,

(b) where the centroid vectors of clusters generated by the group of all Web pages at levels up to ${L_{(out)}}^{th}$ in the forward direction from p_{tgt} reflect on the initial feature vector $\boldsymbol{w}^{p_{tgt}}$,

(c) where the centroid vectors of clusters generated by the group of all Web pages at levels up to ${L_{(in)}}^{th}$ in the backward direction and levels up to ${L_{(out)}}^{th}$ in the forward direction from p_{tgt} reflect on the initial feature vector $\boldsymbol{w}^{p_{tgt}}$.

4.3.2 Experimental Results

Figures 4.4 and 4.5 illustrate the average precision when the values of $L_{(in)}$, $L_{(out)}$, $[L_{(in)}, L_{(out)}]$ vary in Method I–i and ii, respectively. Figures 4.7 to 4.9 and Figures 4.10 to 4.12 illustrate the average precision when the number of clusters K varies in Method II–i and ii, respectively. Moreover, Figures 4.13 to 4.15 and Figures 4.16 to 4.18 illustrate the average precision when the number of clusters K varies in Method III–i and ii, respectively. In order to facilitate the comparison of the average precision obtained by our proposed methods and TF-IDF schme, we also show the retrieval accuracy obtained by TF-IDF scheme in each figure. The precision of TF-IDF scheme does not depend on the values of $L_{(in)}$, $L_{(out)}$, and the number of clusters K. Therefore, the retrieval accuracy obtained by TF-IDF scheme is fixed for those values.

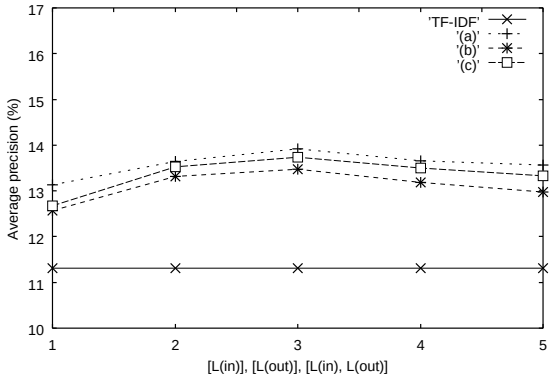


Figure 4.4. Average precision based on Method I–i.

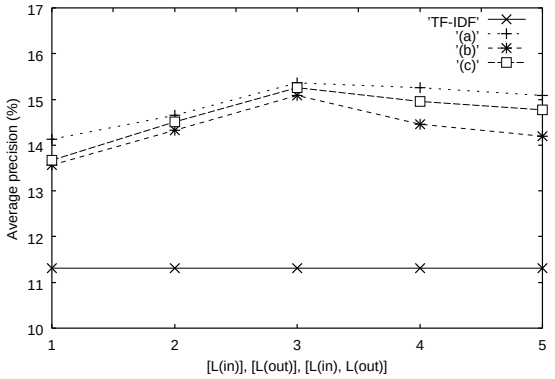


Figure 4.5. Average precision based on Method I–ii.

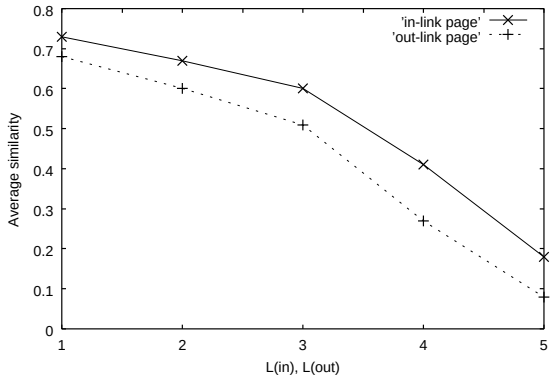


Figure 4.6. Distribution of average similarity.

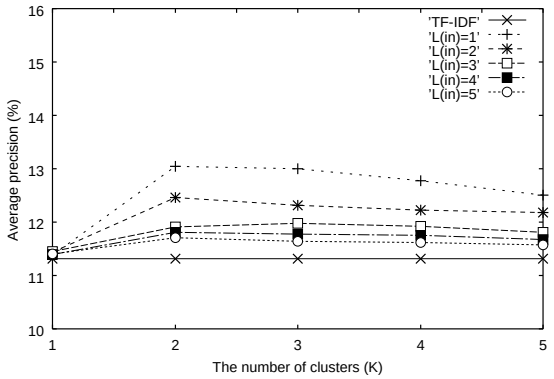


Figure 4.7. Average precision based on Method II–i (a).

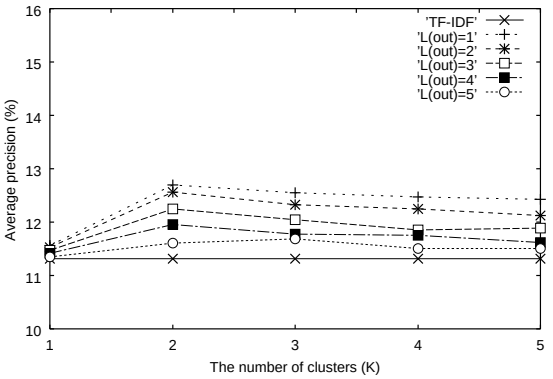


Figure 4.8. Average precision based on Method II–i (b).

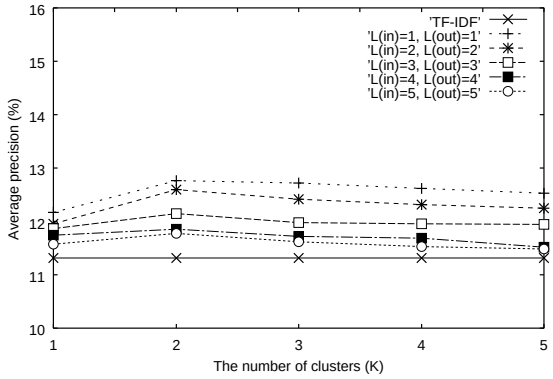


Figure 4.9. Average precision based on Method II–i (c).

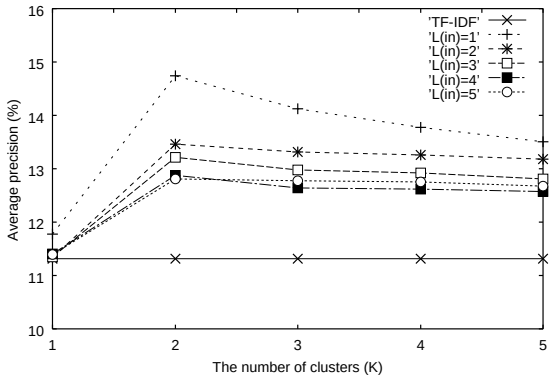


Figure 4.10. Average precision based on Method II–ii (a).

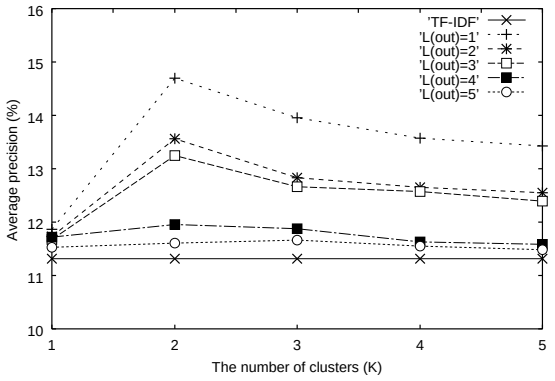


Figure 4.11. Average precision based on Method II–ii (b).

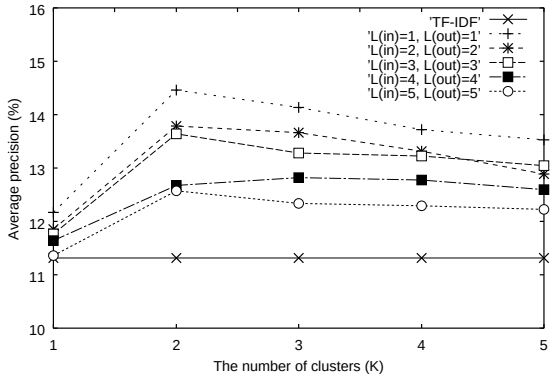


Figure 4.12. Average precision based on Method II–ii (c).

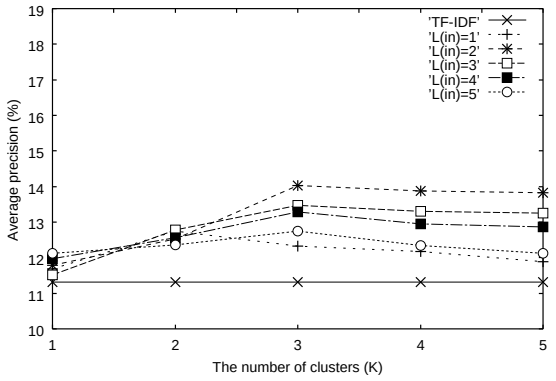


Figure 4.13. Average precision based on Method III–i (a).

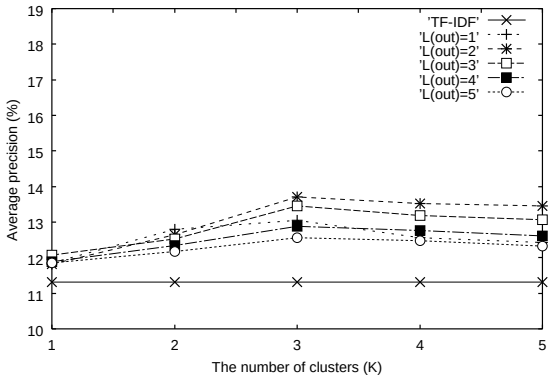


Figure 4.14. Average precision based on Method III–i (b).

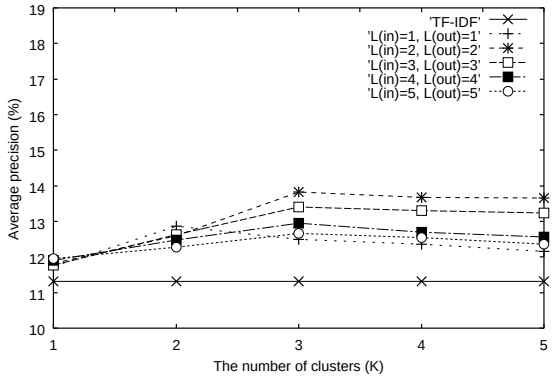


Figure 4.15. Average precision based on Method III–i (c).

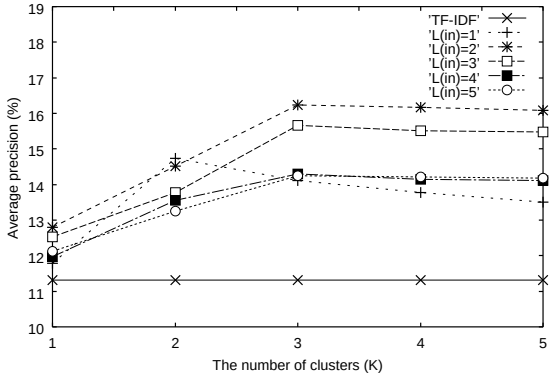


Figure 4.16. Average precision based on Method III–ii (a).

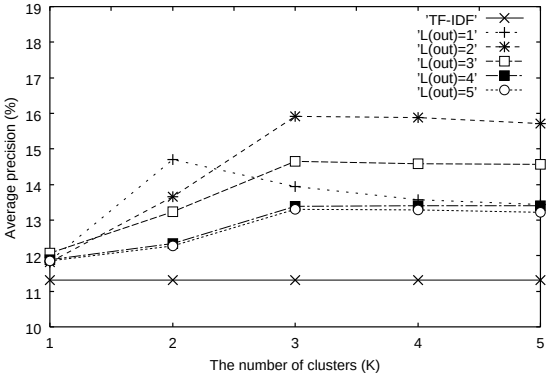


Figure 4.17. Average precision based on Method III–ii (b).

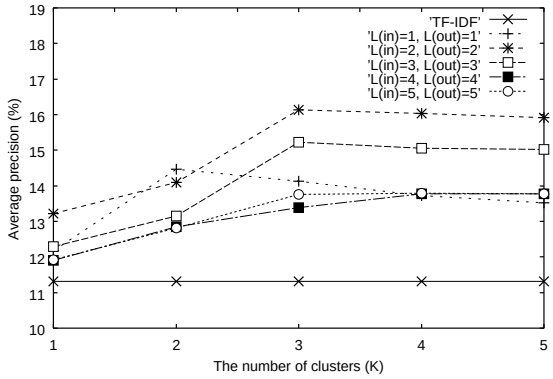


Figure 4.18. Average precision based on Method III-ii (c).

4.3.3 Experiments for Further Improvement

If refined feature vectors of Web pages are iteratively used and each element in the feature vector is updated in each iterative step, it is expected that we can generate further refined feature vectors of Web pages and obtain much better precision. Thus, in this section, we apply this assumption to our proposed method that brought the best results, namely, Method III–ii (a) ($L_{(in)} = 2, K = 3$), and verify the effectiveness of this assumption.

First, we denote the further refined feature vector $\boldsymbol{w}^{p_{tgt}(n)}$ of target page p_{tgt} obtained by n iterations as follows:

$$\boldsymbol{w}^{p_{tgt}(n)} = (w_{t_1}^{p_{tgt}(n)}, w_{t_2}^{p_{tgt}(n)}, \cdots, w_{t_m}^{p_{tgt}(n)}),$$

where m is the number of distinct terms and and t_k $(k = 1, 2, \cdots, m)$ denotes each term. In the iterative algorithm, each element $w_{t_k}^{p_{tgt}(n)}$ of $\boldsymbol{w}^{p_{tgt}(n)}$ is defined as follows:

$$w_{t_k}^{p_{tgt}(n+2)} = w_{t_k}^{p_{tgt}(n+1)} + w_{t_k}^{p_{tgt}(n)}, \tag{4.16}$$

where n is the number of iterations. The two initial weights for Equation (4.16) are as follows:

$w_{t_k}^{p_{tgt}(0)}$: the initial weight given by TF-IDF scheme defined by Equation (4.2),
$w_{t_k}^{p_{tgt}(1)}$: the initial weight given by Equation (4.12).

Based on Equation (4.16), we conducted experiments in order to verify the retrieval accuracy of the above iterative algorithm. Figure 4.19 shows the average precision when the number of iterations varies from 2 to 50.

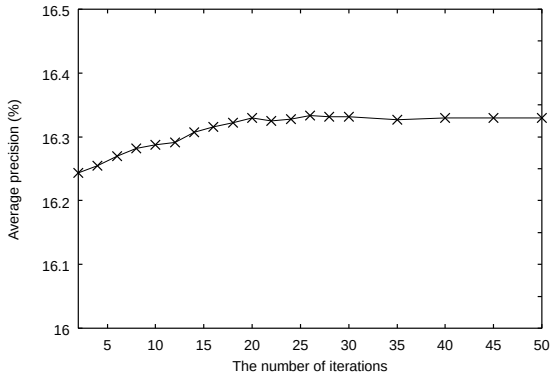


Figure 4.19. Average precision based on iterative algorithm.

4.3.4 Discussion

We can observe the following findings in each method. In Method I, as shown in Figures 4.4 and 4.5, the similarity between the target page p_{tgt} and Web pages at levels up to third ($L_{(in)} \leq 3$) in the backward direction from p_{tgt}, and Web pages at levels up to third ($L_{(out)} \leq 3$) in the forward direction from p_{tgt} is high, and these neighboring Web pages of p_{tgt} contribute for representing the contents of p_{tgt} more accurately. However, the similarity between the target page p_{tgt} and Web pages from forth ($L_{(in)} \geq 4$) level in the backward direction from p_{tgt}, and Web pages from forth ($L_{(out)} \geq 4$) level in the forward direction from p_{tgt} is low, therefore, we cannot observe the effect of representing the contents of p_{tgt} more accurately. Here, in Figure 4.6, we show the distribution of average similarity between Web pages in the WT10g test collection and Web pages in the backward and forward directions from the each Web page in the collection. The similarity between a Web page p in the collection and Web pages at levels up to third ($L_{(in)} \leq 3$) in the backward direction from p, and Web pages at levels up to third ($L_{(in)} \leq 3$) in the forward direction from p is relatively high. However, the average similarity between a Web page p in the collection and Web pages from forth ($L_{(in)} \geq 4$) level in the backward direction from p, and Web pages from forth ($L_{(out)} \geq 4$) level in the forward direction from p is low. We consider that these facts affect the precision. In addition, as Figure 4.4 and 4.5 show, in the case of $L_{(in)} \geq 4$ and $L_{(out)} \geq 4$, the average precision tend to decline. Therefore, it is appropriate that we examined the average precision in the range of $1 \leq L_{(in)} \leq 5$, and $1 \leq L_{(out)} \leq 5$.

In Method II, as shown in Figures 4.7 to 4.12, the farther the distance from target Web page p_{tgt}, in other words, the larger the values of $L_{(in)}$ and $L_{(out)}$ are, the smaller the gap between the graph of average precision obtained by our proposed methods and the graph of average precision obtained by TF-IDF scheme is. In other words, the degree of improvement in retrieval accuracy is small compared with TF-IDF scheme. Thus, we found that with regard to the contents of Web pages, there is strong similarity between the feature vector of target page p_{tgt} and the centroid vector generated by Web page groups at each level up to first in the backward ($L_{(in)} = 1$) and forward ($L_{(out)} = 1$) directions from p_{tgt}. However, we also found that similarity between the feature vector of p_{tgt} and the centroid vector generated by the group of Web pages at each level from p_{tgt} reduces as the value of i, which denotes the length of the shortest

directed path from p_{tgt} to its hyperlinked neighboring pages, increases.

In Method III, as shown in Figures 4.13 to 4.18, the best retrieval accuracy is obtained when we generate refined feature vectors using each centroid vectors of three clusters ($K = 3$) generated in a Web page group produced using all Web pages at levels up to second ($L_{(in)} \leq 2$, $L_{(out)} \leq 2$) in the backward or forward directions from the target page p_{tgt}. Since the best retrieval accuracy is obtained when the number of clusters is three ($K = 3$) in this method, we can infer that the topics of Web pages at levels up to second ($L_{(in)} \leq 2$, $L_{(out)} \leq 2$) in the backward and forward directions from p_{tgt} is usually composed of three topics.

Furthermore, we found the following relations between the number of clusters in Method II and III. Firstly, in Method II, as shown in Figures 4.7 to 4.12, we observed that the average precision tend to decrease in the case of $K \geq 3$. On the other hand, in Method III, we obtain the same results as those of Method II in the case of $L_{(in)} = 1$ and $L_{(out)} = 1$ as shown in Equation (4.8) and (4.12). However, in the case of $L_{(in)} \geq 2$ and $L_{(out)} \geq 2$, we observed that the average precision decrease gradually when the number of clusters K is greater than 4. Therefore, we consider that it is valid that we examined the average precision in the range of $1 \leq K \leq 5$ in Method II and III.

In Method I, II, and III, the approach ii always outperform the approach i. In other words, when we reflect the distance between $\boldsymbol{w}^{p_{tgt}}$ and the feature vector of in- and out-linked pages of p_{tgt} on each element of $\boldsymbol{w}^{p_{tgt}}$, we can obtain better retrieval accuracy. In approach i, the length of the shortest directed path in the backward or forward direction from p_{tgt} to its hyperlinked neighboring pages is reflected on each element of $\boldsymbol{w}^{p_{tgt}}$. This length is the fixed value, namely 1, 2, 3, 4 or 5. However, in approach ii, the distance differs depending on between target Web pages and their hyperlinked Web page, or between target Web page and generated clusters. Therefore, the approach ii can achieve more flexible weight assignment in order to generate refined feature vector.

Table 4.1, which summarizes the results described above, illustrates the average precision when we generated feature vector of Web page using TF-IDF scheme and the best average precision when we generated feature vector of Web page using each of our proposed method. In Method I, the best retrieval accuracy is obtained when we generate the feature vector for a Web page by utilizing the contents of all Web

Table 4.1. Comparison of the best search accuracy obtained using Method I, II, and III.

	% average precision	% improvement
TF-IDF	11.31	–
Method I–i (a) [$L_{(in)} = 3$]	13.92	+2.61
Method I–ii (a) [$L_{(in)} = 3$]	**15.30**	**+3.99**
Method II–i (a) [$L_{(in)} = 1, K = 2$]	13.04	+1.73
Method II–ii (a) [$L_{(in)} = 1, K = 2$]	**14.74**	**+3.43**
Method III–i (a) [$L_{(in)} = 2, K = 3$]	14.03	+2.72
Method III–ii (a) [$L_{(in)} = 2, K = 3$]	**16.23**	**+4.92**

pages at levels up to third ($L_{(in)} = 3$) in the backward direction from the target page p_{tgt}. In Method II, the best retrieval accuracy is obtained when we generate refined feature vectors using each centroid vectors of two clusters ($K = 2$) generated in a Web page group produced using all Web pages at levels up to first ($L_{(in)} = 1$) in the backward direction from the target page p_{tgt}. Moreover, in Mehod III, we obtain the best retrieval accuracy when we generate refined feature vectors using each centroid vectors of three clusters ($K = 3$) generated in a Web page group produced using all Web pages at levels up to second ($L_{(in)} = 2$) in the backward direction from the target page p_{tgt}. Furthermore, as shown in Table 4.1, in any case of Methods I, II, and III, the best retrieval accuracy is obtained in the experiment (a), namely, in the case of using the contents of in-linked pages of a target page. We consider that this finding is obtained because we can easily reach Web pages similar to a target Web page when we follow the hyperlinks in the backward direction from the target page while we reach various Web pages that is not so similar to the contents of the target page when we follow the hyperlinks in the forward direction from the target page. In other words, Web pages have a characteristic that the in-linked pages of a target Web page have Web pages relevant to the contents of the target page. As described in Section 3.1.2, HITS algorithm [50] defines Web pages that have many outgoing links as *hubs*, and also defines the quality of a Web page as *authority* by considering the hubs as its in-linked pages of authority. In addition, focusing on the importance of in-linked pages, the tool that enable to navigate in the backward direction from a Web page is also

developed [17]. We can consider that the results in Table 4.1 suggests the usefulness of in-linked pages.

According to paper [37], in TREC–9 Web Track, the outline of methods used by participants and average precision are shown in Table 4.2. We would like you to refer the papers shown in Table 4.2 about their detailed methods. As shown in Table 4.2, the average precision based on HITS in [52] is 4.88%, and the average precisions based on modified HITS in [23] are 5.91% and 6.37%. These are poor results. Additionally, there are no groups which implemented PageRank algorithm among the participants of TREC–9. Thus, in order to compare our proposed methods with PageRank algorithm, we implemented this algorithm by ourself. As shown in Table 4.2, the average precision based on PageRank is 13.58%. On the other hand, as shown in Tables 4.1 and 4.2, our best average precision is 16.23%. This result is comparable to the result of [20] and [52]. Furthermore, we combine our methods with HITS or PageRank algorithm. In these cases, the average precision is slightly improved as shown in Table 4.2. Considering the computational complexity, these methods are not so effective compared with our pure proposed methods.

In our experiments for further improvement described in Section 4.3.3, we found that the convergence of iterate is relatively rapid and $k = 20$ is sufficient to become stable. The precision obtained after convergence is 16.33%. In other words, we could obtain 0.1% improvement compared with the best precision (16.23% as shown in Table 4.2) among our pure proposed methods. We consider this improvement rate, 0.1% is not so siginificant. Therefore, we believe that our pure proposed methods are effective enough to characterize Web pages more accurately and the obtained results by using them are sufficiently solid.

Table 4.2. Average precision of WT10g using link information.

group	outlines of methods used by group	% average precision
JustSystem [33]	anchor text	20.00
	anchor text + long query	18.38
Waterloo University [20]	content-link	16.31
	4gram content-link	17.94
Twente University [52]	cocitation top 10	16.30
	cocitation top 50	13.37
	HITS	4.88
Neuchatel University [82]	Okapi + probabilistic augmentation	17.36
AT&T [89]	anchor text	12.50
	variant of anchor text	12.88
Johns Hopkins University [62]	back link frequency	10.62
Padova University [23]	modified HITS	5.91
	modified HITS with weighted links	6.37
	PageRank	13.58
our best result	**Method III–ii (a)** [$L_{(in)} = 2$, $K = 3$]	**16.23**
	HITS + our best result	16.27
	PageRank + our best result	16.26
	iterative algorithm based on Method III–ii (a)	**16.33**

4.4 Conclusion of this Chapter

In this chapter, in order to represent the contents of Web pages more accurately, we proposed three approaches to refining the TF-IDF scheme for Web pages using their hyperlinked neighboring pages. Our approach is novel in refining the TF-IDF based feature vector of a target Web page by reflecting the contents of its hyperlinked neighboring pages. Then, we conducted experiments with regard to the following three approaches:

- the approach relies on the contents of all Web pages at levels up to ${L_{(in)}}^{th}$ in the backward direction and levels up to ${L_{(out)}}^{th}$ in the forward direction from the target page p_{tgt},
- the approach relies on the centroid vectors of clusters generated from Web page groups created at each level up to ${L_{(in)}}^{th}$ in the backward direction and each level up to ${L_{(out)}}^{th}$ in the forward direction from the target page p_{tgt},
- the approach relies on the centroid vectors of clusters generated from Web page groups created at levels up to ${L_{(in)}}^{th}$ in the backward direction and levels up to ${L_{(in)}}^{th}$ in the forward direction from the target page p_{tgt},

and evaluated retrieval accuracy of the refined feature vector obtained from each approach using recall-precision curves. Regarding Method I, we can generate a more accurate feature vector for a Web page by utilizing the contents of all Web pages at levels up to at least second in the backward and forward directions from the target page p_{tgt}. In Method II, we found that there is strong similarity between the feature vector of the target page p_{tgt} and the centroid vector generated by the group of Web pages at each level up to first in the backward and forward directions from p_{tgt}. On the other hand, the similarity between p_{tgt} and the centroid vector generated by the group of Web pages at each level from p_{tgt} reduces as the length of the shortest direct paths from p_{tgt} to its hyperlinked neighboring pages increases. With regard to Method III, a more accurate feature vector for a Web page is generated when we use the contents of Web pages at levels up to second in the backward direction from p_{tgt}. Furthermore, compared with the respective best retrieval accuracy obtained using these three approaches, we found that in-linked pages of a target page mainly affect for generating

feature vector that represents the contents of the target page more accurately. Consequently, it is assumed that more accurate feature vectors of Web pages can be generated by assigning higher weight to in-linked pages rather than out-linked pages of a target page. We plan to verify this assumption in future work.

In this chapter, we used the K-means algorithm [60] in order to classify the features of in- and out-linked pages of a target page. However, since we have to set the number of clusters initially in the K-means algorithm, we consider this algorithm to be inappropriate for classifying the features of Web pages that have various link environments. Therefore, in future work, we plan to devise some clustering methods appropriate for various link environments of Web pages. Moreover, in this chapter, we focused on the hyperlink structures of the Web aiming at generating more accurate feature vectors of Web pages. However, in order to satisfy the user's actual information need, it is more important to find relevant Web pages from the enormous Web space. Therefore, we plan to address the technique to provide users with personalized information in the next chapter.

Chapter 5

Web Search Based on User's Preferences

Web search engines help users find useful information on the World Wide Web (WWW). However, when the same query is submitted by different users, typical search engines return the same result regardless of who submitted the query. Generally, each user has different information needs for his/her query. Therefore, the search results should be adapted to users with different information needs. In this chapter, we first propose several approaches to adapting search results according to each user's need for relevant information without any user effort, and then verify the effectiveness of our proposed approaches. Experimental results show that more fine-grained search systems that adapt to each user's preferences can be achieved by constructing user profiles based on modified collaborative filtering.

5.1 Introduction

With the rapid spread of the Internet, anyone can easily access various information by using personal computers, cellular phones, Personal Digital Assistants (PDAs), and such devices. Since information resources on the WWW continue to grow, it has become increasingly difficult for users to find information on the WWW that satisfies their individual needs. Under these circumstances, Web search engines help users find useful information on the WWW. However, when the same query is submitted by different users, most search engines return the same results regardless of who submits

the query. In general, each user has different information needs for his/her query. For example, for the query "Java," some users may be interested in documents dealing with the programming language, "Java," while other users may want documents related to "coffee." Therefore, Web search results should adapt to users with different information needs. In order to predict such information needs, there are several approaches applying data mining techniques to extract usage patterns from Web logs [13, 91, 21, 31, 104]. However, the discovery of patterns from usage data by itself is not sufficient for performing the personalization tasks. Furthermore, Shahabi et al. [86] have pointed out that the item association generated from Web server logs might be wrong because Web usage data from the server side are not reliable. Therefore, these techniques are not so appropriate for Web personalization. Another novel information systems designed to realize such systems have been proposed that personalize information or provide more relevant information for users. As far as we know, three types of Web search systems provide such information: (1) systems using relevance feedback [73], (2) systems in which users register their interest or demographic information, and (3) systems that recommend information based on users' ratings. In these systems, users have to register personal information such as their interests, age, and so on beforehand, or users have to provide feedback on relevant or irrelevant judgements, ratings on a scale from 1 (very bad) to 5 (very good), and so on. These types of registration, feedback, or ratings can become time consuming and users prefer easier methods. Therefore, in this chapter, we propose several approaches that can be used to adapt search results according to each user's information need. We, then, compare the retrieval accuracy of our proposed approaches. Our approach is novel because it allows each user to perform more fine-grained search by capturing changes of each user's preferences without any user effort. Such a method is not performed in typical search engines.

5.2 Proposed Methods

As we described in Section 3.2.1, hyperlink-based personalized search systems have a problem in that they do not clarify whether their search results actually satisfy each user's information need. This is because personalization based on a user's context, i.e., browsing patterns, bookmarks, and so on is not performed. The personalized Web sites described in Section 3.2.2 have the following shortcomings: (1) users have to rate items or adjust sliders to obtain relevant information in "Link Personalization" described in Section 3.2.2(a); (2) in "Content Personalization," described in Section 3.2.2(b), the load on users becomes high because they have to answer questionnaires in advance to register their personal preferences or demographic information, and they have to change their registered information by themselves if their interests change; (3) in "Design-Oriented Personalization" described in Section 3.2.2(c), dynamic changes in user's preferences are not addressed. In addition, the recommender systems described in Section 3.2.3, have the potential to provide serendipitous recommendations if users are only willing to rate items. However, in practice, most users are unwilling to rate items even though user's ratings for items are key factors to achieving better recommendations. As a result, the accuracy of recommendations may be poor. Furthermore, this is also true in personalized multimedia systems described in Section 3.2.4, and especially in collaborative filtering-based personalized multimedia systems, although the time it takes to compute recommended information for users is not much longer since multimedia information with a large amount of data does not need to be analyzed.

We do not necessarily believe that approaches based on user ratings provide users with more relevant information that satisfies each user's information need. Therefore, search system should directly and exactly capture the changes in each user's preferences without any user effort in order to provide more relevant information for each user. In order to construct such systems, we propose several approaches to adapting search results according to each user's information need. Unlike the research studies described in Section 3.2, our approach is novel because it allows each user to perform a fine-grained search by capturing the changes in each user's preferences without any user effort.

Figure 5.1 shows an overview of our system. When a user submits a query to a search engine through a Web browser, the search engine returns search results corre-

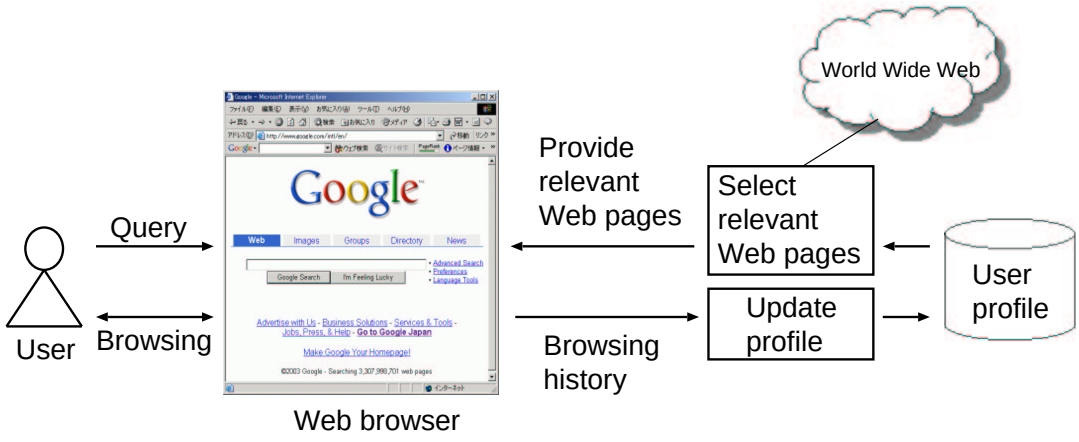


Figure 5.1. System overview.

sponding to the query. Based on the search results, the user may select a Web page in an attempt to satisfy his/her information need. In addition, the user may access more Web pages by following the hyperlinks on his/her selected Web page and continue to browse. Our system monitors the user's browsing history and updates his/her profile whenever his/her browsing page changes. When the user submits a query the next time, the search results adapt based on his/her user profile.

In the following sections, we explain how to construct a user profile in the update profile component illustrated in Figure 5.1. In our approach, the user profile is constructed implicitly. In other words, a user does not need to perform explicit efforts such as feedback, ratings and so on in order to construct his/her profile. We construct each user profile based on the following two methods: (1) Pure browsing history, and (2) Modified collaborative filtering.

5.2.1 User Profile Construction Based on Pure Browsing History

In this method, we assume that the preferences of each user consist of the following two aspects: (1) persistent (or long term) preferences, and (2) ephemeral (or short term) preferences. In persistent preferences, the user profile is incrementally developed over time and it is stored for use in later sessions. The information exploited for constructing the profile usually comes from various sources, so it relies on different aspects of the user. On the other hand, in ephemeral preferences, the information used to construct each user profile is only gathered during the current session, and it is immediately exploited for executing some adaptive process aimed at personalizing the current interaction. From these two factors, we construct each user profile $\boldsymbol{P}$ considering both persistent preferences, $\boldsymbol{P}^{per}$, and ephemeral preferences, $\boldsymbol{P}^{today}$. $\boldsymbol{P}^{per}$ shows a user profile constructed exploiting the user's browsing history of Web page from N days ago (see Figure 5.2). Here, we introduce the concept of window size in order to construct $\boldsymbol{P}^{per}$, and define S_j $(j = 0, 1, 2, \cdots, N)$ as the number of Web pages the user browsed on the j^{th} day. "$j = 0$" means "today" as shown in Figure 5.2. In each day, $\boldsymbol{P}^{today}$ is constructed through the following process. First, we denote the feature vector $\boldsymbol{w}^{hp}$ of browsed Web page hp $(hp = 1, 2, \cdots, S_0)$ as follows:

$$\boldsymbol{w}^{hp} = (w_{t_1}^{hp}, w_{t_2}^{hp}, \cdots, w_{t_m}^{hp}),$$

where m is the number of distinct terms in the Web page hp, and t_k $(k = 1, 2, \cdots, m)$ denotes each term. Using the TF (term frequency) scheme, we also define each element $w_{t_k}^{hp}$ of $\boldsymbol{w}^{hp}$ as follows:

$$w_{t_k}^{hp} = c^{hp} \cdot \frac{tf(t_k, hp)}{\sum_{s=1}^{m} tf(t_s, hp)}, \tag{5.1}$$

where $tf(t_k, hp)$ is the frequency of term t_k in each browsed Web page hp, and c^{hp} is a constant that shows to what extent our system reflects the contents of the Web page on each user profile. We define constant c^{hp} as follows:

$$c^{hp} = \begin{cases} 1; & dr \geq Th, \\ 0; & dr < Th, \end{cases}$$

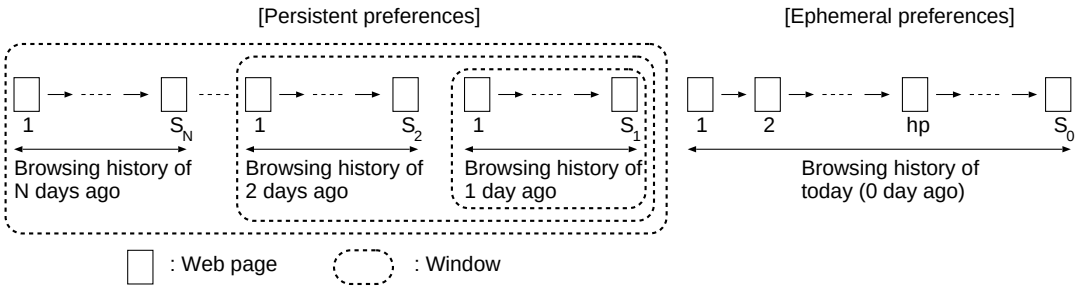


Figure 5.2. Window size for constructing persistent user profile.

where dr denotes the time spent reading normalized by the number of terms in Web page hp. We define threshold Th as 0.317 based on preliminary experiments. We then denote user profile $\boldsymbol{P}^{today}$ as follows:

$$\boldsymbol{P}^{today} = (p_{t_1}^{today}, p_{t_2}^{today}, \cdots, p_{t_m}^{today}),$$

and define each element $p_{t_k}^{today}$ as follows:

$$p_{t_k}^{today} = \frac{1}{S_0} \sum_{hp=1}^{S_0} w_{t_k}^{hp},$$

As described above, $\boldsymbol{P}^{today}$ shows a user profile constructed using the user's browsing history of today's Web page. Moreover, we set the window size N $(N = 1, 2, \cdots, 30)$ to construct $\boldsymbol{P}^{per}$. We also denote $\boldsymbol{P}^{per}$ as follows:

$$\boldsymbol{P}^{per} = (p_{t_1}^{per}, p_{t_2}^{per}, \cdots, p_{t_m}^{per}),$$

and define each element $p_{t_k}^{per}$ as follows:

$$p_{t_k}^{per} = \frac{1}{S_N} \sum_{hp=1}^{S_N} w_{t_k}^{hp} \cdot e^{-\frac{\log 2}{hl}(d - d_{t_k\_init})}, \tag{5.2}$$

where $e^{-\frac{\log 2}{hl}(d - d_{t_k\_init})}$ is a forgetting factor under the assumption that user's preferences gradually decay as days pass. In this factor, $d_{t_k\_init}$ is the day when term t_k initially occur, d is the number of days following to $d_{t_k\_init}$, and hl is a half-life span parameter. We set the half-life span hl to 7. In other words, the intuition behind this

assumption is that user's preferences reduce by 1/2 in one week. Let us assume that each user browsed S_N pages on each day. Of course, the number of browsed Web pages S_N, differs user by user. Therefore, we normalize $p_{t_k}^{per}$ using S_N as shown in Equation (5.2). Using these parameters, we finally construct user profile $\boldsymbol{P}$ as defined in the following equation:

$$\boldsymbol{P} = a\boldsymbol{P}^{per} + b\boldsymbol{P}^{today}, \tag{5.3}$$

where a and b are constants that satisfy $a + b = 1$.

5.2.2 User Profile Construction Based on Modified Collaborative Filtering Algorithm

In this section, we first briefly review the pure collaborative filtering algorithm, especially neighborhood-based algorithms, and then describe how to construct user profiles using the modified collaborative filtering algorithms.

(a) Overview of the Pure Collaborative Filtering Algorithm

Collaborative filtering can be represented as the problem of predicting missing values in a user-item ratings matrix. Figure 5.3 shows a simplified example of a user-item ratings matrix.

In the neighborhood-based algorithm [38], a subset of users is first chosen based on their similarity to the active user, and a weighted combination of their rating is then used to produce predictions for the active user. The algorithm we use can be summarized in the following steps:

1. Weight all users with respect to similarity to the active user. This similarity between users is measured as the Pearson correlation coefficient between their rating vectors.

2. Select n users that have the highest similarity with the active user. These users form the *neighborhood*.

3. Compute a prediction from a weighted combination of the neighbor's ratings.

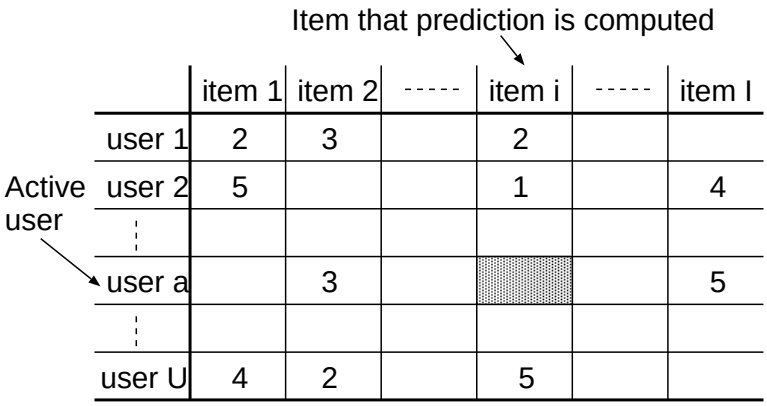


Figure 5.3. User-item ratings matrix for collaborative filtering.

In step 1, $S_{a,u}$, which denotes similarity between users a and u, is computed using the Pearson correlation coefficient defined below:

$$S_{a,u} = \frac{\sum_{i=1}^{I}(r_{a,i} - \bar{r}_a) \times (r_{u,i} - \bar{r}_u)}{\sqrt{\sum_{i=1}^{I}(r_{a,i} - \bar{r}_a)^2 \times \sum_{i=1}^{I}(r_{u,i} - \bar{r}_u)^2}}, \tag{5.4}$$

where $r_{a,i}$ is the rating given to item i by user a, $\bar{r}_a$ is the mean rating given by user a, and I is the total number of items.

In step 2, i.e., neighborhood-based methods, a subset of appropriate users is chosen based on their similarity to the active user, and a weighted aggregate of their ratings is used to generate predictions for the active user in the next step 3.

In step 3, predictions are computed as the weighted average of deviations from the neighbor's mean:

$$p_{a,i} = \bar{r}_a + \frac{\sum_{u=1}^{n}(r_{u,i} - \bar{r}_u) \times S_{a,u}}{\sum_{u=1}^{n} S_{a,u}},$$

where $p_{a,i}$ is the prediction for active user a for item i, $S_{a,u}$ is the similarity between users a and u as described at Equation (5.4), and n is the number of users in the neighborhood.

(b) User Profile Construction using Modified Collaborative Filtering Algorithm

In the pure collaborative filtering algorithms described in Section 5.2.2(a), we considered a user-item ratings matrix. Similarly, in the construction of a user profile, we can consider a user-term weights matrix like that shown in Figure 5.4(a). In addition, based on the pure collaborative filtering algorithms, we can apply their predictive algorithms to predict a term weight in each user profile. In other words, since each user profile is computed based on term weights in a Web page the user browsed and the browsed pages are different according to each user, the user profile is constructed in the form of a user-term weights matrix with missing values, as illustrated in Figure 5.4. This is very analogous to the user-item ratings matrix used in the pure collaborative filtering algorithms. Therefore, we expect that a more accurate user profile is constructed since these missing values are predicted using the algorithms in collaborative filtering. In this approach, we propose the following two methods: (1) user profile construction based on the static number of users in the neighborhood, and (2) user profile construction based on dynamic number of users in the neighborhood.

(b-1) User Profile Construction Based on the Static Number of Users in the Neighborhood

In this method, our proposed algorithms are explained in the following steps (note the similarity to the collaborative filtering algorithms described in Section 5.2.2(a)):

1. Weight all users with respect to similarity to the active user. This similarity between users is measured as the Pearson correlation coefficient between their term weight vectors unlike the rating vectors described in Section 5.2.2(a).

2. Select n users that have the highest similarity to the active user. These users form the neighborhood.

3. Compute a prediction from a weighted combination of the neighbor's term weights.

In step 1, $S_{a,u}$, which denotes similarity between users a and u, is computed using the Pearson correlation coefficient, defined below:

$$S_{a,u} = \frac{\sum_{i=1}^{T}(w_{a,i} - \bar{w}_a) \times (w_{u,i} - \bar{w}_u)}{\sqrt{\sum_{i=1}^{T}(w_{a,i} - \bar{w}_a)^2 \times \sum_{i=1}^{T}(w_{u,i} - \bar{w}_u)^2}}, \tag{5.5}$$

Term weight that prediction is computed

	term 1	term 2	-----	term i	-----	term T
user 1	0.745	0.362				0.718
Active user 2		0.835		0.534		0.126
⋮						
user a		0.639				0.485
⋮						
user U	0.247	0.461		0.928		

(a)

Term weight that prediction is computed

	term 1	term 2	-----	term i	-----	term T	term T+1	term T+2	-----	term T+v
user 1	0.745	0.362				0.718		0.451		
Active user 2		0.835		0.534		0.126	0.723			
⋮										
user a		0.639				0.485		0.328		0.563
⋮										
user U	0.247	0.461		0.928			0.686			0.172

(b)

Figure 5.4. User-term weights matrix for modified collaborative filtering [(a) when each user browsed k Web pages, (b) when each user browsed $k + 1$ Web pages].

where $w_{a,i}$ is the weight of term i regarding user a computed based on term frequency in a browsed Web page defined by Equation (5.1), and $\bar{w}_a$ is the mean term weight regarding user a, and T is the total number of terms.

In step 2, i.e., neighborhood-based methods, a subset of appropriate users is chosen based on their similarity to the active user, and a weighted aggregate of their term weights is used to generate predictions for the active user in the coming step 3. In this step, the number of selected users is fixed to n for any user. That is why we call this method "static."

In step 3, predictions are computed as the weighted average of deviations from the neighbor's mean:

$$p_{a,i} = \bar{w}_a + \frac{\sum_{u=1}^{n}(w_{u,i} - \bar{w}_u) \times S_{a,u}}{\sum_{u=1}^{n} S_{a,u}},$$

where $p_{a,i}$ is the prediction for the active user a for weight of term i, $S_{a,u}$ is the similarity between users a and u as described at Equation (5.5), and n is the number of users in the neighborhood.

(b-2) User Profile Construction Based on Dynamic Number of Users in the Neighborhood

In this method, our proposed algorithms are explained in the following steps (note the similarity to the collaborative filtering algorithms described in Section 5.2.2(a), and the aforementioned static approach):

1. Generate clusters of users by means of the k-Nearest Neighbor clustering algorithms [45]. The similarity between user a and these clusters is measured as the Pearson correlation coefficient between their term weight vectors.

2. Select n clusters that have higher similarity to the active user than the threshold. We consider the centroid vectors of these selected clusters as the neighborhood of the active user.

3. Compute a prediction from a weighted combination of the term weights using centroid vectors of clusters.

In step 1, $S_{a,g}$, which denotes similarity between users a and centroid vectors of clusters g, is computed using the Pearson correlation coefficient, defined below:

$$S_{a,g} = \frac{\sum_{i=1}^{T}(w_{a,i} - \bar{w}_a) \times (w_{g,i} - \bar{w}_g)}{\sqrt{\sum_{i=1}^{T}(w_{a,i} - \bar{w}_a)^2 \times \sum_{i=1}^{T}(w_{g,i} - \bar{w}_g)^2}}, \tag{5.6}$$

where $w_{a,i}$ is the weight of of term i regarding user a computed based on term frequency in a browsed Web page defined by Equation (5.1), and $\bar{w}_a$ is the mean term weight regarding user a, and T is the total number of terms.

In step 2, several clusters are chosen based on their similarity to the active user, and a weighted aggregate of their term weights is used to generate predictions for the active user in the next step 3. In this step, the number of selected clusters is different user by user. That is why we call this method "dynamic." Therefore, it is expected that this method allows each user to perform more fine-grained search.

In step 3, predictions are computed as the weighted average of deviations from the neighbor's mean:

$$p_{a,i} = \bar{w}_a + \frac{\sum_{g=1}^{n}(w_{g,i} - \bar{w}_g) \times S_{a,g}}{\sum_{g=1}^{n} S_{a,g}},$$

where $p_{a,i}$ is the prediction for the active user a for term weights i, $S_{a,g}$ is the similarity between users a and centroid vectors of clusters g as described at Equation (5.6), and n is the number of centroid vectors of clusters in the neighborhood.

5.3 Experiments

5.3.1 Experimental Setup

We conducted experiments in order to verify the effectiveness of the three approaches: (1) relevance feedback and implicit approaches, (2) user profiles based on pure browsing history as described in Section 5.2.1, and (3) user profiles based on the modified collaborative filtering algorithm described in Section 5.2.2. While users have to provide feedback explicitly in relevance feedback, users do not have to provide any effort in our proposed methods (2) and (3) since our system implicitly captures changes in user's preference. The experiments were implemented using Perl on a workstation (CPU: UltraSparc-II 480MHz$\times$ 4, Memory: 2GBytes, OS: Solaris8) using 50 query

topics that were employed as test topics in the TREC WT10g test collection [37]. Note that we only used query topics of the test collection, and did not use the contents of the test collection. In our experiments, we observed the browsing history of 20 subjects for 30 days. The subjects browsed 12 Web pages in one day on average. In addition, the number of terms in user profiles accumulated during the 30 days is about 810,000. In the following, let the h^{th} Web page in the search results and the user profile as defined by Equation (5.3) be rp_h and $\boldsymbol{P}$, respectively. Then, the feature vector of the h^{th} Web page rp_h in the search results, $\boldsymbol{w}^{rp_h}$, is defined as follows:

$$\boldsymbol{w}^{rp_h} = (w_{t_1}^{rp_h}, w_{t_2}^{rp_h}, \cdots, w_{t_m}^{rp_h}),$$

where m is the number of distinct terms in the Web page rp_h, and t_k $(k = 1, 2, \cdots, m)$ denotes each term. We also define each element $w_{t_k}^{rp_h}$ of $\boldsymbol{w}^{rp_h}$ based on the TF (term frequency) scheme as follows:

$$w_{t_k}^{rp_h} = \frac{tf(t_k, rp_h)}{\sum_{s=1}^{m} tf(t_s, rp_h)},$$

where $tf(t_k, rp_h)$ is the frequency of term t_k in the rp_h. The similarity $sim(\boldsymbol{P}, \boldsymbol{w}^{rp_h})$ between the user profile $\boldsymbol{P}$ and the feature vector of the h^{th} Web page in search results $\boldsymbol{w}^{rp_h}$ is computed by the following equation. The $sim(\boldsymbol{P}, \boldsymbol{w}^{rp_h})$ is defined as follows:

$$sim(\boldsymbol{P}, \boldsymbol{w}^{rp_h}) = \frac{\boldsymbol{P} \cdot \boldsymbol{w}^{rp_h}}{|\boldsymbol{P}| \cdot |\boldsymbol{w}^{rp_h}|}. \tag{5.7}$$

Based on the value obtained by Equation (5.7), the search results are adapted to each user according to his/her profile. These results were compared with the search results of Google [11]. We then evaluate the retrieval accuracy using R-precision [3]. We employed 30 as the value of R.

5.3.2 Experimental Results

(a) User Profile Based on Relevance Feedback

Relevance feedback [73] is the most popular query reformulation strategy. In a relevance feedback process, the user is presented with a list of the retrieved documents and marks those that are relevant after examining them. The basic idea is to reformulate the original query vector $\boldsymbol{Q}^{org}$ into new query vector $\boldsymbol{Q}^{new}$ such that it gets closer to the term-weight vector space of the relevant documents. In our experiment, we use the Rocchio formulation defined as follows:

$$\boldsymbol{Q}^{new} = \alpha \boldsymbol{Q}^{org} + \frac{\beta}{|D_r|} \sum_{\boldsymbol{d}_j \in D_r} \boldsymbol{d}_j - \frac{\gamma}{|D_n|} \sum_{\boldsymbol{d}_j \in D_n} \boldsymbol{d}_j,$$

where D_r and D_n are the set of relevant and non-relevent documents as identified by the user among the retrieved documents, respectively, and $|D_r|$ and $|D_n|$ are the number of documents in the sets D_r and D_n, respectively (see Section 2.5.1). We set α, β and γ that are tuning constants to 1, 1 and 0, respectively.

We believe that the new query vector $\boldsymbol{Q}^{new}$ obtained by the user's judgement, whether the retrieved documents are relevant or not, reflects the user's preferences. Therefore, we treat $\boldsymbol{Q}^{new}$ as $\boldsymbol{P}^{today}$ defined by Equation (5.3), and employ $\boldsymbol{Q}^{new}$ as an initial preference of a user to construct a user profile. In this case, using Equation (5.3), the user profile $\boldsymbol{P}$ is defined as follows:

$$\boldsymbol{P} = a\boldsymbol{P}^{per} + b\boldsymbol{Q}^{new}, \tag{5.8}$$

We asked each subject to judge if the search results returned by the search engine are relevant, and constructed user profile $\boldsymbol{P}$ based on Equation (5.8). In this experiment, we varied the number of feedbacks FB that each user provided from 1 to 3. Figures 5.5 to 5.7 show the R-precision when the values of a and b are varied such that these values satisfy $a + b = 1$ under the condition that the numbers of feedbacks are 1, 2, and 3.

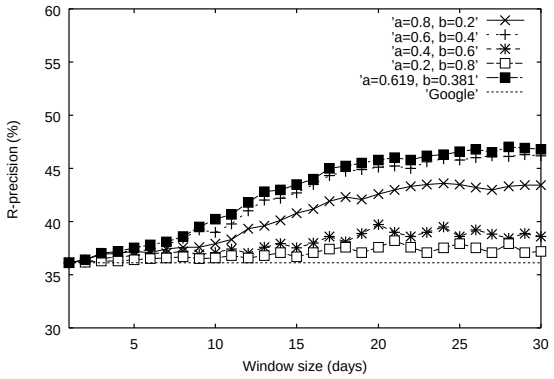


Figure 5.5. R-precision obtained using relevance feedback-based user profile (FB=1).

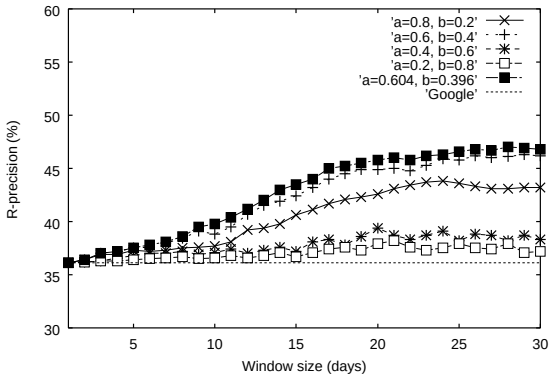


Figure 5.6. R-precision obtained using relevance feedback-based user profile (FB=2).

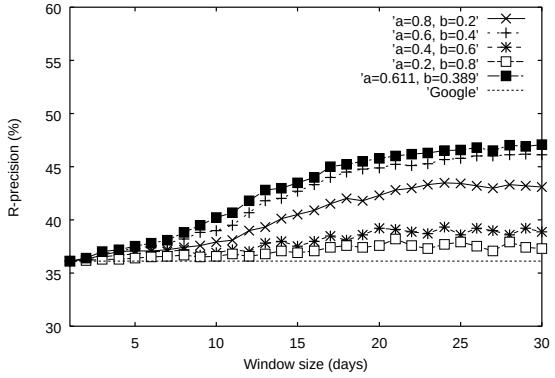


Figure 5.7. R-precision obtained using relevance feedback-based user profile (FB=3).

(b) User Profile Based on Pure Browsing History

In this approach, each user profile is constructed as mentioned in Section 5.2.1. The user profile $\boldsymbol{P}$ is defined as follows:

$$\boldsymbol{P} = a\boldsymbol{P}^{per} + b\boldsymbol{P}^{today}.$$

Figure 5.8 shows the R-precision when the values of a and b are varied such that these values satisfy $a + b = 1$.

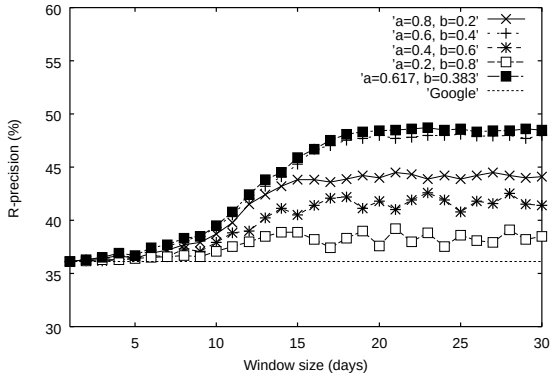


Figure 5.8. R-precision obtained using pure browsing history-based user profile.

(c) User Profile Based on Modified Collaborative Filtering

In this approach, when the user browses a new Web page, new terms are added to his/her user profile. However, other users do not always browse the same pages, so missing values occur in the user-term weights matrix as illustrated in Figure 5.4. These missing values are predicted using the algorithms described in Section 5.2.2, and then the matrix is filled. We consider that this user-term vector reflects the user's preferences. Let this user-term vector with predicted value be $\boldsymbol{V}^{pre}$. We treat $\boldsymbol{V}^{pre}$ as $\boldsymbol{P}^{today}$ defined by Equation (5.3), and employ $\boldsymbol{V}^{pre}$ as an initial preference of a user to construct a user profile. In this case, using Equation (5.3), the user profile $\boldsymbol{P}$ is defined as follows:

$$\boldsymbol{P} = a\boldsymbol{P}^{per} + b\boldsymbol{V}^{pre}, \tag{5.9}$$

Figures 5.9 to 5.12 show the R-precision of static approaches when the values of a and b are varied such that these values satisfy $a + b = 1$ under the condition that the numbers of neighbors n are 5, 10, 15, and 20, respectively. In addition, Figure 5.13 shows the R-precision of dynamic approaches.

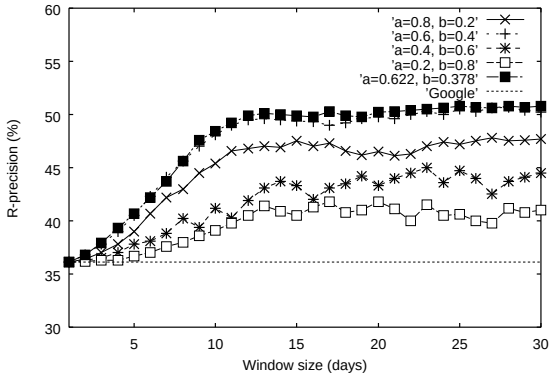


Figure 5.9. R-precision obtained using modified collaborative filtering-based user profile (static, $n = 5$).

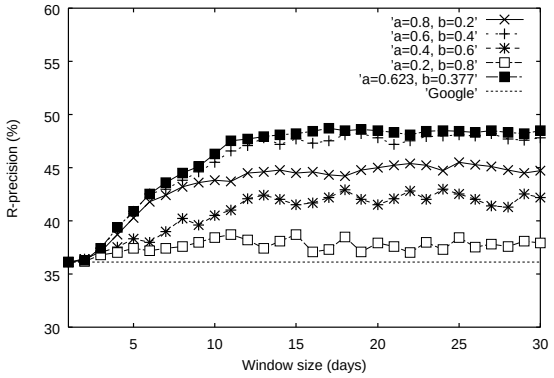


Figure 5.10. R-precision obtained using modified collaborative filtering-based user profile (static, $n = 10$).

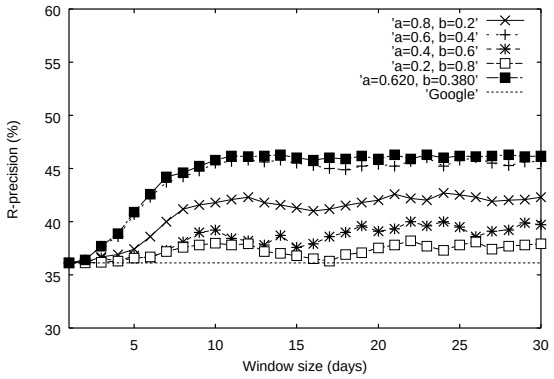


Figure 5.11. R-precision obtained using modified collaborative filtering-based user profile (static, $n = 15$).

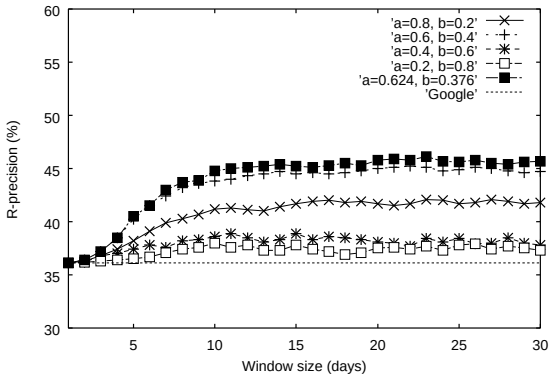


Figure 5.12. R-precision obtained using modified collaborative filtering-based user profile (static, $n = 20$).

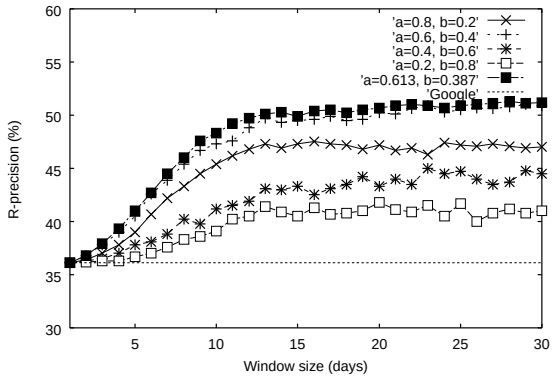


Figure 5.13. R-precision obtained using modified collaborative filtering-based user profile (dynamic).

5.3.3 Experiments for Further Improvement

In Section 5.2.1, user's searches and browsing activities fell into one logical session. However, users might do different task in one day and may well do several searches and browsing activities in that time period. Therefore, we need to analyze user's browsing behavior in more detail. Figure 5.14 illustrates the detaild user's browsing history in today and N days before today. In this figure, we consider that users perform n_{bh} different searches before the current session cur in today. In other words, the cur^{th} session, that is the newest session in today, is subsequent to the ${n_{bh}}^{th}$ session. Therefore, the relation between n_{bh} and cur is defined by the following equation:

$$cur = n_{bh} + 1.$$

In each day, $\boldsymbol{P}^{today}$ is constructed through the following process. At first, we denote the feature vector $\boldsymbol{w}^{hp^{(r)}}$ of browsed Web page $hp^{(r)}$ $(hp = 1, 2, \cdots, S_0)$ in the r^{th} $(r = 1, 2, \cdots, n_{bh})$ session as follows:

$$\boldsymbol{w}^{hp^{(r)}} = (w_{t_1}^{hp^{(r)}}, w_{t_2}^{hp^{(r)}}, \cdots, w_{t_m}^{hp^{(r)}}),$$

where m is the number of distinct terms in the Web page $hp^{(r)}$, and t_k $(k = 1, 2, \cdots, m)$ denotes each term. Using the TF (term frequency) scheme, each element $w_{t_k}^{hp^{(r)}}$ of $\boldsymbol{w}^{hp^{(r)}}$ is defined as follows:

$$w_{t_k}^{hp^{(r)}} = c^{hp^{(r)}} \cdot \frac{tf(t_k, hp^{(r)})}{\sum_{s=1}^{m} tf(t_s, hp^{(r)})}, \tag{5.10}$$

where $tf(t_k, hp^{(r)})$ is the frequency of term t_k in each browsed Web page $hp^{(r)}$, and $c^{hp^{(r)}}$ is a constant that shows to what extent our system reflect the contents of the Web page on each user profile. As well as Equation (5.2), we define constant $c^{hp^{(r)}}$ as follows:

$$c^{hp^{(r)}} = \begin{cases} 1; & dr \geq Th, \\ 0; & dr < Th, \end{cases}$$

where dr denotes the time spent reading normalized by the number of terms in Web page $hp^{(r)}$, and threshold Th is set to 0.317 based on our preliminary experiments. We then define partial user profile $\boldsymbol{P}^{(r)}$ at the r^{th} browsing history in today as follows:

$$\boldsymbol{P}^{(r)} = (p_{t_1}^{(r)}, p_{t_2}^{(r)}, \cdots, p_{t_m}^{(r)}),$$

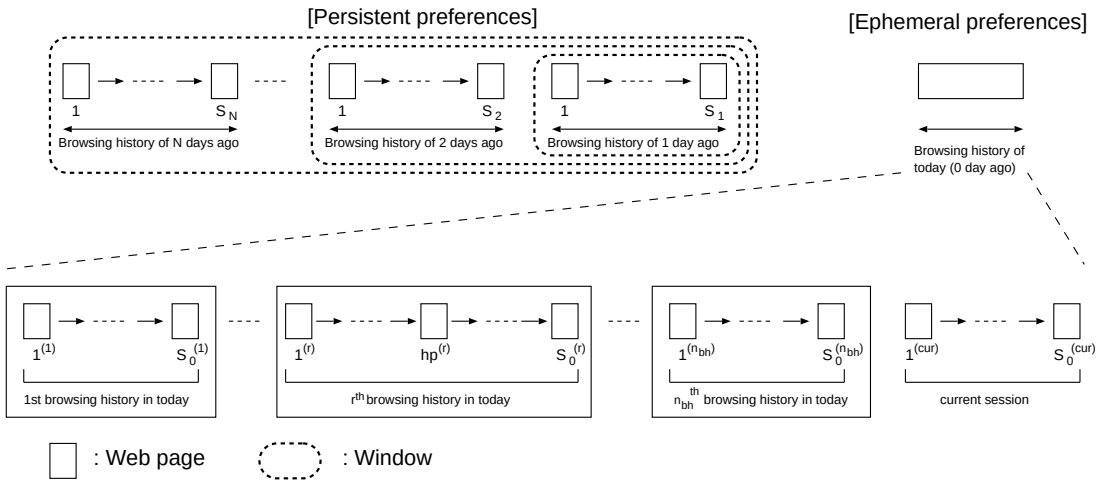


Figure 5.14. Detailed user's browsing history in today and N days before today.

and define each element $p_{t_k}^{(r)}$ using Equation (5.10) as follows:

$$\begin{aligned} p_{t_k}^{(r)} &= \frac{1}{{S_0}^{(r)}} \sum_{hp=1}^{S_0} w_{t_k}^{hp^{(r)}} \\ &= \frac{1}{{S_0}^{(r)}} \sum_{hp=1}^{S_0} c^{hp^{(r)}} \cdot \frac{tf(t_k, hp^{(r)})}{\sum_{s=1}^{m} tf(t_s, hp^{(r)})}. \end{aligned} \tag{5.11}$$

We then define user profile $\boldsymbol{P}^{(br)}$ obtained by browsing history up to the current session as follows:

$$\boldsymbol{P}^{(br)} = (p_{t_1}^{(br)}, p_{t_2}^{(br)}, \cdots, p_{t_m}^{(br)}).$$

Using Equation (5.11), each element $p_{t_k}^{(br)}$ is also defined as follows:

$$\begin{aligned} p_{t_k}^{(br)} &= \sum_{r=1}^{n_{bh}} p_{t_k}^{(r)} \\ &= \sum_{r=1}^{n_{bh}} \frac{1}{{S_0}^{(r)}} \sum_{hp=1}^{S_0} c^{hp^{(r)}} \cdot \frac{tf(t_k, hp^{(r)})}{\sum_{s=1}^{m} tf(t_s, hp^{(r)})}. \end{aligned}$$

Similarly, we denote the feature vector $\boldsymbol{w}^{hp^{(cur)}}$ of browsed Web page $hp^{(cur)}$ $(hp = 1, 2, \cdots, S_0)$ in the current session as follows:

$$\boldsymbol{w}^{hp^{(cur)}} = (w_{t_1}^{hp^{(cur)}}, w_{t_2}^{hp^{(cur)}}, \cdots, w_{t_m}^{hp^{(cur)}}),$$

where m is the number of distinct terms in the Web page $hp^{(cur)}$, and t_k $(k = 1, 2, \cdots, m)$ denotes each term. Using the TF (term frequency) scheme, each element $w_{t_k}^{hp^{(cur)}}$ of $\boldsymbol{w}^{hp^{(cur)}}$ is defined as follows:

$$w_{t_k}^{hp^{(cur)}} = c^{hp^{(cur)}} \cdot \frac{tf(t_k, hp^{(cur)})}{\sum_{s=1}^{m} tf(t_s, hp^{(cur)})}. \tag{5.12}$$

where $tf(t_k, hp^{(cur)})$ is the frequency of term t_k in each browsed Web page $hp^{(cur)}$, and $c^{hp^{(cur)}}$ is a constant that shows to what extent our system reflects the contents of the Web page on each user profile defined as well as Equation (5.11). Then, we define partial user profile $\boldsymbol{P}^{(cur)}$ obtained at the current session in today as follows:

$$\boldsymbol{P}^{(cur)} = (p_{t_1}^{(cur)}, p_{t_2}^{(cur)}, \cdots, p_{t_m}^{(cur)}),$$

and define each element $p_{t_k}^{(cur)}$ using Equation (5.12) as follows:

$$\begin{aligned} p_{t_k}^{(cur)} &= \frac{1}{{S_0}^{(cur)}} \sum_{hp=1}^{S_0} w_{t_k}^{hp^{(cur)}} \\ &= \frac{1}{{S_0}^{(cur)}} \sum_{hp=1}^{S_0} c^{hp^{(cur)}} \cdot \frac{tf(t_k, hp^{(cur)})}{\sum_{s=1}^{m} tf(t_s, hp^{(cur)})}. \end{aligned}$$

Using $\boldsymbol{P}^{(br)}$ and $\boldsymbol{P}^{(cur)}$, $\boldsymbol{P}^{today}$ is constructed as follows:

$$\boldsymbol{P}^{today} = x\boldsymbol{P}^{(br)} + y\boldsymbol{P}^{(cur)}, \tag{5.13}$$

where x and y are constants that satisfy $x + y = 1$. In order to emphasize the current session, we assign larger weight to y than x. In other words, y is larger than 0.5, and x is smaller than 0.5 under the condition, $x + y = 1$.

Additionally, we also construct user profile $\boldsymbol{P}^{per}$ considering persistent preferences. In order to do that, as described in Section 5.2.1, we also introduce the concept of window size N $(N = 1, 2, \cdots, 30)$, and define S_j $(j = 0, 1, 2, \cdots N)$ as the number of Web pages the user browsed on the j^{th} day. "$j = 0$" also means "today" as shown in Figure 5.14. The user profile $\boldsymbol{P}^{per}$ is denoted as follows:

$$\boldsymbol{P}^{per} = (p_{t_1}^{per}, p_{t_2}^{per}, \cdots, p_{t_m}^{per}), \tag{5.14}$$

and each element $p_{t_k}^{per}$ is defined as follows:

$$p_{t_k}^{per} = \frac{1}{S_N} \sum_{hp=1}^{S_N} w_{t_k}^{hp} \cdot e^{-\frac{\log 2}{hl}(d - d_{t_k\_init})}, \tag{5.15}$$

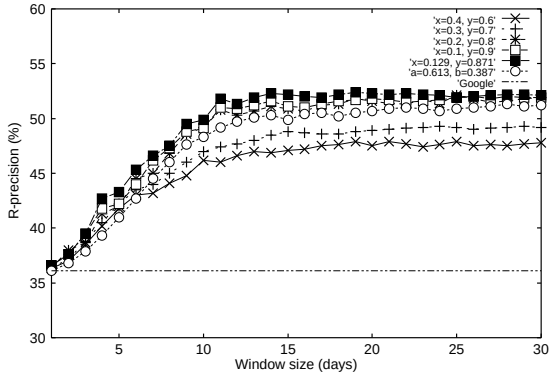


Figure 5.15. R-precision in the experiments for further improvement .

where $e^{-\frac{\log 2}{hl}(d-d_{t_{k\_init}})}$ is a forgetting factor under the assumption that user's preferences gradually decay as days pass. In this factor, $d_{t_{k\_init}}$ is the day when term t_k initially occur, d is the number of days following to $d_{t_{k\_init}}$, and hl is a half-life span parameter. The half-life span hl is set to 7. In other words, we assume that user's preferences reduce by 1/2 in one week. We also assume that each user browsed S_N pages on each day. This value S_N is different user by user. Therefore, we normalize $p^{per}_{t_k}$ using S_N as shown in Equation (5.15). Using $\boldsymbol{P}^{today}$ defined by Equation (5.13), and $\boldsymbol{P}^{per}$ defined by Equation (5.14), we construct user profile $\boldsymbol{P}$ as follows:

$$\begin{aligned} \boldsymbol{P} &= a\boldsymbol{P}^{per} + b\boldsymbol{P}^{today} \\ &= a\boldsymbol{P}^{per} + bx\boldsymbol{P}^{(br)} + by\boldsymbol{P}^{(cur)}. \end{aligned} \tag{5.16}$$

We apply Equation (5.16) to our proposed method that brought the best results, namely, user profile construction based on dynamic number of users in the neighborhood, ($a = 0.613, b = 0.387$), and verify the effectiveness of the approach described in this section.

Figure 5.15 shows the R-precision when the value of x and y are varied such that these values satisfy $x + y = 1$. As described at Equation (5.13), in order to emphasize the current user's preferences, we assign weight larger than 0.5 to y, and weight smaller than 0.5 to x.

5.3.4 Discussion

In this section, we discuss the experimental results obtained using each approach discussed in Section 5.3.2. Note that, in Figures 5.5 to 5.13, the R-precision of Google is constant because it does not depend on the window size. Table 5.1 summarizes the best precisions obtained using the methods described in Sections 5.2 and 5.3.3.

In the relevance feedback-based user profile shown in Figures 5.5 to 5.7, we found that a user profile that provides search results adaptive to a user can be constructed when a window size with about 20 days is used regardless of the number of feedbacks. The best precision (46.91%) is obtained in the case of the number of feedbacks $FB = 2$ with 26 days window size as shown in Figure 5.6 and Table 5.1. As mentioned in Section 5.3.2(a), we used query vector reformulated by relevance feedback as an initial preference of a user. However, we could not observed significant improvement in precision even if the number of feedbacks increases. We consider that this effect is caused because the initial preference of a user is absorbed by persistent preferences constructed using the window size. In addition, it is valid that we conducted experiments by examining the number of feedbacks from 1 to 3 since the precision is not improved largely in this range.

In the user profile based on pure browsing history shown in Figure 5.8, we found that a user profile that provides search results adaptive to a user can be constructed when a window size with about 15 days is used. In this method, the best precision (48.77%) is obtained when a window size with 18 days is used. This approach can achieve 1.86% higher precision compared with the best precision (46.91%) in the relevance feedback-based user profiles, and the result shows that the user's browsing history strongly reflects the user's preference.

In addition, in the user profile based on modified collaborative filtering shown in Figures 5.9 to 5.13, we found that a user profile that provides search results adaptive to a user can be constructed when a window size with about 10 days is utilized. In user profile construction based on the static number of users in the neighborhood described in Section 5.2.2(b-1), the best precision (50.82%) is obtained in the case of $n = 5$ with 17 days window size as illustrated in Figure 5.9 and Table 5.1; in other words, the 5 nearest neighbors of each user are taken. Therefore, as shown in Figures 5.9 to

5.12 and Table 5.1, we found that it is not so effective to adapt search results to each user even if more nearest neighbors are used. In addition, the user preferences of not only a certain user but also other users are exploited in this approach. We consider that this method obtained higher precision than the aforementioned approaches. In user profile construction based on the dynamic number of users in the neighborhood described in Section 5.2.2(b-2), we could obtain the best precision (51.34%) in the case of using 28 days windowsize. As shown in Figure 5.13 and Table 5.1, this is 0.52% higher precision compared with the best precision (50.82%) in the user profile construction based on the static number of users in the neighborhood described in Section 5.2.2(b-1). In this method, the neighborhood of each user is determined by the centroid vectors of clusters of users, and the number of the clusters is different user by user. Therefore, we believe that this method allows each user to perform more fine-grained search compared with the static method.

The best precision of any of the methods is obtained when a is larger than 0.6 and b is less than 0.4. This shows that search results that adapt to each user can be returned by weighting persistent preferences a little larger than ephemeral preferences. Moreover, the smaller the value of a and the larger the value of b, the larger the fluctuation in precision. Therefore, it is difficult to return search results that adapt to each user when the user profile is constructed by weighting ephemeral preferences larger than persistent preferences. Furthermore, the precision obtained by each of our proposed methods can outperform the precision obtained by Google as shown in Figures 5.5 to 5.13 and Table 5.1.

In our experiments for further improvement described in Section 5.3.3, we obtained the following findings. As shown in Figure 5.15, when the value of x is relatively large and the value of y is relatively small under the condition described at Equation (5.13), the user profile focused on browsing history in today is constructed. Therefore, when the window size is large, the user profile further focused on the past history is constructed, and that result in difficulty with following the current user's preferences. Thus, we could not obtain higher precision than user profile construction based on the dynamic number of users in the neighborhood described in Section 5.2.2(b-2). However, in this case, user's preferences can be captured using the small number of windows. On the other hand, when the value of x is relatively small and the value of y

is relatively large under the condition described at Equation (5.13), the user profile focused on the today's current session is constructed. In particular, when the window size is small, the user profile focused on the user's ephemeral preferences is constructed. Therefore, it is difficult to construct user profile that captured user's persistent preferences. We consider that this is the cause of the large fluctuation in precision in small window size. Futhermore, in this experiment, we could obtained the best precision (52.31%) when the values of x and y are 0.129 and 0.871, respectively and a widow size with 14 days is used. The improvement rate is 0.97% compared with the best precision (51.34%) in user profile construction using modified collaborative filtering based on dynamic number of users in the neighborhood described in Section 5.2.2(b-2). Therefore, in this case, we believe that the user profile that appropriately captured user's persistent and ephemeral preferences can be constructed, and that allows each user to perform much more fine-grained search that adapt to his/her information need.

Table 5.1. Comparison of the best precision obtained using our proposed methods.

	% best precision	% improvement
Google	36.10	–
Relevance feedback-based user profile ($FB = 1, a = 0.619, b = 0.381, N = 26$)	46.87	+10.77
Relevance feedback-based user profile ($FB = 2, a = 0.604, b = 0.396, N = 26$)	**46.91**	**+10.81**
Relevance feedback-based user profile ($FB = 3, a = 0.611, b = 0.389, N = 28$)	46.85	+10.75
Pure browsing history-based user profile ($a = 0.617, b = 0.383, N = 18$)	**48.77**	**+12.67**
Modified collaborative filtering-based user profile (static, $n = 5, a = 0.622, b = 0.378, N = 17$)	**50.82**	**+14.72**
Modified collaborative filtering-based user profile (static, $n = 10, a = 0.623, b = 0.377, N = 17$)	48.53	+12.43
Modified collaborative filtering-based user profile (static, $n = 15, a = 0.620, b = 0.380, N = 21$)	46.38	+10.28
Modified collaborative filtering-based user profile (static, $n = 20, a = 0.624, b = 0.376, N = 23$)	45.86	+9.76
Modified collaborative filtering-based user profile (dynamic, $a = 0.613, b = 0.387, N = 28$)	**51.34**	**+15.24**
Modified collaborative filtering-based user profile with detailed analysis of user's browsing history in one day (dynamic, $a = 0.613, b = 0.387, x = 0.129, y = 0.871, N = 14$)	**52.31**	**+16.21**

5.4 Conclusion of this Chapter

In this chapter, in order to provide each user with more relevant information, we proposed several approaches to adapting search results according to each user's need for information. Our approach is novel in that it allows each user to perform a fine-grained search, which is not performed in typical search engines, by capturing changes in each user's preferences. We conducted experiments in order to verify the effectiveness of the approaches: (1) relevance feedback and implicit approaches, (2) user profiles based on pure browsing history, and (3) user profiles based on the modified collaborative filtering. We evaluated the retrieval accuracy of these approaches. The user profile constructed based on modified collaborative filtering achieved the best accuracy. This approach allows us to construct a more appropriate user profile and perform a fine-grained search that is better adapted to each user's preferences. In the future, if broadband networks spread widely, information is expected to be provided in a variety of forms such as music, movies and so on. In addition, more information will be provided for mobile terminals such as cellular phones, PDAs, or terminals in cars for Intelligent Transportation Systems (ITS). We believe that the technique proposed in this paper can be applied to situations where users require more relevant information to satisfy their information needs. In future work, we plan to conduct experiments with a greater number of subjects and attempt to improve our proposed approaches by using a longer term of the user's browsing history.

Chapter 6

Conclusions

In this thesis, in order to improve retrieval accuracy of Web search, we studied methods for indexing the contents of Web pages more accurately, and adapting search results according to each user's need for relevant information.

In Chapter 2, we first described the framework of information retrieval, and then briefly showed the techniques for information retrieval. In Chapter 3, we reviewed related work on Web search based on its hyperlink structures, and Web search based on user's preferences.

In Chapter 4, in order to address the problem of Web search that accurate indexing of Web pages by considering their contents is not performed, we proposed the following three approaches to refining TF-IDF scheme for a target Web page by using the contents of its hyperlinked neighboring pages:

- the approach relies on the contents of all Web pages at levels up to ${L_{(in)}}^{th}$ in the backward direction and levels up to ${L_{(out)}}^{th}$ in the forward direction from the target page p_{tgt},
- the approach relies on the centroid vectors of clusters generated from Web page groups created at each level up to ${L_{(in)}}^{th}$ in the backward direction and each level up to ${L_{(out)}}^{th}$ in the forward direction from the target page p_{tgt},
- the approach relies on the centroid vectors of clusters generated from Web page groups created at levels up to ${L_{(in)}}^{th}$ in the backward direction and levels up to ${L_{(in)}}^{th}$ in the forward direction from the target page p_{tgt}.

We found that the most accurate feature vectors of a target Web page can be generated when we generate refined feature vectors using each centroid vectors of three clusters generated in a Web page group produced using all Web pages at levels up to second in the backward direction from the target page in the third method above. We could obtained 4.92% improvement in precision compared with TF-IDF scheme-based feature vector. Furthermore, when this refined feature vector of Web pages are iteratively used, the average precision is further increased by 0.1%.

In Chapter 5, in order to address the problem of Web search that most search engines cannot provide search results that satisfy each user's need for information, we proposed the following two methods for adapting search results according to each user's need for relevant information:

- pure browsing history,
- modified collaborative filtering.

We found that more fine-grained search systems that adapt to a user's preferences can be achieved by constructing user profiles based on modified collaborative filtering, especially, using dynamic number of users in the neighborhood. In this case, we could obtain 15.2% improvement in precision compared with Google. In addition, when we analyze user's browsing history behavior in more detail, we could achieve an additional 1.1% improvement.

Our proposed approaches described in this thesis contribute for indexing a target Web page more accurately, and allowing each user to perform more fine-grained search that satisfy his/her information need.

In future work, we will address the following task:

- In Chapter 4, based on the experimental results, we could infer that more accurate feature vector of Web pages can be generated by assigning higher weight to in-linked pages rather than out-linked pages of a target page. Thus, we plan to verify this inference.
- We used the K-means algorithm in order to classify the features of in- and out-linked pages of a target page. However, we have to set the number of clusters initially in this algorithm. Therefore, we plan to devise some clustering methods.

- In Section 5, we conducted experiments using the browsing history of 20 subjects for 30 days. Therefore, we plan to conduct experiments with a greater number of subjects and attempt to improve our proposed approaches by using a longer term of the user's browsing history.

In this thesis, we focused on Web page, namely text data. However, in the future, if broadband networks spread widely, information is expected to be provided in a variety of forms such as music, movies and so on. Therefore, we plan to extend method for indexing using hyperlinked neighboring pages to other media, and also plan to extend the method for adapting search results to each user's information need to other media.

References

[1] P. M. Andersen, P. J. Hayes, A. K. Heuttner, L. M. Schmandt, and I. B. Nirenberg. Automatic Extraction. In *Proc. of the Conference of the Asssociation for Artificial Intelligence*, pages 1089–1093, 1993.

[2] Ask Jeeves, Inc. Search with Authority: The Teoma Difference. http://sp.teoma.com/docs/teoma/about/searchwithauthority.html.

[3] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press, 1999.

[4] R. Baeza-Yates, F. Saint-Jean, and C. Castillo. Web Structure, Dynamics and Page Quality. In *Proc. of the 9th International Symposium on String Processing and Information Retrieval (SPIRE 2002)*, pages 117–130, 2002.

[5] M. Balabanovic and Y. Shoham. Fab: Content-Based, Collaborative Recommendation. *Communications of the ACM*, 40(3):pages 66–72, 1997.

[6] C. Basu, H. Hirsh, and W. Cohen. Recommendation as Classification: Using Social and Content-Based Information in Recommendation. In *Proc. of the 15th National Conference on Artificial Intelligence (AAAI '98)*, pages 714–720, 1998.

[7] K. Bharat and M. R. Henzinger. Improved Algorithms for Topic Distillation in a Hyperlinked Environment. In *Proc. of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '98)*, pages 104–111, 1998.

[8] D. Billsus and M. J. Pazzani. Learning Collaborative Information Filters. In *Proc. of the 15th International Conference on Machine Learning (ICML '98)*, pages 46–53, 1998.

[9] C. Blaschke and A. Valencia. The Potential Use of SUISEKI as a Protein Interaction Discovery Tool. *Genomeinformatics*, pages 123–134, 2001.

[10] J. S. Breese, D. Heckerman, and C. Kadie. Empirical Analysis of Predictive Algorithms for Collaborative Filtering. In *Proc. of the 14th Conference on Uncertanity in Artificial Intelligence (UAI '98)*, pages 43–52, 1998.

[11] S. Brin and L. Page. The Anatomy of a Large-Scale Hypertextual Web Search Engine. In *Proc. of the 7th International World Wide Web Conference (WWW7)*, pages 107–117, 1998.

[12] A. Broder and P. Raghavan. Combining Text- and Link-Based Information Retrieval on the Web. In *Pre-Conference Tutorial Note of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2001)*, 2001.

[13] A. G. Buchner and M. D. Mulvenna. Discovering Internet Marketing Intelligence through Online Analytical Web Usage Mining. *SIGMOD Record*, 27(4):pages 54–61, 1998.

[14] S. Ceri, P. Fraternali, and A. Bongio. Web Modeling Language (WebML): A Modeling Language for Designing Web Sites. In *Proc. of the 9th International World Wide Web Conference (WWW9)*, pages 137–157, 2000.

[15] S. Chakrabarti. Integrating the Document Object Model with Hyperlinks for Enhanced Topic Distillation and Information Extraction. In *Proc. of the 10th International World Wide Web Conference (WWW10)*, pages 211–220, 2001.

[16] S. Chakrabarti, B. Dom, P. Raghavan, S. Rajagopalan, D. Gibson, and J. Kleinberg. Automatic Resource Compilation by Analyzing Hyperlink Structure and Associated Text. In *Proc. of the 7th International World Wide Web Conference (WWW7)*, pages 65–74, 1998.

[17] S. Chakrabarti, D. Gibson, and K. McCurley. Surfing the Web Backwards. In *Proc. of the 8th International World Wide Web Conference (WWW8)*, pages 1679–1693, 1999.

[18] S. Chakrabarti, M. Joshi, and V. Tawde. Enhanced Topic Distillation using Text, Markup Tags, and Hyperlinks. In *Proc. of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2001)*, pages 208–216, 2001.

[19] H. Chang, D. Cohn, and A. K. McCallum. Learning to Create Customized Authority Lists. In *Proc. of the 17th International Conference on Machine Learning (ICML 2000)*, pages 49–54, 2002.

[20] C. Clake, G. Cormack, D. Kisman, and T. Lynam. Question Answering by Passage Selection (MultiText Experiments for TREC-9). http://trec.nist.gov/pubs/trec9/papers/mt9.pdf, 2000.

[21] R. Cooley, B. Mobasher, and J. Srivastava. Data Preparation for Mining World Wide Web Browsing Patterns. *Knowledge and Information Systems*, 1(1):pages 5–32, 1999.

[22] J. R. Cowie. Automatic Analysis of Descriptive Texts. In *Proc. of the ACL First Conference on Applied Natural Language Processing (ANLP-83)*, pages 117–123, 1983.

[23] F. Crivellari and M. Melucci. Web Document Retrieval using Passage Retrieval, Connectivity Information, and Automatic Link Weighting-TREC-9 Report. http://trec.nist.gov/pubs/trec9/papers/jhuapl.pdf, 2000.

[24] B. D. Davison. Topical Locality in the Web. In *Proc. of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2000)*, pages 272–279, 2000.

[25] G. F. DeJong. Prediction and Sustantiation: A New Approach to Natural Language Processing. *Cognitive Sciences*, 3:pages 251–273, 1979.

[26] G. F. DeJong. An Overview of the FRUMP System. In W. G. Lehnert and M. H. Ringle, editors, *Strategies for Natural Language Processing*, pages 149–176. Erlbaum, Hillsdale, N.J., 1982.

[27] J. Ding and D. Berleant. Mining MEDLINE: Abstracts, Sentences or Phrases? In *Proc. of the Pacific Symposium on Biocomputing (PSB 2002)*, pages 326–337, 2002.

[28] A. Field, P. Hartel, and W. Mooij. Personal DJ, an Architecture for Personalised Content Delivery. In *Proc. of the 10th International World Wide Web Conference (WWW10)*, pages 1–8, 2001.

[29] W. Frakes and R. Baeza-Yates. *Information Retrieval: Data Structures & Algorithms*. Prentice Hall, 1992.

[30] C. Friedman, P. Kra, H. Yu, M. Krauthammer, and A. Rzhetsky. GENIES: A Natural-Language Processing System for the Extraction of Molecular Pathways from Journal Articles. *Bioinformatics*, 17 (Suppl 1):pages S84–S82, 2001.

[31] X. Fu, J. Budzik, and K. J. Hammond. Mining Navigation History for Recommendation. In *Proc. of the 5th International Conference on Intelligent User Interfaces (IUI 2000)*, pages 106–112, 2000.

[32] N. Fuhr. Probabilistic Models in Information Retrieval. *The Computer Journal*, 35(3):pages 243–255, 1992.

[33] S. Fujita. Reflections on "Aboutness" TREC-9 Evaluation Experiments at Justsystem. http://trec.nist.gov/pubs/trec9/papers/jscbt9w_paper.pdf, 2000.

[34] D. Goldberg, D. Nichols, B. M. Oki, and D. B. Terry. Using Collaborative Filtering to Weave an Information Tapestry. *Communications of the ACM*, 35(12):pages 61–70, 1992.

[35] N. Good, J. B. Schafer, J. Konstan, A. Borchers, B. Sarwar, J. Herlocker, and J. Riedl. Combining Collaborative Filtering with Personal Agents for Better Recommendations. In *Proc. of the 16th National Conference on Artificial Intelligence (AAAI '99)*, pages 439–446, 1999.

[36] T. H. Haveliwala. Topic-Sensitive PageRank. In *Proc. of the 11th International World Wide Web Conference (WWW2002)*, pages 517–526, 2002.

[37] D. Hawking. Overview of the TREC-9 Web Track. *NIST Special Publication 500-249: The Ninth Text REtrieval Conference (TREC-9)*, pages 87–102, 2001.

[38] J. Herlocker, J. Konstan, A. Borchers, and J. Riedl. An Algorithmic Framework for Performing Collaborative Filtering. In *Proc. of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '99)*, pages 230–237, 1999.

[39] W. Hill, L. Stead, M. Rosenstein, and G. W. Furnas. Recommending and Evaluating Choice in a Virtual Community of Use. In *Proc. of the Conference on Human Factors in Computing Systems (CHI '95)*, pages 194–201, 1995.

[40] R. Hjelsvold, S. Vdaygiri, and Y. Léauté. Web-based Personalization and Management of Interactive Video. In *Proc. of the 10th International World Wide Web Conference (WWW10)*, pages 129–139, 2001.

[41] T. Hofmann. Collaborative Filtering via Gaussian Probabilistic Latent Semantic Analysis. In *Proc. of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '03)*, pages 259–266, 2003.

[42] K. Humphreys, G. Demetriou, and R. Gaizauskas. Two Applications of Information Extraction to Biological Science Journal Articles: Enzyme Interaction and Protein Structures. In *Proc. of the Pacific Symposium on Biocomputing (PSB 2000)*, pages 502–513, 2000.

[43] IBM Almaden Research Center. Clever Searching. http://www.almaden.ibm.com/cs/k53/clever.html.

[44] E. Ide. New Experiments in Relevance Feedback. In G. Salton, editor, *The Smart Retrieval System: Experiments in Automatic Document Processing*, pages 337–354. Prentice-Hall, Englewood Cliffs, NJ, 1971.

[45] R. A. Jarvis and E. A. Patrick. Clustering Using a Similarity Measure Based on Shared Near Neighbors. *IEEE Transactions on Computers*, C22(11):pages 1025–1034, 1973.

[46] G. Jeh and J. Widom. Scaling Personalized Web Search. In *Proc. of the 12th International World Wide Web Conference (WWW2003)*, pages 271–279, 2003.

[47] T. Joachims, D. Freitag, and T. M. Mitchell. WebWatcher: A Tour Guide for the World Wide Web. In *Proc. of the 15th International Joint Conference on Artificial Intelligence (IJCAI'97)*, pages 770–777, 1997.

[48] W. M. S. Jr, R. Burgin, and P. Howell. Performance Standards and Evaluations in IR Test Collections: Cluster-Based Retrieval Models. *Information Processing & Management*, 33(1):pages 1–14, 1997.

[49] D. Kelly and J. Teevan. Implicit Feedback for Inferring User Preference: A Bibliography. *SIGIR Forum*, 37(2):18–28, 2003.

[50] J. M. Kleinberg. Authoritative Sources in a Hyperlinked Environment. *Journal of the ACM*, 46(5):pages 604–632, 1999.

[51] J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, and J. Riedl. GroupLens: Applying Collaborative Filtering to Usenet News. *Communications of the ACM*, 40(3):pages 77–87, 1997.

[52] W. Kraaij and T. Westerveld. TNO/UT at TREC-9: How Different Are Web Documents? http://trec.nist.gov/pubs/trec9/papers/tno-ut.pdf, 2000.

[53] V. Krishnan and S. G. Chang. Customized Internet Radio. In *Proc. of the 9th International World Wide Web Conference (WWW9)*, pages 609–618, 2000.

[54] L. Li, Y. Shang, and W. Zhang. Improvement of HITS-based Algorithms on Web Documents. In *Proc. of the 11th International World Wide Web Conference (WWW2002)*, pages 527–535, 2002.

[55] W.-S. Li, K. S. Candan, Q. Vu, and D. Agrawal. Retrieving and Organizing Web Pages by "Information Unit". In *Proc. of the 10th International World Wide Web Conference (WWW10)*, pages 230–244, 2001.

[56] H. Lieberman. Letizia: An Agent That Assists Web Browsing. In *Proc. of the 14th International Joint Conference on Artificial Intelligence (IJCAI '95)*, pages 924–929, 1995.

[57] H. Lieberman. Autonomous Interface Agents. In *Proc. of the Conference on Human Factors in Computing Systems (CHI '97)*, pages 67–74, 1997.

[58] J. B. Lovins. Development of a Stemming Algorithm. *Mechanical Translation and Computational Linguistics*, 11:pages 22–31, 1968.

[59] S. Lytinen and A. Gershman. ATRANS: Automatic Processing of Money Transfer Messages. In *Proc. of the 5th National Conference of the American Association for Artificial Intelligence*, pages 93–99, 1993.

[60] J. MacQueen. Some Methods for Classification and Analysis of Multivariate Observations. In *Proc. of the 5th Berkeley Symposium on Mathmatical Statistics and Probability*, pages 281–297, 1967.

[61] U. Manber, A. Patel, and J. Robison. Experience with Personalization on Yahoo! *Communications of the ACM*, 43(8):pages 35–39, 2000.

[62] P. McNamee, J. Mayfield, and C. Piatko. The HAIRCUT System at TREC-9. http://trec.nist.gov/pubs/trec9/papers/jhuapl.pdf, 2000.

[63] P. Melville, R. J. Mooney, and R. Nagarajan. Content-Boosted Collaborative Filtering for Improved Recommendations. In *Proc. of the 18th National Conference on Artificial Intelligence (AAAI2002)*, pages 187–192, 2002.

[64] B. Merialdo, K. T. Lee, D. Luparello, and J. Roudaire. Automatic Construction of Personalized TV News Programs. In *Proc. of the 7th ACM International Conference on Multimedia (Multimedia '99)*, pages 323–331, 1999.

[65] M. Morita and Y. Shinoda. Information Filtering Based on User Behavior Analysis and Best Match Text Retrieval. In *Proc. of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '94)*, pages 272–281, 1994.

[66] L. Page. The PageRank Citation Ranking: Bringing Order to the Web. http://google.stanford.edu/˜backrub/pageranksub.ps, 1998.

[67] C. D. Paice. Another Stemmer. *SIGIR Forum*, 24:pages 56–61, 1990.

[68] M. F. Porter. An Algorithm for Suffix Stripping. *Program*, 14(3):pages 130–137, 1980.

[69] D. Rafiei and A. O. Mendelzon. What is this Page Known for? Computing Web Page Reputations. In *Proc. of the 9th International World Wide Web Conference (WWW9)*, pages 823–835, 2000.

[70] L. Rau. Extracting Company Names from Text. In *Proc. of the 7th IEEE Conference on Artificial Intelligence Applications (CAIA-91)*, pages 189–194, 1991.

[71] P. Resnick, N. Iacovou, M. Suchak, and J. R. P. Bergstorm. GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In *Proc. of the ACM 1994 Conference on Computer Supported Cooperative Work (CSCW '94)*, pages 175–186, 1994.

[72] S. E. Robertson and K. S. Jones. Relevance Weighting of Search Terms. *Journal of the American Society for Information Sciences*, 27(3):pages 129–146, 1976.

[73] J. Rocchio. Relevance Feedback in Information Retrieval. In G. Salton, editor, *The Smart Retrieval System: Experiments in Automatic Document Processing*, pages 313–323. Prentice-Hall, Englewood Cliffs, NJ, 1971.

[74] G. Rossi, D. Schwade, and R. Guimaraes. Designing Personalized Web Applications. In *Proc. of the 10th International World Wide Web Conference (WWW10)*, pages 275–284, 2001.

[75] N. Sager. *Natural Language Information Processing: A Computer Grammer of English and its Application*. Addison-Wesley, 1981.

[76] G. Salton. *The Smart Retrieval System: Experiments in Automatic Document Processing*. Prentice-Hall, Englewood Cliffs, NJ, 1971.

[77] G. Salton and C. Buckley. Term-weighting Approaches in Automatic Text Retrieval. *Information Processing & Management*, 24(5):pages 513–523, 1988.

[78] G. Salton and M. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, 1983.

[79] B. M. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-Based Collaborative Filtering Recommendation algorithms. In *Proc. of the 10th International World Wide Web Conference (WWW10)*, pages 285–295, 2001.

[80] B. M. Sarwar, G. Karypis, and J. A. Konstan. Analysis of Recommendation Algorithms for E-commerce. In *Proc. of the 2nd ACM Conference on Electronic Commerce (EC '00)*, pages 158–167, 2000.

[81] B. M. Sarwar, J. Konstan, A. Borchers, J. L. Herlocker, B. Miler, and J. Riedl. Using Filtering Agents to Improve Prediction Quality in the GroupLens Research Collaborative Filtering System. In *Proc. of the ACM 1998 Conference on Computer Supported Cooperative Work (CSCW '98)*, pages 345–354, 1998.

[82] J. Savoy and Y. Rasolofo. Report on the TREC-9 Experiment: Link-Based Retrieval and Distributed Collections. http://trec.nist.gov/pubs/trec9/papers/unine9.pdf, 2000.

[83] J. B. Schafer, J. A. Konstan, and J. Riedl. Meta-recommendation Systems: User-controlled Integration of Diverse Recommendations. In *Proc. of the 11th International Conference on Information and Knowledge Management (CIKM '02)*, pages 43–51, 2002.

[84] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock. Methods and Metrics for Cold-Start Recommendations. In *Proc. of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '02)*, pages 253–260, 2002.

[85] T. Sekimizu, H. Park, and J. Tsujii. Identifying the Interaction between Genes and Gene Products Based on Frequently Seen Verbs in Medline Abstracts. *Genome Informatics*, 9:pages 62–71, 1998.

[86] C. Shahabi and Y.-S. Chen. An Adaptive Recommendation System without Explicit Acquisition of User Relevance Feedback. *Distributed and Parallel Databases*, 14(3):pages: 173–192, 2003.

[87] U. Shardanand and P. Maes. Social Information Filtering: Algorithms for Automating "Word of Mouth". In *Proc. of the Conference on Human Factors in Computing Systems (CHI '95)*, pages 210–217, 1995.

[88] G. Silva and D. Dwiggins. Towords a Prolog Text Grammar. In *ACM SIGART Newsletter*, volume 73, pages 20–25, 1980.

[89] A. Singhal and M. Kaszkiel. AT& T at TREC-9. http://trec.nist.gov/pubs/trec9/papers/att-trec9.pdf, 2000.

[90] B. Smyth and P. Cotter. A Personalized Television Listings Service. *Communications of the ACM*, 43(8):pages 107–111, 2000.

[91] M. Spiliopoulou and L. Faulstich. WUM–A Tool for WWW Utilization Analysis. In *Proc. of the International Workshop on the World Wide Web and Databases (WebDB'98)*, pages 184–203, 1998.

[92] K. Sugiyama. A Method of Re-ranking Search Results Using their Hidden Hyperlink Structure. In *Doctoral Posters Program of 28th International Conference on Very Large Data Bases (VLDB2002)*, http://www.cs.ust.hk/vldb2002/VLDB2002-proceedings/papers/S34P14.pdf, 2002.

[93] K. Sugiyama, K. Hatano, M. Yoshikawa, and S. Uemura. A Method of Improving Feature Vector for Web Pages Reflecting the Contents of their Out-Linked Pages. In *Proc. of the 13th International Conference on Database and Expert Systems Applications (DEXA2002)*, pages 891–901, 2002.

[94] K. Sugiyama, K. Hatano, M. Yoshikawa, and S. Uemura. Extracting Information on Protein-Protein Interactions from Biological Literature Based on Machine Learning Approaches. 14:699–700, 2003.

[95] K. Sugiyama, K. Hatano, M. Yoshikawa, and S. Uemura. Refinement of TF-IDF Schemes for Web Pages Using their Hyperlinked Neighboring Pages. In *Proc. of the 14th ACM Conference on Hypertext and Hypermedia (HT '03)*, pages 198–207, 2003.

[96] K. Tajima, K. Hatano, T. Matsukura, R. Sano, and K. Tanaka. Discovery and Retrieval of Logical Information Units in Web. In *Proc. of the 1999 ACM Digital Libraries Workshop on Organizing Web Space (WOWS '99)*, pages 13–23, 1999.

[97] K. Tajima, Y. Mizuuchi, M. Kitagawa, and K. Tanaka. Cut as a Querying Unit for WWW, Netnews, E-mail. In *Proc. of the 9th ACM Conference on Hypertext and Hypermedia (HYPERTEXT '98)*, pages 235–244, 1998.

[98] L. Terveen, W. Hill, B. Amento, D. McDonald, and J. Creter. PHOAKS: A System for Sharing Recommendations. *Communications of the ACM*, 40(3):pages 59–62, 1997.

[99] the World Wide Web Consortium.
Synchronized Multimedia. http://www.w3.org/AudioVideo/, 1998.

[100] J. Thomas, D. milward, C. Ouzounis, and S. Pulman. Automatic Extraction of Protein Interactions from Scientific Abstracts. In *Proc. of the Pacific Symposium on Biocomputing (PSB 2000)*, pages 538–549, 2000.

[101] O. D. Troyer and C. J. Leune. WSDM: A User Centered Design Method for Web Sites. In *Proc. of the 7th International World Wide Web Conference (WWW7)*, pages 85–94, 1998.

[102] T. Tsandilas and M. C. Schraefel. User-Controlled Link Adaptation. In *Proc. of the 14th ACM Conference on Hypertext and Hypermedia (HT '03)*, pages 152–160, 2003.

[103] C. J. van Rijsbergen. *Information Retrieval*. Butterworths, 1979.

[104] J. Wang, Z. Chen, L. Tao, W.-Y. Ma, and L. Wenyin. Ranking User's Relevance to a Topic through Link Analysis on Web Logs. In *Proc. of the 4th ACM CIKM International Workshop on Web Information and Data Management (WIDM'02)*, pages 49–54, 2002.

[105] A. Yakushiji, Y. Tateisi, Y. Miyano, and J. Tsujii. Event Extraction from Biomedical Papers using a Full Parser. In *Proc. of the Pacific Symposium on Biocomputing (PSB 2001)*, pages 408–419, 2001.

[106] G. P. Zarri. Automatic Representation of the Semantic Relationships Corresponding to a French Surface Expression. In *Proc. of the ACL First Conference on Applied Natural Language Processing (ANLP-83)*, pages 143–147, 1983.

Appendix

A Clustering Algorithm

In this section, we review clustering algorithms used in this thesis. In the following, we define the set of N patterns as follows:

$$P = \{\boldsymbol{P}_1, \boldsymbol{P}_2, \cdots, \boldsymbol{P}_N\}, \tag{6.1}$$

where $\boldsymbol{P}_i$ ($i = 1, 2, \cdots, N$) is each feature vector. We denote each cluster as C_i ($i = 1, 2, \cdots$). In additiion, the distance between two patterns $\boldsymbol{P}_i$ and $\boldsymbol{P}_j$, and the distance between a pattern $\boldsymbol{P}_i$ and a cluster C_j are denoted as $d(\boldsymbol{P}_i, \boldsymbol{P}_j)$ and $d(\boldsymbol{P}_i, C_j)$, respectively. Furthermore, we denote the number of processed patterns and generated clusters as NP and NC, respectively.

A.1 K-Means Clustering

1. Initial K patterns are defined, and each of these patterns is regarded as standard patterns of clusters $C_1 \sim C_K$. In addition, NP and NC are set to 1 and K, respectively.

2. The pattern $\boldsymbol{P}_{NP}$ is assigned to a cluster C_j when the following equation is satisfied:

   $$d(\boldsymbol{P}_{NP}, C_j) < d(\boldsymbol{P}_{NP}, C_i), \tag{6.2}$$

   where $i = 1, 2, \cdots, K$ and $i \neq j$. The value of NP is increased by 1. If $NP > N$, NP is set to 1.

3. New standard pattern $\boldsymbol{p}_j$ is computed as follows:

   $$\boldsymbol{p}_j = \frac{1}{N_j} \sum_{\boldsymbol{P}_i \in C_j} \boldsymbol{P}_i, \tag{6.3}$$

   where N_j is the total number of patterns assigned to C_j.

4. With regard to all clusters, if the new standard pattern computed in step 3 is equal to the previous standard pattern, all of the steps are finished. Otherwise, return to step 2.

A.2 K-Nearest Neighbor Clustering

1. NP and NC are set to 1, respectively. In addition, Cluster C_{NC} whose standard pattern is $\boldsymbol{P}_{NP}$ is generated.

2. The distance defined by the following equation is computed:

$$d_j = d(\boldsymbol{P}_{(NP+1)}, C_j). \tag{6.4}$$

   d_j that is less than threshold T is sorted in ascending order, then, pattern $\boldsymbol{P}_{(NP+1)}$ is duplicately assigned up to the K^{th} cluster. The value of NP is increased by 1.

3. If $NP > N$, all of the steps are finished. If $NP \leq N$, return to step 2.

List of Publications

Journal Papers

1. K. Sugiyama, K. Hatano, M. Yoshikawa, and S. Uemura. Improvement in TF-IDF Scheme for Web Pages Based on the Contents of their Hyperlinked Neighboring Pages. *The Transactions of The Institute of Electronics, Information and Communication Engineers (IEICE)*, J87-D-I(2):pages 113–125, 2004. (in Japanese).

Letters

1. K. Sugiyama, K. Hatano, M. Yoshikawa, and S. Uemura. Improvement in TF-IDF Scheme for Web Pages and its Retrieval Accuracy. *The Database Society of Japan (DBSJ) Letters*, 2(1):pages 23–26, 2003.

International Conferences

1. K. Sugiyama, K. Hatano, M. Yoshikawa, and S. Uemura. User-Oriented Adaptive Web Information Retrieval Based on Implicit Observations. In *Proc. of the 6th Asia Pacific Web Conference (APWeb'04)*, 2004.

2. K. Sugiyama, K. Hatano, M. Yoshikawa, and S. Uemura. Refinement of TF-IDF Schemes for Web Pages Using their Hyperlinked Neighboring Pages. In *Proc. of the 14th ACM Conference on Hypertext and Hypermedia (HT '03)*, pages 198–207, 2003.

3. K. Sugiyama, K. Hatano, M. Yoshikawa, and S. Uemura. A Method of Improving Feature Vector for Web Pages Reflecting the Contents of their Out-Linked Pages. In *Proc. of the 13th International Conference on Database and Expert Systems Applications (DEXA2002), Lecture Notes in Computer Science (LNCS) 2453*, pages 891–901, Springer-Verlag, 2002.

4. K. Sugiyama. A Method of Re-ranking Search Results Using their Hidden Hyperlink Structure. In *Doctoral Posters Program of the 28th International Conference on Very Large Data Bases (VLDB2002)*, http://www.cs.ust.hk/vldb2002/VLDB2002-proceedings/papers/S34P14.pdf, 2002. (poster presentation).

Other Publications

International Conferences

1. K. Sugiyama, K. Hatano, M. Yoshikawa, and S. Uemura. Extracting Information on Protein-Protein Interactions from Biological Literature Based on Machine Learning Approaches. *The 2nd NAIST Bio-COE International Symposium on Molecular Network in Cellular Signal Transduction and Environment Responses*, 2004. (poster presentation).

2. K. Sugiyama, K. Hatano, M. Yoshikawa, and S. Uemura. Extracting Information on Protein-Protein Interactions from Biological Literature Based on Machine Learning Approaches. In *Proc. of the 14th International Conference on Genome Informatics (GIW 2003), Genome Informatics*, 14:pages 699–700, Universal Academy Press, 2003.

3. K. Yamada, K. Sugiyama, Y. Yonamine, and H. Nakagawa. Identification of Coreference Between Names and Faces. In *Proc. of the 37th Annual Meetings of the Association for Computational Linguistics (ACL'99) Workshop on Coreference and Its Applications*, pages 17–24, 1999.

Domestic Conferences

1. K. Sugiyama, K. Hatano, M. Yoshikawa, and S. Uemura. Adaptive Web Search Based on User's Implicit Preference. In *Proc. of the 15th Data Engineering Workshop (DEWS2004), The Institute of Electronics, Information and Communication Engineers (IEICE)*, 2004.

2. K. Sahoda, K. Sugiyama, K. Hatano, M. Yoshikawa, and S. Uemura. A Method for Analyzing Bookmark to Support Information Retrieval. In *Technical Report of The Institute of Electronics, Information and Communication Engineers (IEICE)*, 103(192), DE2003-81, pages 19–24, / In *SIG Technical Report of Information Processing Society of Japan (IPSJ)*, 2003(72), 2003-DBS-131-74, pages 25–32, 2003. (in Japanese).

3. K. Sugiyama, K. Hatano, M. Yoshikawa, and S. Uemura. On Some Methods for Improving Feature Vectors for Web Pages and their Retrieval Accuracy. In *Proc. of the 14th Data Engineering Workshop (DEWS2003), The*

Institute of Electronics, Information and Communication Engineers (IEICE), http://www.ieice.org/iss/de/DEWS/proc/2003/papers/2-B/2-B-03.pdf, 2003. (English Session).

4. K. Sugiyama, K. Hatano, M. Yoshikawa, and S. Uemura. The Method of Improving Feature Vectors for Web Pages Reflecting the Contents of Its Linked Pages. In *Proc. of the 13th Data Engineering Workshop (DEWS2002), The Institute of Electronics, Information and Communication Engineers (IEICE)*, http://www.ieice.org/iss/de/DEWS/proc/2002/papers/A1-3.pdf, 2002. (in Japanese).

5. K. Sugiyama, K. Hatano, M. Yoshikawa, and S. Uemura. The Method of Extracting Subject of Web Page Exploiting Forwarded Links. In *Proc. of the DB-Web2001, Information Processing Society of Japan (IPSJ) Symposium Series*, 2001(17), pages 209–216, 2001. (in Japanese).

6. K. Sugiyama, K. Yamada, Y. Yonamine, and H. Nakagawa. 'Topical Face' Detection Using Decision Tree in Photograph Newspaper. In *Technical Report of The Institute of Electronics, Information and Communication Engineers (IEICE)*, 99(450), PRMU99–154, pages 31–38, 1999. (in Japanese).

7. K. Yamada, K. Sugiyama, Y. Yonamine, and H. Nakagawa. Making Relations Between Expressions and Pictures in Newspaper. In *Proc. of the 4th Symposium on Intelligent Information Media (IIM98), The Institute of Electronics, Information and Communication Engineers (IEICE)*, pages 39–46, 1998. (in Japanese).

8. K. Yamada, K. Sugiyama, and H. Nakagawa. Learning Relations Between Linguistic Expressions and Pictures in Newspaper. In *Technical Report of The Institute of Electronics, Information and Communication Engineers (IEICE)*, 97(593), NLC97–63, pages 65–70, / 97(595), PRMU97–258, pages 55–60, 1998. (in Japanese).