

# Weighting Document Genre in Enterprise Search

by

Peter Chun Kai Yeung

A thesis  
presented to the University of Waterloo  
in fulfillment of the  
thesis requirement for the degree of  
Master of Mathematics  
in  
Computer Science

Waterloo, Ontario, Canada, 2007

©Peter Yeung, 2007

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Peter C.K. Yeung

## Abstract

The creation of an Enterprise Search system involves many challenges that are not present in Web search. Searching a corporate collection is influenced both by the structure of the data present in the collection and by the policies of the corporation. These structures and policies may differ from corporation to corporation, and from collection to collection.

In particular, an Enterprise Search system must take a document's genre into account. Examples of document genre within a corporate collection might include FAQs, white papers, technical reports, memos, emails and chat messages. Depending on an individual's current work task, it might be appropriate to give one genre a greater weight than another during the processing of a search request. Moreover, this weighting may change as the individual's work task changes.

The work presented in this thesis adapts the Okapi BM25 scoring function to weight term frequency based on the relevance of a document genre to a work task. The method utilizes two user-provided resources, *relevance judgments* and *clickthrough data*, to estimate a realistic weight for each task-genre relationship. Using this approach, the method matches the purpose of each user search request with the purpose of each document. Therefore, the proper documents are returned to the user and her/his need can be fulfilled.

The method has been incorporated into a prototype search engine, X-site, currently deployed on a corporate intranet. X-Site is a contextual search engine that uses the relationships between work tasks and document genres to improve search precision for software engineers. The system provides a customized and user-controlled means of refining search results to suit the task context of a user. Through X-Site, each employee can make a single search request and has access to documents from the Internet, a corporate intranet, and Lotus Notes databases.

## Acknowledgments

I would like to thank my supervisor, Dr. Charles L.A. Clarke, for giving me a chance to do what I truly enjoy and for providing an environment in which I was able to succeed. I could not have asked for a better opportunity. I enjoyed those walks to Waterloo Park. I would also like to thank my readers, Dr. Gordon Cormack and Dr. Olga Vechtomova, for their valuable feedback on this thesis.

Thanks to Luanne Freund for her suggestions and ideas.

Thanks to everyone in the Information Retrieval Group and Programming Language Group for their friendships. Special thanks to Stefan Büttcher for his valuable suggestions, constructive feedback on this thesis, and his development on Wumpus. Thanks to Maheedhar Kolla for his recommendation on the evaluation set. Thanks to Brad Lushman for those late night discussions.

I would like to thank my parents, Martin and Lai Fong, and my sisters, Rainbow and Tiffany, for their support throughout the years.

Thanks to Raymond So and Doris So for their encouragement and care. Special thanks to Fiona So (and Barbie So) for being my audience before every presentation and her support, care, and love.

Thanks to Mark Floyd, Craig Howard, and Karney Li for their loyal friendships. I will always cherish Waterloo as a place where we spent good times together.

Finally, thanks to the University of Waterloo for providing a great environment for me to succeed and grow.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Introduction . . . . .	1
1.2	Outline . . . . .	5
1.2.1	Work Task and Document Genre . . . . .	5
1.2.2	Incorporating Task-Genre Weights into Okapi BM25 . . . . .	5
1.2.3	Learning Weights for Task-Genre Pairs . . . . .	6
1.2.4	X-Site . . . . .	6
<b>2</b>	<b>Background Information</b>	<b>7</b>
2.1	Chapter Overview . . . . .	7
2.2	Enterprise Search . . . . .	7
2.3	Search Task . . . . .	10
2.4	Document Genre . . . . .	13
2.5	Document Scoring Function—Okapi Best Match . . . . .	14
2.5.1	Okapi BM15 . . . . .	15
2.5.2	Okapi BM11 . . . . .	15
2.5.3	Okapi BM25 . . . . .	16
2.6	Relevance Feedback . . . . .	17
2.7	Implicit Feedback . . . . .	18
2.7.1	Clickthrough Data . . . . .	19
2.8	Chapter Summary . . . . .	20

<b>3</b>	<b>Task-Genre Relationships</b>	<b>21</b>
3.1	Work Tasks and Document Genres . . . . .	21
3.2	The Simple Approach — Filtering . . . . .	22
3.3	A More Practical Approach — Weighting . . . . .	26
3.4	BM25 Retrieval Model . . . . .	27
3.5	Okapi BM25 with Task-Genre Weights . . . . .	28
3.6	An Example Scenario . . . . .	29
3.7	Chapter Summary . . . . .	30
<b>4</b>	<b>Learning Weights for Document Genres</b>	<b>31</b>
4.1	Weighting Document Types using Relevance Judgments . . . . .	31
4.1.1	Relevance Judgments . . . . .	31
4.1.2	Pros and Cons of Relevance Judgments . . . . .	32
4.1.3	Learning the Weights . . . . .	33
4.1.4	Experimental Setup — The Corpus . . . . .	35
4.1.5	Limitations . . . . .	37
4.1.6	The Weights . . . . .	37
4.1.7	Results . . . . .	38
4.2	Weighting Document Types based on Click Frequency . . . . .	40
4.2.1	Pros and Cons of Clickthrough Data . . . . .	41
4.2.2	Weighting Document Genres . . . . .	42
4.2.3	Experimental Results . . . . .	42
4.3	Chapter Summary . . . . .	44
<b>5</b>	<b>X-Site</b>	<b>45</b>
5.1	Chapter Overview . . . . .	45
5.2	The System . . . . .	46
5.3	Project Motivation . . . . .	49
5.4	Anticipated Benefits . . . . .	50
5.5	The Components . . . . .	51
5.5.1	Genre Classifier . . . . .	52
5.5.2	Language Identifier . . . . .	52

5.5.3	Document Type Converter . . . . .	54
5.5.4	Clickthrough Data Collection . . . . .	55
5.5.5	Wumpus Search Engine . . . . .	59
5.6	A Walk-through of the Search Process . . . . .	67
5.7	Chapter Summary . . . . .	78
<b>6</b>	<b>Conclusions</b>	<b>79</b>
<b>7</b>	<b>Future Work</b>	<b>81</b>
7.1	User Study . . . . .	81
7.2	Using Annotations . . . . .	83
7.3	SVM Scores . . . . .	84
7.4	Additional Information on Result Pages . . . . .	84
7.4.1	Best Links by Project . . . . .	86
7.4.2	Expert List . . . . .	86

# List of Tables

3.1	Document Genres and Work Tasks in an Software Engineering Work Domain	23
4.1	W3C collection . . . . .	36
4.2	Relevance Judgments . . . . .	38
4.3	Genre Weights . . . . .	38
4.4	Click Frequencies and Genre Weights . . . . .	43
4.5	<i>p-values</i> for BM25+CF . . . . .	44
5.1	Categories for Document Genres in X-Site . . . . .	53
5.2	Field weights of a structured document. . . . .	60
5.3	Categories for Work Tasks in X-Site's Task Profile . . . . .	63
5.4	Categories for Information Goals in X-Site's Task Profile . . . . .	64
5.5	Task-Genre Association Matrix in X-Site . . . . .	66



# List of Figures

2.1	An employee's view of the information world . . . . .	9
2.2	A screen shot of the Trevi Toolbar . . . . .	11
2.3	The classic Information Retrieval model . . . . .	12
3.1	A <i>cookbook</i> document . . . . .	24
3.2	A <i>F.A.Q.</i> document . . . . .	25
3.3	An example scenario for weighted term frequency . . . . .	30
4.1	Google search results for "David Beckham" . . . . .	34
4.2	A sample topic and the corresponding qrels file . . . . .	36
4.3	Precisions for topics 52-78, using topics 79-105 for training . . . . .	39
4.4	Precisions for topics 79-105, using topics 52-78 for training . . . . .	39
4.5	Precisions for BM25 and BM25+CF . . . . .	43
5.1	X-Site: Contextual Search System . . . . .	46
5.2	Screen shot of X-Site . . . . .	47
5.3	Features and Components of X-Site . . . . .	48
5.4	Converting Different Document Types into ASCII text . . . . .	54
5.5	A query log file . . . . .	55
5.6	Collecting clickthrough data . . . . .	56
5.7	A clickthrough data log file . . . . .	57
5.8	A result page log file . . . . .	58
5.9	Wumpus: File System Search . . . . .	59
5.10	Metadata of a document . . . . .	61

5.11 X-Site: Basic Query with no work task identified . . . . .	68
5.12 A non-classified document . . . . .	70
5.13 X-Site: Architecture and Learn a Topic . . . . .	71
5.14 X-Site: Performance and Learn a Topic . . . . .	73
5.15 X-Site: Performance and How-To . . . . .	75
5.16 X-Site: Proof of Concept and How-To . . . . .	77
7.1 Relevance Feedback form on ibm.com . . . . .	82
7.2 Best Links and Expert List on result page . . . . .	85

# Chapter 1

## Introduction

### 1.1 Introduction

An information retrieval system is used when a user needs to find relevant documents to satisfy her/his information need. When a user comes upon a problem or an information need, s/he uses a query of terms to describe her/his need and submits the query to a retrieval system. A retrieval system then compares the query with the content of each document in the collection and returns a list of ranked documents, which are believed to be relevant to the need, to the user. Given this list of documents, the user can choose documents that are believed to be relevant and obtain the necessary information by reading the selected documents. However, this approach creates a number of problems:

- an information need cannot always be expressed by a string of words;
- English words are ambiguous; and
- user-submitted queries are often short thus, it is difficult to predict the information need.

Consider the World Wide Web (WWW) as an example. There are literally billions of documents, millions of different terms, and thousands of topics. Retrieving a suitable list of documents for an information need in this setting can be a difficult task. If the

submitted query is short and not descriptive of the user's need, a retrieval system cannot accurately understand what the user is seeking and, in essence, cannot determine which documents are relevant (or irrelevant). As a result, the system's performance is worsened and users are unable to find what they seek.

The creation of an Enterprise Search system faces similar challenges along with many others that are not present in Web search. Searching a corporate collection is influenced both by the structure of the data present in the collection and by the policies of the corporation. These structures and policies may differ from corporation to corporation, and from collection to collection. For instance, anchor text is a very useful piece of information for Web search but, it is not the case for enterprise search because of the lack of relationships between documents in the collection.

Unlike a Web search request, the purpose behind each query can be precisely defined in the enterprise search setting. A searcher typically uses an enterprise search system to seek answer to a problem that s/he has on hand. This problem should relate to some *work task* that s/he is currently working on. In other words, each query submitted to an enterprise search system is motivated by a task that relates to the work environment. If a user is not searching for a work-related task, then s/he would most likely not be searching through the organization's collection and would be using a WWW search engine (e.g., Google<sup>1</sup>). Therefore, we can make a strong assumption that, when an employee uses an enterprise retrieval system, s/he is seeking documents that are related to a work task.

Depending on an individual's current work task, it might be appropriate to give one document genre a greater weight than another during the processing of a search request. *Document genre* is a class of documents, grouped together based on similar subject, form, and content. It defines the purpose of a document. Examples of document genre within a corporate collection might include FAQs, white papers, technical reports, memos, emails and chat messages. The purpose of a *FAQs* document is to answer questions that employees often encounter. The purpose of a *technical reports* document is to illustrate the technical details of a software product.

Recently, Freund et al. [FTC05] showed that there exists a relationship between work task (or information task) and document genre in a software engineering workplace. By

---

<sup>1</sup><http://www.google.com>

relating work task and document genre, an enterprise search system can consider which document genre is preferred by a user and then rank documents in this genre higher in the result list. Conversely, documents from any unrelated (or irrelevant) genres would be filtered out from the result list. As a result, retrieval accuracy is improved.

Filtering out documents from any unrelated genres is a simple approach for improving retrieval accuracy, however, it may not be the ideal approach. The main drawback for this is that if a relevant document is classified into an unrelated genre, either by the automatic classifier or mistakenly by the human assessor, then this document is filtered out from the result list. Hence, the user would never get a chance to choose to read this document. This breaks the principle of information retrieval: any document that has a chance of being relevant should be included in the result list.

Another drawback of the filtering approach is that strengths between the relationships of work task and document genre are different. Each task-genre pair has its own level of association. If a document genre is closely related to a work task, this task-genre pair is believed to have a strong relationship. If a document genre is only somewhat related to a work task, this pair has a weak relationship. For the filtering approach, work task and document genre only have a binary relationship (related or not related). Therefore, this approach eliminates valuable knowledge on each task-genre relationship.

A more practical approach is to *weight* each task-genre pair by the strength of its relationship. That is, if a work task and a document genre have a strong relationship, more weight should be given to documents from this genre so that it would be ranked higher by the retrieval system. If a task and a genre have a weak relationship, then less weight should be given while these documents would still have a chance to be retrieved by the system. If a task and a genre have no relationship, then zero weight would be given and documents from this genre would be eliminated from the result list. The latter is an extreme case and it requires the human assessor to be absolutely certain that the task-genre pair is unrelated.

Weighting each task-genre relationship seems like a realistic and logical approach. So, the question is “*How do you assign or learn the weight for each task-genre relationship accurately?*” It would be inappropriate to simply guess or assign a value to each relationship as its weight. Each weight should be a realistic estimate of the strength of each relation-

ship based on some reliable evidence. In information retrieval, two helpful resources for determining document relevance are human relevance judgments and clickthrough data.

The evaluation of retrieval systems requires the existence of a set of relevance judgments on the documents in a collection for a given query or topic. The quality of a system is measured by the number of relevant documents it retrieved at the top result page. Relevance judgments are produced by human assessors thus, it is impossible to judge every document in the collection for all topics. Some methods have been proposed to minimize the effect of incomplete judgment set (e.g., pooling). Recently, Büttcher et al. [BCYS07] proposed building a complete judgment set from an incomplete set using a linear text classifier. In general, relevance judgments are important hints for understanding which document genre is relevant to a topic and which one is not.

At the European Conference on Information Retrieval (ECIR) in 2007, a method was proposed for learning the weight of each task-genre relationship from relevance judgments [YBCK07]. This poster used document type instead of document genre for simplicity. The conclusion is clear: given a work task, the weights of these relationships can be estimated using a Bayesian approach on relevance judgments and, hence, retrieval accuracy can be improved.

Another known fact that can be used to estimate genre relevance is clickthrough data. At the International ACM SIGIR Conference in 2007, a method was proposed for estimating each task-genre weight using clickthrough data [YCB07]. Clickthrough data is a history about user-submitted queries and user-selected documents on the corresponding search result page. Although clickthrough data do not provide direct indication on document relevance and can also be *noisy*, they provide useful hints for determining which document (or type of document) is relevant to a user's need. Many different methods of utilizing clickthrough data to improve retrieval performance have been proposed (e.g., [Joa02b, STZ05a, XZC<sup>+</sup>04]). Agichtein et al. [ABDR06, ABD06] showed how *noisy* clickthrough data can be incorporated into the ranking model of a retrieval system.

In this thesis, we take a simpler approach of utilizing clickthrough data in the retrieval process. Clickthrough data are grouped together based on different task-genre pairs. To determine the weight for each task-genre pair, we consider the click frequency of the document genre when the work task was given. For example, given a work task, if genre

A is clicked more frequently than genre B, then genre A's weight would be larger than genre B's. Depending on the document's genre and on the given work task, we apply the corresponding weight to Okapi BM25F to compute the relevance score.

In the remainder of this chapter, a detailed outline of this thesis is shown.

## 1.2 Outline

### 1.2.1 Work Task and Document Genre

User context plays a key role in how documents should be ranked in the result list. User context is the set of circumstances or facts that surrounds how a user forms his/her search query. These circumstances include user needs, goals, preferences, interests, work tasks, expertise, etc. The belief is that, if a retrieval system knows the user context behind each search session, retrieval accuracy can be improved by tailoring the list of documents returned to the user.

In this thesis, we focus on one important contextual factor in enterprise search: work task. Once the work task is known for a search session, a retrieval system can focus on retrieving documents that contain information about the work task. Document genre has previously been shown [FTC05] to be a good indicator for determining which documents contain useful information for a specific work task.

### 1.2.2 Incorporating Task-Genre Weights into Okapi BM25

Task-genre weights are important factors for ranking documents in the retrieval process. Many approaches can be taken to incorporate these weights into the retrieval process. For example, after determining the relevance score of a document, a system can modify the score by multiplying it with the weight. However, this is a naive approach and it is theoretically unproven.

In Chapter 3, we present a theoretically sound approach for incorporating task-genre weights into the popular relevance scoring function, Okapi BM25. Okapi BM25 is used to determine how relevant a document is to a given query based on term frequency, document length, and other collection statistics. We modify Okapi BM25 and treat document genre

as an important field of a structured document. By incorporating task-genre weights into Okapi BM25, each document's relevance scoring is influenced by the genres that it belongs to and by the work task given by the user. In essence, the weight-influenced relevance scores produce a new ranking that is more suitable to the work task and the user.

### 1.2.3 Learning Weights for Task-Genre Pairs

After illustrating an approach for incorporating task-genre weights into Okapi BM25, we need to explore methods for estimating the appropriate adjustments for these weights. As mentioned, these weights should be estimated based on some reliable source. Chapter 4 explains two approaches for learning the weight for each task-genre pair in detail. The two approaches are: 1) learning the weights using document judgments and 2) estimating the weights using clickthrough data. It also discusses the advantages and disadvantages of using these sources to learn the weights for task-genre pairs.

Chapter 4 also presents experimental results to verify that both approaches improve retrieval accuracy. For simplicity, the experiments studied the relationships between work tasks and document types, instead of document genres. *Document type* is defined as the source of a document (e.g., email message, web page, etc.), whereas document genre is classified based on similar subject, form, and content.

### 1.2.4 X-Site

Using the methodologies described above, we have implemented a contextual search tool and deployed it in a major technology corporation. *X-Site* is an enterprise search tool for the software engineering domain that exploits relationships between the user's tasks and the document genres in a collection. The analysis enabled us to identify task-dependent patterns of genre preference, which we incorporated into the ranking algorithm of X-Site.

Chapter 5 illustrates each contextual component inside X-Site and demonstrates how retrieval quality can be improved by utilizing its *task profile* in a search session.



# Chapter 2

## Background Information

### 2.1 Chapter Overview

This thesis is an investigation on weighting document genres, based on a specified search task, to improve retrieval accuracy in an enterprise search environment. The intuition is that, for a particular work task, some document genres are more important than others and, thus, documents from these more important genres should be ranked higher in the result list.

This chapter provides the background materials for the method introduced in this thesis. This includes information on enterprise search, work task, document genre, and the retrieval scoring function that is considered in this thesis, *Okapi BM25*.

### 2.2 Enterprise Search

A retrieval system for enterprise search operates in a different environment from any other retrieval system (e.g., Web Search Engine). The enterprise search environment is mainly influenced by the structure and policy of a corporation, which affects how each document is created or modified. For instance, in a corporate collection,

- there are no spam or unsolicited documents;
- documents generally follow a similar structure;

- the purpose of a document is usually defined; and
- the amount of anchor text is limited.

The difference between a corporate collection and a web collection is the intention of their creators. Each document in a corporate collection is useful for a particular purpose. For instance, A *F.A.Q.* document is intended to answer questions and a *cookbook* document describes a step-by-step process for implementing a technology. In addition, employees are more willing to cooperate with the search engine to improve search quality. Therefore, each document may contain valuable information to improve search quality and a corporate collection is dependably structured.

From a user's point of view, enterprise search is also different from web search. For web search, a retrieval system is searching millions of documents in an unknown environment. There is no evidence to tell which document is related to a user and which is relevant to a search request. To determine document relevance, a web search system looks at the content of documents in the collection and uses various search techniques such as, PageRank<sup>1</sup> [PBMW98] and anchor text<sup>2</sup> [CDR<sup>+</sup>98].

For enterprise search, a retrieval system is searching through a hierarchy of documents. Figure 2.1 shows a user's view of the information world. An employee has direct access to documents on his/her own computer, shared files, and portals within her/his local office. Then, s/he has access to documents in a corporate collection. For a world-wide corporation, this collection contains information from offices located in different countries. Finally, the employee has access to the Internet. This hierarchy is helpful in determining document relevance. For instance, a document from the local office has a higher chance of being relevant than one from the Internet. The reason is that documents from the local office are often related to a user's work task while it might not be the case for any web pages. Therefore, an enterprise search system operates in a different environment from a web search system and effective web search techniques might not have the same results in enterprise search.

---

<sup>1</sup>PageRank is a link analysis algorithm that assigns a weight to each document with the purpose of measuring its relative importance within a hyperlinked set.

<sup>2</sup>Anchor text is the text shown on a hyperlink. It is the text that a person clicks on when s/he clicks a link.

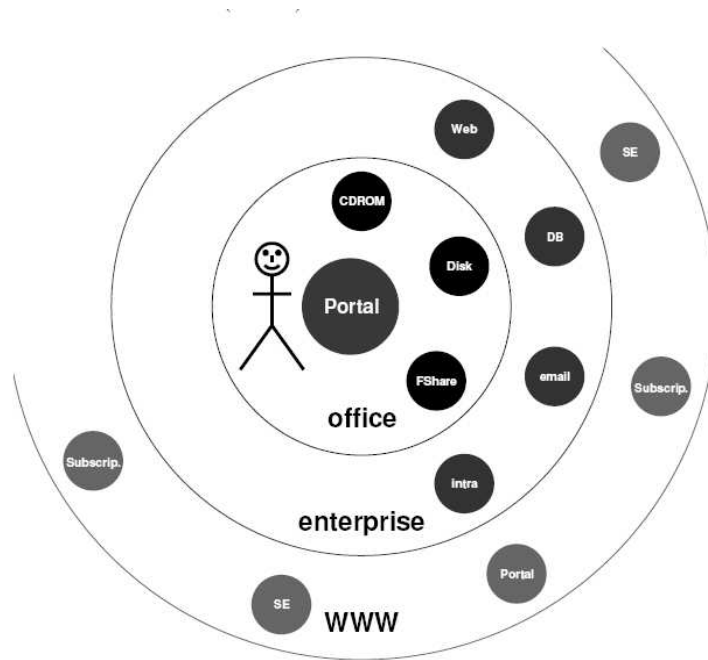


Figure 2.1: An employee's view of the information world

Since searching a corporate collection is very different from searching a web collection, enterprise search faces many different challenges. Hawking [Haw04] characterized *enterprise search* as follows:

- any organization with text content in electronic form;
- search of the organization's external web site;
- search of the organization's internal web sites (its intranet); and
- search of the other electronic text held by the organization in the form of email, database records, documents on fileshares and the like.

Many studies have been conducted in an enterprise search environment. However, many ideas generated by these studies failed when deployed in a real-world enterprise setting because of the complexity of actual enterprise information spaces. Search tools

that achieved high performance in a laboratory could still fail in a real enterprise setting. It is difficult for researchers to show that their ideas are feasible solutions to the enterprise search problem.

The Enterprise track at the Text REtrieval Conference (TREC) <sup>3</sup> [CSdV06] provides a domain where researchers can experimentally evaluate their retrieval techniques with a realistic enterprise collection. This data collection is crawled from the World Wide Web Consortium (W3C) <sup>4</sup>, which contains emails, web pages, wiki pages, development pages, personal pages, and some miscellaneous pages. This collection is a close reflection of what a typical enterprise collection would appear to be. Experimental results obtained from this collection can, to some extent, be convincing.

In addition, one major difference between enterprise search and web search is the lack of anchor text in a corporate collection. Dmitriev et al. [DEFS06] studied using annotations in enterprise search to replace the application of anchor text in web search. Their experiments were conducted in a major technology firm. Annotation is a short description of the contents in a document. Since there are many limitations in an enterprise collection, anchor text might not be as powerful in enterprise search as it is in web search. Thus, their main intuition is to substitute annotation for anchor text to improve retrieval performance.

## 2.3 Search Task

Figure 2.3 is the classic model for Information Retrieval. For each search session introduced by a user, it is motivated by an information need that s/he wishes to satisfy by retrieving relevant documents from a collection. S/he represents this information need using a string of query terms, which is usually vocabulary familiar to her/him. Also, this information need is related to a search task that the user is currently encountering. For enterprise search, this search task is often related to the user's work responsibilities.

The job of a retrieval system is to estimate the relevance of each document in the collection and produce a list of documents that are believed to be the most relevant. Finally, the user can select and view any documents that s/he believes are relevant. If

---

<sup>3</sup><http://trec.nist.gov>

<sup>4</sup><http://www.w3.org>

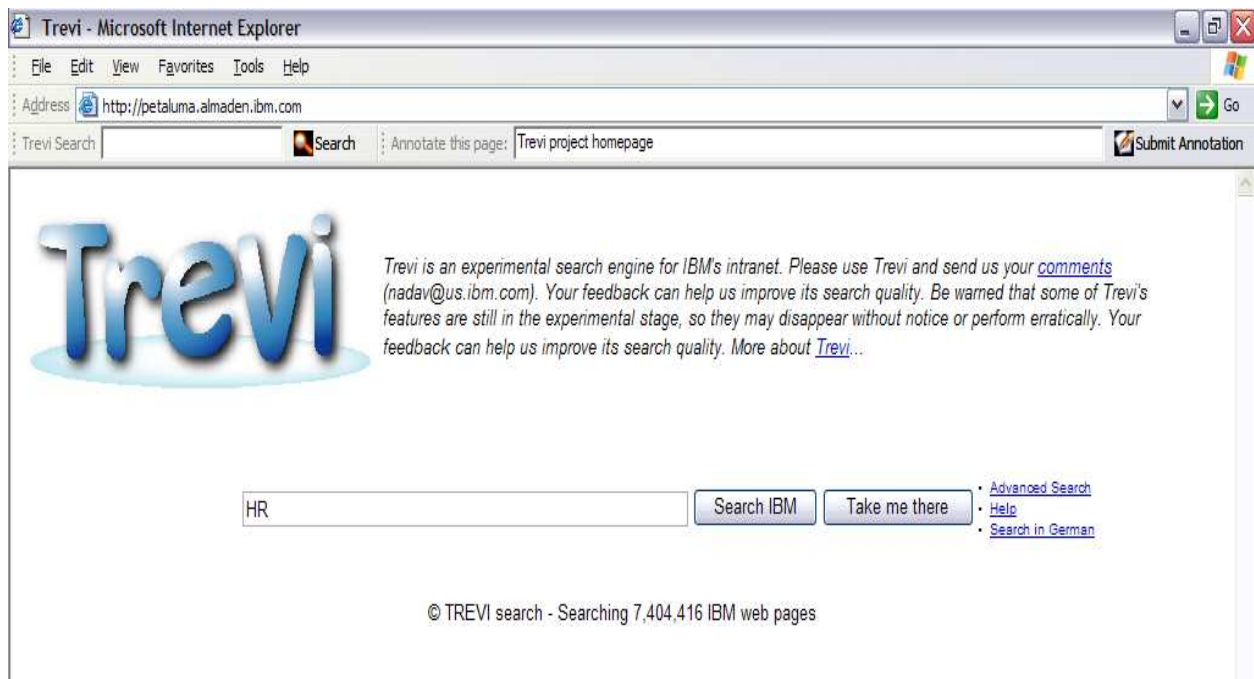


Figure 2.2: A screen shot of the Trevi Toolbar

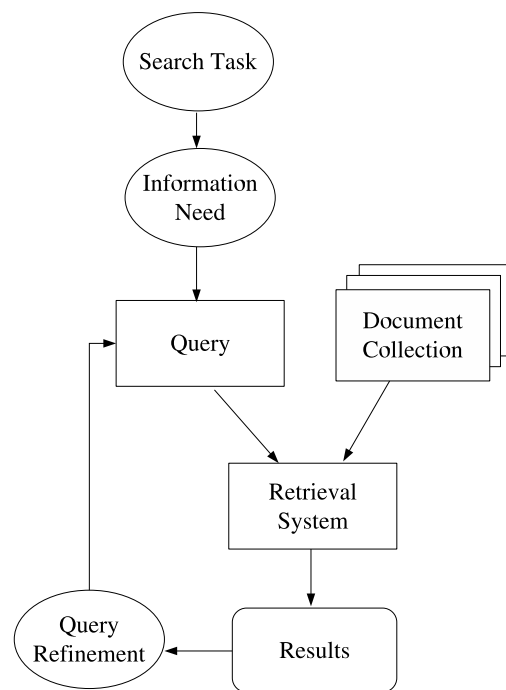


Figure 2.3: The classic Information Retrieval model

the user finds the necessary information from the result list, his/her information need is satisfied and the search session is finished. Conversely, if his/her information need is not fully satisfied, the user might choose to refine his/her query in hope of extracting a *better* result list from the retrieval system.

In order to improve retrieval accuracy, some researchers have introduced a more-refined set of search tasks in attempts to pinpoint the work task more precisely.

Several research groups have analyzed the use of context in retrieval by introducing different user interfaces. White [Whi06] proposed a search interface based on the principle of polyrepresentation, which is to offer different cognitive structures to users and to be used by them. Kelly and Fu [KF06] experimented different search interfaces for suggesting query expansion terms that require users to explicitly select terms for relevance feedback. Although these approaches improved retrieval accuracy, they alter the search process that many users are familiar and comfortable with.

## 2.4 Document Genre

User context plays a key role in how documents should be ranked in the result list. User context is the set of circumstances or facts that surrounds how a user forms her/his search query. These circumstances include user needs, goals, preferences, interests, work tasks, expertise, etc. The belief is that, if a retrieval system knows the user context behind each search session, retrieval accuracy can be improved by tailoring the list of relevant documents accordingly.

One important contextual factor in information retrieval is *document genre*. There is a rich history of classifying documents into genres and many definitions for document genre have been proposed. Some defined document genre based on the form or structure of a document [DVDM01] and some based on the purpose or function of a document [ES04]. Most commonly, some defined genre as a combination of both form and purpose [OY94]. Hence, recognizing the form of a document also means discovering the purpose of a document.

For example, a *Frequently Asked Questions* document is a sequence of questions and answers relating to a specific topic and its purpose is to answer questions that users often

have. Another example is that a *Tutorial* document provides a step-by-step lesson, which is designed for self-study.

This thesis defines *document genre* as:

*Document genre*: a class of documents, grouped together based on similar subject, form, and content. Each document may be classified into more than one genres.

Since *document genre* identifies the purpose of a document, it can be useful in information retrieval. Recognizing the genre(s) of a document means identifying the purpose of a document, which is useful for satisfying a search request. Given the goal of a search request, a retrieval system can satisfy this request by providing documents that are aimed to achieve this goal. In other words, by matching the purpose of each document and the desire of a user, documents whose purposes are related to a search request would be ranked higher on the result page. Therefore, a retrieval system can improve its search effectiveness.

For this reason, we classify each document into genres and match each genre with a set of related search tasks. Given a search task, documents from related genres would be ranked higher.

## 2.5 Document Scoring Function—Okapi Best Match

Given a query, a retrieval system calculates a relevance score for each document in the collection. This score is an estimate on document relevance, comparing to the query terms, by considering term frequency, length of each document, and other collection statistics. Documents with higher scores are *believed* to be more relevant to the query than those with low scores.

A popular choice of such scoring function is *Okapi BM25* [RWHB<sup>+</sup>94]. However, two earlier Okapi BM<sup>5</sup> models, *Okapi BM15* and *BM11*, are essential to the development of Okapi BM25. The intuitions behind these models were to consider term frequency as the main ingredient for determining document relevance.

---

<sup>5</sup>Best Match



The methods proposed in this thesis are implemented into an open source search system, Wumpus<sup>6</sup>. Wumpus implemented a variant of Okapi BM25. For this reason, this thesis uses the corresponding variants of Okapi BM15, BM11, and BM25 to illustrate our methods. The methods can be transformed to the original version, or other variants, of Okapi BM25 in a straight forward manner.

### 2.5.1 Okapi BM15

The main intuition behind Okapi BM15 is that the more frequent a query term appears in a document, the more relevant the document is to the query. For query terms  $Q_1, Q_2, \dots, Q_n$ , the BM15 relevance score of a document  $D$  is

$$S_{BM15}^{(D)} = \sum_{i=1}^n w_{Q_i} * \frac{(k_1 + 1) * f_{D,Q_i}}{f_{D,Q_i} + k_1} \quad (2.1)$$

where  $f_{D,Q_i}$  = the frequency of  $Q_i$  in document  $D$ ;

$k_1$  = a free parameter (set to 1.2).

### 2.5.2 Okapi BM11

There are two important factors that might influence the term frequency of a document: number of documents that contain the term and document length. The latter would be discussed later in this chapter.

Document length is also an important factor for determining relevance score for a document. One reason is that a long document might contain a number of unrelated stories, which are concatenated together. Users do not wish to scan over these unrelated stories before viewing the related one. Therefore, a good, relevant document is one that contain mostly related materials. Another reason is that a long document might contain mostly related materials, but it is a longer version of a short, similar document. That means, the longer version uses more words to cover a similar scope of a short document. It would then be unfair to score this longer document higher than its shorter version. In fact, a user might wish to view the shorter version because s/he can satisfy his/her information

---

<sup>6</sup><http://www.wumpus-search.org>

need quickly. Therefore, term frequency must be normalized according to the document length.

$$S_{BM11}^{(D)} = \sum_{i=1}^n w_{Q_i} * \frac{f_{D,Q_i}}{f_{D,Q_i} + k_1 * \frac{|D|}{avgdl}} \quad (2.2)$$

where  $|D|$  = the length of document  $D$ ;

$avgdl$  = the average document length in the collection.

### 2.5.3 Okapi BM25

There is much debate on which function is more proficient. In investigating the answer, Robertson et al. [RWHB<sup>+</sup>94] combined the two functions into a single function—Okapi BM25.

$$S_{BM25}^{(D)} = \sum_{i=1}^n w_{Q_i} * \frac{(k_1 + 1) * f_{D,Q_i}}{f_{D,Q_i} + k_1 * ((1 - b) + b * \frac{|D|}{avgdl})} \quad (2.3)$$

$k_1$  and  $b$  are constants used to vary the effectiveness of the scoring function with respect to different components.  $k_1$  controls the non-linear  $f_{D,Q_i}$  effect and  $b$  controls the document length normalization. Varying the values of  $k_1$  and  $b$  may lead to different retrieval performance. If  $b = 1$ , the function becomes BM11, and if  $b = 0$ , it becomes BM15. Depending on the search task, the optimal values of  $k_1$  and  $b$  are difficult to determine [BCY06]. In this thesis, the default value of  $k_1$  is 1.2 and  $b$  is 0.75.

### Inverse Document Frequency

If most of the documents in the collection contain a query term, then this term is not a good discriminator on how relevant a document is. Otherwise, most documents would be considered relevant to the query. This is not helpful because users are only interested in viewing a small number of top relevant documents. It is inappropriate for a retrieval system to present millions of documents to the users.

$w_{Q_i}$  is the *inverse document frequency (IDF)* weight, which is the significance of  $Q_i$  in determining document relevance.

$$w_{Q_i} = \log \left( \frac{\# \text{ documents}}{\# \text{ documents containing } Q_i} \right) \quad (2.4)$$

The intuition behind IDF is that if a term appears in many documents of a collection, then this term is not a good discriminator and it should be given less weight than ones that appear in fewer documents. For example, the term “the” may appear in every document in the collection, which does not help determine whether any particular document is useful or not. As a result, the IDF component gives a small weight (even possibly zero) to this term and relies on other terms in the query to provide useful judgments on document relevance. Conversely, if a term appears in only a few documents, then it provides good evidence that these documents are very likely to be relevant. In this case, more weight should be given to this term so that it can have strong influence on scoring a document. Therefore, IDF measures the importance of a term in determining document relevance and adjusts the score of a document accordingly.

Aside from scoring document relevance with a retrieval function like Okapi BM25, many re-ranking techniques have been shown for improving retrieval accuracy. These techniques include anchor text [CDR<sup>+</sup>98], PageRank [PBMW98], document structure [FTC05], user behaviour [ABDR06], user context [STZ05a], etc. All these techniques have been shown to be helpful and play an important role in standard retrieval systems.

## 2.6 Relevance Feedback

Relevance Feedback (RF) [SB97] was introduced with the intention for a retrieval system to understand which documents are relevant or irrelevant to the user. After the list of resulting documents are presented to a user, the user may explicitly specify document relevance. Given this specification, the retrieval system can automatically adjust the representation of the user’s information need by adding useful terms to the query. These additional terms are obtained from those user-defined relevant documents. As a result, the improved query is intended to improve retrieval performance. Relevance Feedback has been shown in experiments [SB97] as an effective method for improving retrieval performance.

## 2.7 Implicit Feedback

One disadvantage of relevance feedback is the extra workload introduced to the users during the search process. This might be undesirable for some users because the effect of these extra activities might not be noticeable to them. An alternative to relevance feedback is to use implicit feedback for evaluating user context. Implicit feedback is information about user interaction with the system, which includes clickthrough data, viewing time of a document, exit method, etc. Fox et al. [FKM<sup>+</sup>05] examined the relationships between implicit and relevance feedback in Web search and developed models to associate explicit ratings and implicit measures of user interest. A more extensive overview on other implicit feedback studies is presented by Kelly and Teevan [KT03].

Some research groups have also attempted to apply user context and implicit feedback together for improving retrieval performance. Shen et al. [STZ05b] introduced a decision theoretic framework for modeling user preference based on implicit feedback. Their model took a user's search context and inferred implicit feedback to build a user model for personalized search. The two main types of implicit feedback that were utilized in their approach are *query chain* and *clickthrough data*. Query chain was utilized to expand a query based on terms that were previously submitted to the system. Clickthrough data were utilized to re-rank documents presented on a result page. Experimental results showed that better retrieval accuracy was achieved by employing their framework, compared to the Google search engine.

Teevan et al. [TDH05] formulated search algorithms that consider implicit feedback to personalize a user's search request. Implicit information about a user's interests was used to re-rank search results within a relevance feedback framework. In their approach, previously submitted queries and clickthrough data were utilized to explore rich models of user interests. In addition, they utilized other information about the user such as documents and email s/he has previously read and/or created. Their methods showed that implicit feedback can be used to approximate a representation of user interests, which can significantly improve retrieval performance.

### 2.7.1 Clickthrough Data

One of the most useful types of implicit feedback is clickthrough data, which is information about user-submitted queries and user-selected documents on the corresponding search results page. The intuition is that users only click on those documents which they believe are relevant when looking at the information shown on the result page (i.e., title, URL, and a short phrase from the document). However, clickthrough data do not provide direct indication of document relevance and, thus, can also be *noisy* and *biased*.

Clickthrough data can be biased in two ways [JGP<sup>+</sup>05]. First, there is a *trust bias* that leads the user to click on a higher ranked document, even if the information shown is less relevant. The user has a certain amount of trust in the search engine that one believes a highly ranked document will more likely to be relevant. Second, there is a *quality bias* that leads the user to click on a document that is better presented by the search engine, even if the content of the document is less relevant. Therefore, clickthrough data can only be interpreted as relative to the order of presentation and relative to the quality of other abstracts.

Despite these limitations, clickthrough data still provide some attractive advantages. They are information stored in the search engine log files and can be collected in large quantities, at low cost and without burden on the users to explicitly specify which documents are relevant and which are not. Clickthrough data provide useful hints for determining which document (or type of document) is relevant to a user's need.

Many different methods of utilizing clickthrough data to improve retrieval performance have been proposed (e.g., [Joa02b, STZ05a, XZC<sup>+</sup>04]). Recently, Joachims et al. [JGP<sup>+</sup>05] presented a set of strategies for interpreting clickthrough data to determine document relevance. By performing eye tracking experiments and correlating documents with their explicit ratings, the authors showed that clickthrough data can be interpreted accurately to assess document relevance.

More recently, Agichtein et al. [ABDR06, ABD06] showed how implicit feedback data can be incorporated into the ranking model of a retrieval system. Their approach focused on incorporating *noisy* implicit feedback into a ranking model for a real world environment. In contrast, this thesis focuses on incorporating implicit feedback into the BM25 scoring function based on click frequencies in different document types.

Aside from building a re-ranking model or personalizing user searches, clickthrough data can also be utilized for other purposes. Joachims [Joa02a] showed that clickthrough data can be used to evaluate retrieval performance of a search engine. Traditional evaluation methods require relevance judgments explicitly generated by human assessors. However, Joachims proposed an evaluation method that generates unbiased feedback about the relative quality of two search results using clickthrough data. A theoretical analysis shows that this evaluation method provides the same assessments as traditional method.

## **2.8 Chapter Summary**

This chapter covered background information for the method proposed in this thesis. This thesis utilizes the relationships between work tasks and document genres, and incorporates these relationships into the document relevance scoring function—Okapi BM25. The background information for each component was presented in this chapter.

# Chapter 3

## Task-Genre Relationships

Every search session is motivated by a user’s information need. The user can satisfy this need by viewing relevant documents from a collection. In order to retrieve these relevant documents from a collection, a user chooses a string of query terms to represent her/his information need and submits the query to a retrieval system. The retrieval system then extracts all documents from a collection that are believed to be relevant to the user’s need, or relevant to the submitted query.

In the enterprise search setting, a searcher typically uses a retrieval system to seek answers to a problem that s/he has on hand. This problem should relate to some work task that s/he is currently working on. In other words, each query submitted to an enterprise search system is motivated by a task that relates to the work environment. Therefore, we can make a strong assumption that, when an employee uses an enterprise retrieval system, s/he is seeking documents that are related to a work task.

In this chapter, we describe how each work task and document genre pair can be weighted and incorporated into a retrieval scoring function, Okapi BM25, to improve retrieval accuracy.

### 3.1 Work Tasks and Document Genres

For every query submitted to an enterprise search system, there is a corresponding work task associated to it. This work task is closely related to a user’s job responsibilities. This

work task is a useful hint for a retrieval system to understand a user's information need.

Given a query and the work task associated with it, a retrieval system can focus on documents that both contain the query terms and relate to the work task. Conversely, a retrieval system can eliminate documents that are completely irrelevant to the work task. As a result, retrieval accuracy can be improved and a user is presented with a more suitable list of results.

One important point here is that, given a work task, there is a group (or groups) of relevant documents and there are groups of irrelevant documents. The challenge is to define a set of groups so that every document can be appropriately classified. Freund et al. [FTC05] classified documents into different document genres. Table 3.1 shows all work and information tasks and document genres identified in their experiments. They conducted interviews with the target population—software engineering consultants—to understand the contextual factors that influence user search behaviour and selections.

Figures 3.1 and 3.2 are sample documents from two of the genres defined in Table 3.1.

## **3.2 The Simple Approach — Filtering**

Given a user's work task, a retrieval system can determine the related relevant document genres as well as the irrelevant ones. Relevant and irrelevant document genres are useful information for a retrieval system to improve its accuracy. Filtering out documents from any irrelevant genres is a simple approach for improving retrieval accuracy, however, it may not be the ideal approach. There are a few disadvantages to this approach:

- classifying documents is not a trivial task and documents can be mis-classified;
- any mis-classified, but relevant, documents are eliminated from the result list, which means the user could not choose to view it; and
- filtering ignores the difference between the strengths of the task-genre relationships.

If a relevant document is classified into an irrelevant genre, either by the automatic classifier or mistakenly by the human assessor, then this document is filtered out from the result list. Hence, the user is stripped of the chance to choose this relevant document from



Document Genres	Work Tasks	Information Tasks
architecture/design	administration/install	compare
collection	architecture/design	contacts
cookbook	capacity planning	demonstrate
demo	competitive	document
discussion	configuration	educate
engagement	debugging	example
summary	deployment	guide/manual
lecture/lab	development	index
legal material	discovery session	market/sell
presentation	evaluation	methodology
product feedback	implementation	reference
reading material	installation	road-map
sales kit	integration	standards
schedule	migration	support
source code	performance tuning	technical info
tools	proof of concept	tool
web-site/repository	product presentation	
	project management	
	project review	
	security	
	test	

Table 3.1: Document Genres and Work Tasks in an Software Engineering Work Domain

```

<DOC>
<DOCNO> X1124895354-00012-92738 </DOCNO>
<URL> http://www-128.ibm.com/developerworks/websphere/... </URL>
<TITLE> Integration Cookbook for WebSphere Business Integration Modeler and
WebSphere Studio Application Developer Integration Edition </TITLE>
<SOURCE> Based on v14 Template Generator, Template 14.0 </SOURCE>
<CLASS> cookbook </CLASS>
<BODY>
The integration of IBM WebSphere Business Integration Modeler 5.1 (hereafter
called Modeler) and IBM WebSphere Studio Application Developer Integration Edi-
tion 5.1 (hereafter called Application Developer) is very important for every cus-
tomer using IBM's Business Integration software. Modeler enables the business
analyst to design, simulate, optimize and document the processes of the company.
Application Developer is the IT department tool for implementing and maintaining
these processes. The integration of Modeler and Application Developer is key to the
integration of business analysis and IT systems.

Our sample will be a real-life process from a bank customer that implements a
funds order process. The process checks the validity of the order details and decides
whether a manager approval is needed. Depending on the results of these verifi-
cations and approvals, the process forwards the order to an error queue, creates a
work item for the manager approval, sends a rejection mail to the customer, or – if
everything is OK – executes the order.

This cookbook contains three chapters:
Chapter 1: Model the process in Modeler – creates the sample process with Modeler
and adds data objects, decision conditions and resources.
Chapter 2: Finalize the process implementation in Application Developer – explains
the export procedure to Application Developer in BPEL format.
Chapter 3: Deploy and test the process – shows the technical implementation of
the process including the reuse of existing services and the creation of new service
implementations.
You should have basic skills in the usage of these two solutions and with the Eclipse
development environment.

...
</BODY></DOC>

```

Figure 3.1: A *cookbook* document

```

<DOC>
<DOCNO> X1124895384-00020-92782 </DOCNO>
<URL> http://www.ibm.com/developerworks/xml/library/x-dita3/ </URL>
<TITLE> Frequently Asked Questions about the Darwin Information Typing Ar-
chitecture </TITLE>
<SOURCE> Based on v14 Template Generator, Template 14.0 </SOURCE>
<CLASS> faq </CLASS>
<BODY>
General DITA questions
Why is “Darwin” in the name of this architecture?
Where can I learn more about topic-oriented writing and user assistance?
How does DITA differ from DocBook?
How will changes to the DTD be made and controlled?
May I use this DTD in my own company?
Is DITA integrated into any IBM products?
Is there an XML schema for the DITA DTDs?

The topic architecture of DITA
What is a topic?
Why topics?
What is the topic structure in the architecture?
What is progressive disclosure in a topic?
Can topics be nested?
What is an information type?
Why information types?
What is specialization?
...
Q: Why is “Darwin” in the name of this architecture?
A: The entire name of the architecture has this combined explanation:
Darwin: it uses the principles of specialization and inheritance
Information Typing: it capitalizes on the semantics of topics (concept, task, refer-
ence) and of content (messages, typed phrases, semantic tables)
Architecture: it provides vertical headroom (new applications) and edgewise exten-
sion (specialization into new types) for information ...
</BODY></DOC>

```

Figure 3.2: A *F.A.Q.* document

the result list. This contradicts the principle of information retrieval: any document that has a chance of being relevant should be included in the result list. That is, only those completely irrelevant documents are excluded.

The filtering approach also ignores the difference between the strengths of the task-genre relationships in the retrieval process. Work task and document genre only have a binary relationship (related or not related). If a document genre is closely related to a work task, this task-genre pair is believed to have a strong relationship. If a document genre is only somewhat related to a work task, this pair has a weak relationship. Therefore, this approach eliminates valuable knowledge on each task-genre relationship.

### 3.3 A More Practical Approach — Weighting

A more practical approach is to *weight* each task-genre pair by the strength of its relationship. For each task-genre relationship, it is given a value (weight) to represent the strength of the relationship. This value is then incorporated into Okapi BM25 to influence the relevance score of each document in an attempt to improve retrieval accuracy.

If a work task and a document genre have a strong relationship, more weight should be given to documents from this genre so that it would be ranked higher by the retrieval system. Conversely, less weight should be given to a weak relationship so that these documents would still have a chance to be retrieved by the system. If a task and a genre have no relationship, then zero weight would be given and documents from this genre would be eliminated from the result list. The latter is an extreme case and it requires the human assessor to be absolutely certain that the task-genre pair is unrelated.

Weighting document genres has a few advantages:

- all documents still have a chance of being retrieved regardless of the genres that they are classified into; and
- the degree of associations between work tasks and document genres can be realistically reflected in the retrieval process.

No document is automatically removed from the result list unless the task-genre pair has absolutely no relationship or the document does not contain any occurrence of the

query terms. In the latter case, the document is removed from the result list regardless of the relationship between the work task and document genre. This is the same in the unweighted model. In the former case, the system administrator must be absolutely certain that there is no relationship between the task-genre pair.

The second advantage of the weighting approach is that genre relevance is not binary; instead, document genre can be judged by different levels of relevance (e.g. irrelevant, relevant, closely relevant). The weighting approach allows the difference in genre relevance to be incorporated in the retrieval process as well. The strength of each task-genre relationship is used to weight and affect retrieval results. Thus, this provides a more realistic approach to incorporating task-genre relationships.

In this thesis, we use this approach to incorporate the task-genre relationships into the retrieval process. Now, the question is *how?* More specifically, *how can we incorporate these relationships into Okapi BM25?*

## 3.4 BM25 Retrieval Model

During the retrieval process, every document is evaluated for its relevance based on the term frequency in the document. If one document contains more occurrences of the query terms than another document, this document is considered to be more relevant than the less frequent document. If a document does not contain any occurrence of all query terms, it is highly likely that it is completely irrelevant to the query. Okapi BM25 [RWHB<sup>+</sup>94] is a popular choice for scoring document relevance based on term frequency, document length, and other collection statistics. The details of Okapi BM25 are discussed in the previous chapter.

For query terms  $Q_1, Q_2, \dots, Q_n$ , the BM25 relevance score of a document  $D$  is

$$S_{BM25}^{(D)} = \sum_{i=1}^n w_{Q_i} * \frac{(k_1 + 1) * f_{D,Q_i}}{f_{D,Q_i} + k_1 * ((1 - b) + b * \frac{|D|}{avgdl})} \quad (3.1)$$

where  $f_{D,Q_i}$  = the frequency of  $Q_i$  in document  $D$ ;

$|D|$  = the length of document  $D$ ;

$avgdl$  = the average document length in the collection;

$k_1$  = a free parameter (1.2);

$b$  = a free parameter (0.75).

### 3.5 Okapi BM25 with Task-Genre Weights

Okapi BM25 has shown its effectiveness from many TREC participations. However, one of its drawbacks is that it does not consider the structure of a document when scoring the document's relevance. Relevance estimation can be improved by considering the internal structure of a document. These structures include document title, author, abstract, content, etc. The intuition for this approach is to consider structured documents and rank them according to the importance of each structure. For example, query terms that appear in a document's title should be considered *more important* than terms that appear in the body. Hence, a more accurate relevance estimation should lead to an improved retrieval accuracy.

Recently, Robertson et al. [RZT04] introduced a modified version of BM25, Okapi BM25F, for incorporating *weights* into different fields of a structured document and calculating its relevance score by a *linear combination of the term frequencies* for all fields. For each term in a query, its frequency in document  $D$  is treated as a combination of its unweighted frequency  $f_{D,Q_i}$  and the corresponding weight  $w_j$ .

$$f'_{D,Q_i} = \sum_{j=1}^N w_j * f_{D,Q_i} \quad (3.2)$$

A similar approach can be taken for incorporate task-genre weights into Okapi BM25. The relevance score of a document is calculated by a linear combination of term frequencies, field weights, and task-genre weight.

$$f''_{D,Q_i} = w_G * f'_{D,Q_i} \quad (3.3)$$

where  $w_G =$  the weight of each task-genre pair.

### 3.6 An Example Scenario

Figure 3.3 shows an example scenario of how task-genre weights influence term frequency. Suppose the query term is **cat** and the user is trying to search for some background information on cats.

Assume that there are only three genres: wiki pages ( $G_1$ ), blog pages ( $G_2$ ), and advertisement pages ( $G_3$ ). For this particular search task, the weight for wiki pages ( $w_{G_1}$ ) is 2, for blog pages ( $w_{G_2}$ ) is 1, and for advertisement pages ( $w_{G_3}$ ) is 0.5. To search for background information, a wiki page is highly likely to contain such information on a subject. A blog page might contain this information if the author decides to write about it in his/her blog, but it is still less likely than a wiki page. Finally, it is least likely that an advertisement page would contain background information on a subject. Therefore, we place most weight on wiki pages and least on advertisement pages.

The unweighted term frequencies for *Doc1*, *Doc2*, and *Doc3* are 2, 3, and 2 respectively. For simplicity, if we ignore document length and other collection statistics, this indicates that *Doc2* is the most relevant document for the search query. However, as you can see, *Doc2* is not the most relevant document for the user's search goal. To satisfy the user's search goal, *Doc1* is more suitable and thus, should be ranked higher than *Doc2*.

Since a wiki page tends to provide background information on the subject, the system places more weights on terms that occur in such documents. By combining genre weights with term frequencies, *Doc1* becomes the most relevant document. In addition, the weighted term frequencies indicate that *Doc1* is more relevant than *Doc3*, whereas the unweighted ones could not distinguish the two. By incorporating task-genre weights into the scoring function, more information are used to determine document relevance and to provide a more accurate estimation.

This scenario shows how document genre can be used to improve search quality. When a search system understands a user's search goal, it can focus on related document genres

<p><i>Doc 1</i> A <b>cat</b> is a small carnivorous mammal that is often valued by humans for its companionship. <b>Cat</b> is intelligent and can be trained to obey simple commands....</p>	<p><i>Doc 2</i> Can a man who's never had <b>cats</b> and who is allergic to <b>cats</b> become a great "Cat Dad?" Sure, read my blog to find how...</p>	<p><i>Doc 3</i> The College Royal <b>Cat</b> Show is in its 31<sup>st</sup> year and this year's show promises to entertain all <b>cat</b> lovers...</p>
<p><i>G1 = wiki page</i> <math>f_{D,Q_i} = 2</math> <math>w_{G_1} = 2</math> <math>f'_{D,Q_i} = 4</math></p>	<p><i>G2 = blog page</i> <math>f_{D,Q_i} = 3</math> <math>w_{G_2} = 1</math> <math>f'_{D,Q_i} = 3</math></p>	<p><i>G3 = advertisement</i> <math>f_{D,Q_i} = 2</math> <math>w_{G_3} = 0.5</math> <math>f'_{D,Q_i} = 1</math></p>

Figure 3.3: An example scenario for weighted term frequency

and aim to rank documents from those genres higher on the result page. Therefore, this approach considers both *how often* a term appears in a document and *where* it appears in the collection.

### 3.7 Chapter Summary

In this chapter, we discussed that given a work task in enterprise search, document genre can be weighted and incorporated into a relevance scoring function, Okapi BM25. The weight for each task-genre pair depends on the strength of their relationship. Finally, the weight is incorporated into a modified version of Okapi BM25.



# Chapter 4

## Learning Weights for Document Genres

The last chapter illustrated that the relationships between work tasks and document genres can be used to improve retrieval accuracy. It also introduces an approach to use weights to represent the strength of these relationships. In this chapter, we discuss two different approaches to learn the appropriate weight for each task-genre pair.

The first approach takes an elementary approach to estimate the appropriate weights using human assessment on document relevance. The second approach estimates the weights using clickthrough data, which reflects any change in user preference over time.

### 4.1 Weighting Document Types using Relevance Judgments

#### 4.1.1 Relevance Judgments

The first approach takes an elementary approach to estimate the appropriate weights using existing human assessments on document relevance. Human assessments are relevance judgments, produced by a group of human assessors, on whether documents in a collection are relevant or irrelevant to a given query. For a set of search queries, a corresponding list

of relevance judgments were produced. The quality of a retrieval system is evaluated by its performance on the set of search queries and its relevance judgments. For instance, for a given query, the number of relevant documents retrieved by the system is defined as *recall* and the number of retrieved documents that were relevant is defined as *precision*.

The *recall*  $R@r$  of a retrieval system is the total number of relevance documents that were retrieved in the top  $r$  documents:

$$R@r = \frac{\text{Number of relevant documents that were ranked in the top } r}{\text{total number of relevant documents}} \quad (4.1)$$

The *precision*  $P@r$  of a retrieval system is the number of top  $r$  retrieved documents that were relevant:

$$P@r = \frac{\text{Number of top } r \text{ retrieved documents that were relevant}}{r} \quad (4.2)$$

Relevance judgments can be binary (e.g., *relevant* or *irrelevant*) or graded (e.g., *closely relevant*, *somewhat relevant*, or *irrelevant*). For those standard measurements that consider binary judgments, any level of relevance is simply considered *relevant* (e.g., *closely relevant* and *somewhat relevant* are both considered *relevant*). For the purpose of this thesis, we only consider binary relevance judgments for learning the weights of all task-genre relationships.

### 4.1.2 Pros and Cons of Relevance Judgments

We use the set of document judgments to estimate the weights (or to train our model).

The main advantage for using relevance judgments is that they are actually judgments assessed by humans. They can be treated as if there is no *noise* in this information. Each relevance judgment can be interpreted as it is because each document was carefully evaluated by a human assessor. There is, of course, the chance that the human assessor mistakenly marked a document with a wrong judgment. However, we would ignore this case.

The main disadvantage is that, since all judgments are produced by human assessors, it is nearly impossible to produce a complete set of judgments on every document in the

collection. That is, for an evaluation topic, it is not feasible to judge every document manually. In fact, it is nearly impossible to produce judgments for all documents returned by a retrieval system.

Consider the query “David Beckham”. The popular search engine Google<sup>1</sup> returns more than 12,000,000 web pages from the World Wide Web (see Figure 4.1). Consider a scenario where there are 50 human assessors and each of whom can judge 2 documents in a minute (this is really fast!), it would take approximately 2000 hours, which is nearly a month without stopping, for each assessor before the entire set of returned documents is judged. Therefore, relevance judgments used in many retrieval evaluations are often incomplete.

### 4.1.3 Learning the Weights

Using relevance judgments, we take an elementary approach to estimate the appropriate weight for each task-genre relationship. Since each weight influences the relevance score of a document, it should reflect the relative difference in document relevance between all document genres.

A trivial way to estimate the weight is by considering the likelihood that documents from a given genre are relevant to a specific task. Given a set of relevance judgments, we count the number of relevant judgments for one document genre and the number of documents in the collection.

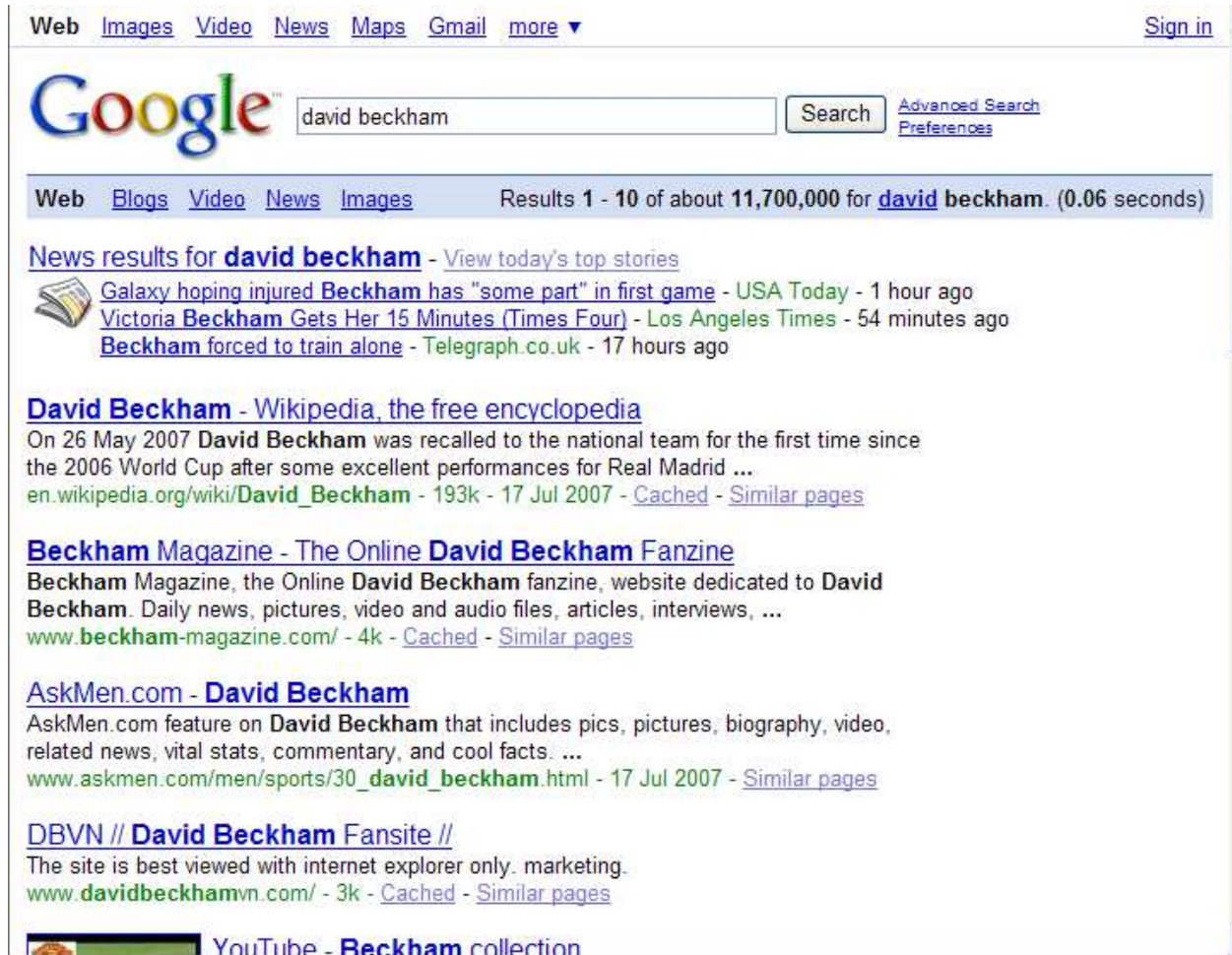
$$w_{G_j} = \frac{\text{Number of relevant documents in a document genre}}{\text{Number of documents in the collection}} \quad (4.3)$$

Since the set of relevance judgments is incomplete, the number of relevant documents between genres might not be proportionally correct. For instance, if genre A has most of its documents judged while genre B only has a small portion of its documents judged, then genre A would contain more relevant documents than genre B because unjudged documents are considered irrelevant. Therefore, the relative differences in their weights are not realistic.

A complete set of relevance judgments can be built from the incomplete set by using the methods proposed by Büttcher [BCYS07]. However, this is outside the scope of this

---

<sup>1</sup><http://www.google.com>



The image shows a screenshot of a Google search results page for the query "david beckham". At the top, there are navigation links for "Web", "Images", "Video", "News", "Maps", "Gmail", and "more". The Google logo is on the left, and the search bar contains "david beckham" with a "Search" button. To the right of the search bar are links for "Advanced Search" and "Preferences". Below the search bar, a blue bar indicates "Results 1 - 10 of about 11,700,000 for david beckham. (0.06 seconds)".

The results are categorized under "News results for david beckham - View today's top stories". The first three results are:

- [Galaxy hoping injured Beckham has "some part" in first game](#) - USA Today - 1 hour ago
- [Victoria Beckham Gets Her 15 Minutes \(Times Four\)](#) - Los Angeles Times - 54 minutes ago
- [Beckham forced to train alone](#) - Telegraph.co.uk - 17 hours ago

The next result is a Wikipedia entry: [David Beckham - Wikipedia, the free encyclopedia](#). The snippet reads: "On 26 May 2007 David Beckham was recalled to the national team for the first time since the 2006 World Cup after some excellent performances for Real Madrid ...". The URL is [en.wikipedia.org/wiki/David\\_Beckham](http://en.wikipedia.org/wiki/David_Beckham) - 193k - 17 Jul 2007 - Cached - Similar pages.

The following result is [Beckham Magazine - The Online David Beckham Fanzine](#). The snippet says: "Beckham Magazine, the Online David Beckham fanzine, website dedicated to David Beckham. Daily news, pictures, video and audio files, articles, interviews, ...". The URL is [www.beckham-magazine.com/](http://www.beckham-magazine.com/) - 4k - Cached - Similar pages.

Next is [AskMen.com - David Beckham](#). The snippet states: "AskMen.com feature on David Beckham that includes pics, pictures, biography, video, related news, vital stats, commentary, and cool facts. ...". The URL is [www.askmen.com/men/sports/30\\_david\\_beckham.html](http://www.askmen.com/men/sports/30_david_beckham.html) - 17 Jul 2007 - Similar pages.

The final result is [DBVN // David Beckham Fansite //](#). The snippet notes: "The site is best viewed with internet explorer only. marketing.". The URL is [www.davidbeckhamvn.com/](http://www.davidbeckhamvn.com/) - 3k - Cached - Similar pages.

At the bottom, there is a partial result for YouTube: [YouTube - Beckham collection](#).

Figure 4.1: Google search results for "David Beckham"

thesis.

Since the relative differences in their weights are not realistic, we need to introduce a new normalization constant,  $\alpha$ .

$$w_{G_j} = \alpha * \frac{\text{Number of relevant documents in a document genre}}{\text{Number of documents in the collection}} \quad (4.4)$$

Consider the situation where there is no difference between the relevance of document genres. Then the weight should have no effect on the document score. This results in reverting our method back to the unstructured case where  $w_{G_j} = 1$  for all  $G_j$ . Thus,  $\alpha$  is used to scale the weight of each document genre so that the sum of all  $w_{G_j}$  equals the number of defined document genres,  $N$ .

$$\sum_{j=1}^N w_{G_j} = N \quad (4.5)$$

The new normalization constant is multiplied to  $w_{Rel,G_j}$  and  $w_{Rel}$  so that  $w_{G_j}$  is proportional to  $N$ .

$$\alpha = \frac{N}{|Rel| * \sum_{j=1}^N |G_j = Rel|} \quad (4.6)$$

where  $|G_j = Rel| =$  is the number of relevant documents in genre  $G_j$ ; and  $|Rel| =$  the number of relevant documents.

Given the weight of each document genre for a specific work task, the weighted term frequency is

$$f''_{D,Q_i} = w_{G_j} * f'_{D,Q_i}. \quad (4.7)$$

#### 4.1.4 Experimental Setup — The Corpus

For our experiments, we employ the W3C collection used in the TREC 2006 Enterprise track [CSdV06]. The W3C collection contains 331,037 documents with a total uncompressed size of 5.7 gigabytes. For simplicity, our experiments categorized documents into *document types* instead of *document genres*.

Scope	Corpus Size	Avg Doc Size	# Docs
www	1.043 (gigs)	23.8 (kbs)	45,975
lists	1.855	9.8	198,394
dev	2.578	43.2	62,509
people	0.003	3.6	1,016
other	0.047	14.1	3,538
esw	0.181	9.7	19,605
<b>all</b>	<b>5.7</b>	<b>18.1</b>	<b>331,037</b>

Table 4.1: W3C collection

The W3C documents are categorized into six different types: mailing lists (lists), public CVS repository (dev), public pages (www), wiki pages (esw), personal pages (people), and other pages (other). Table 4.1 shows the number of documents in each type and their average document sizes.

We use the topics and qrels file provided by the *expert search task* in the TREC Enterprise track. A qrels file contains a list of relevant documents identified for each topic, which is used for performance evaluations. Figure 4.2 shows a sample topic and its corresponding list of relevant documents.

Topic 52	ontology engineering
qrels	52 0 dev-000-3132379 1 52 0 lists-000-0424397 1 ... 52 0 www-064-8842949 1

Figure 4.2: A sample topic and the corresponding qrels file

There were 54 different topics relating to the W3C collection. Participants were required to find an expert on each topic and retrieve supporting documents that indicate this person is an expert on the topic. Their results were then evaluated with the qrels file to determine their performances.

### 4.1.5 Limitations

Our experiments are limited by the nature of the *expert search* task in the TREC Enterprise track. There are two limitations:

- the set of relevance judgments is incomplete; and
- there is only one task behind all 54 topics.

The topics used by this task are limited in a way that the set of document judgments is incomplete. The reason is that some documents might be relevant to a topic but they do not provide the name of an expert. As a result, although these documents contain relevant information, they are not identified in the set of document judgments. On the other hand, any supporting document identified for an expert can be treated as relevant to the topic.

Since the queries were used by the *expert search* task in TREC Enterprise track, they were created with the objective of finding an expert for a particular topic. Thus, we define the specific work task for these 54 queries to be a task for finding documents that support a person as an expert regarding the topic of each query. Our objective is to determine the document type that is most likely to contain supporting documents for an expert and returning a list of documents that mostly are from this type.

### 4.1.6 The Weights

For the *expert search* task, there were 54 different topics relating to the W3C collection. We separate this set of topics into *training* and *testing* sets: 50% of the topics, along with their relevance judgments, form the *training set* and the other 50% of the topics form the *testing set*. The training set is used to calculate the weight for each document type while the testing set is used to evaluate the performance of our method. Experiments were carried out with each half of the topics taking turn being the training set and the testing set.

Table 4.2 shows the number of relevant documents in each document type for each training set. These number would be used to compute the weights of our approach.

Scope	# Docs	# Relevant Docs (Topic 52-78)	# Relevant Docs (Topic 79-105)
www	45,975	17,222	20,410
lists	198,394	28,936	31,655
dev	62,509	1,280	807
people	1,016	17	22
other	3,538	271	290
esw	19,605	1,057	633
<b>all</b>	<b>331,037</b>	<b>50,113</b>	<b>54,726</b>

Table 4.2: Relevance Judgments

Table 4.3 shows the weights of each document genre.

	(Topic 52-78)	(Topic 79-105)
Scope	$w_{G_j}$	$w_{G_j}$
www	2.118197	2.266035
lists	3.558948	3.514518
dev	0.157430	0.089597
people	0.002089	0.002443
other	0.033333	0.032197
esw	0.130002	0.070281

Table 4.3: Genre Weights

#### 4.1.7 Results

Using topics 79-105 as the training set, Figure 4.3 shows that  $P@5$  improves from 0.5615 to 0.6692 for an 19% increase over the BM25 baseline model. Using topics 52-78 as the training set, Figure 4.4 shows that  $P@5$  improves from 0.7043 to 0.7739 for a 10% increase over BM25.



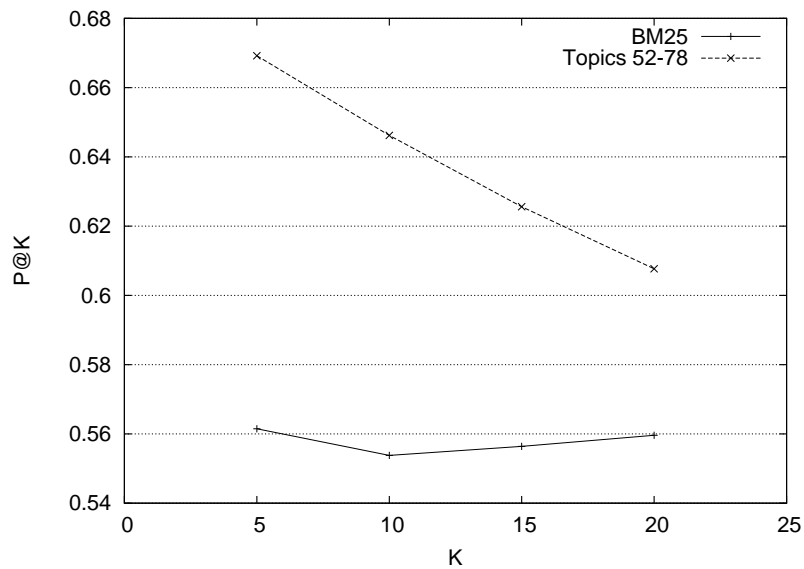


Figure 4.3: Precisions for topics 52-78, using topics 79-105 for training

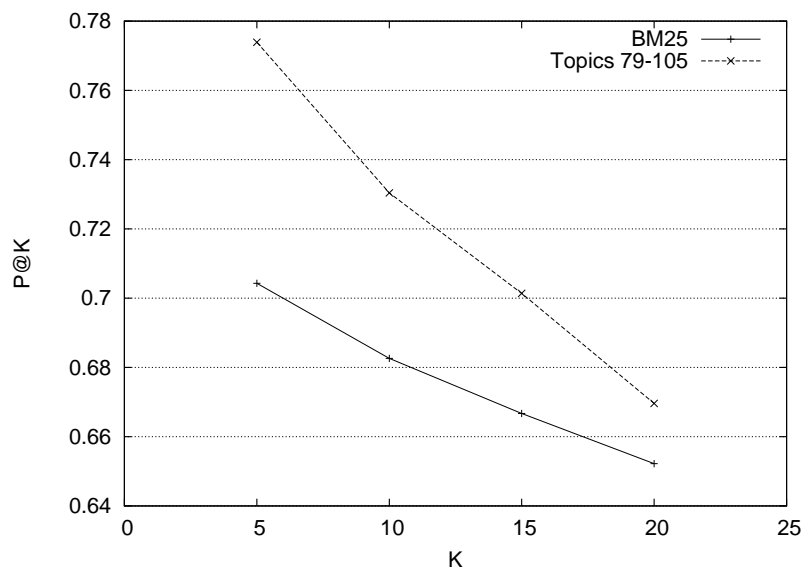


Figure 4.4: Precisions for topics 79-105, using topics 52-78 for training

We run a series of paired t-tests to determine the significance of our results. A paired t-test compares two paired sets of data. It calculates the difference between each paired set and analyzes the differences based on the assumption that the differences in the entire population follow a Gaussian distribution. The null hypothesis for our model is that it achieves worse precision at  $N$  ( $= 5, 10, 15, 20$ ) documents than the original BM25 model. We calculate the confidence level where each null hypothesis is wrong (i.e. our model performs better than BM25).

Using topics 52-78 as training data, our model is significantly better than the BM25 baseline model at  $P@5$  and  $P@10$  with a 95% confidence.

## 4.2 Weighting Document Types based on Click Frequency

One major drawback of using relevance judgments is that it requires a large amount of time and workload from users before a well-suited result page is presented to them [JFM97]. An alternative is to use implicit feedback for evaluating document relevance. Implicit feedback is information about user interaction with the system, which includes clickthrough data, viewing time of a document, exit method, etc.. A retrieval system can analyze this information and predict document relevance based on which documents were selected by the user, how long did s/he spent on it, etc.. The general belief is that, if a user selects a document and spends a good amount of time viewing it, this document has a high probability of being relevant. Fox et al. [FKM<sup>+</sup>05] examined the relationships between implicit and explicit feedback in web search and developed models to associate explicit ratings and implicit measures of user interest.

One of the most useful types of implicit feedback is clickthrough data, which is a history about user-submitted queries and user-selected documents on the corresponding search result page. Although clickthrough data do not provide a direct indication of document relevance, they provide useful hints for determining which document (or type of document) is relevant to a user's need. Many different approaches of utilizing clickthrough data to improve retrieval performance have been proposed (e.g. [ABD06, STZ05a]).

In the next approach, clickthrough data are grouped together based on different task-

genre pairs. To determine the weight for each task-genre pair, we consider the click frequency of the document genre when the work task was given. For example, given a work task, if genre A is clicked more frequently than genre B, then genre A's weight would be larger than genre B's. Depending on the document's genre and on the given work task, we apply the corresponding weight to the modified BM25 to compute the relevance score.

### 4.2.1 Pros and Cons of Clickthrough Data

Clickthrough data is an attractive commodity for understanding user preference. Clickthrough data can be obtained:

- in large quantity through search logs;
- at a low cost; and
- without putting additional burden on the users.

For these reasons, many approaches have been proposed to interpret and utilize clickthrough data.

Another advantage of using clickthrough data to estimate the weights is that clickthrough data can be obtained continuously. That is, the weights are estimated dynamically and adjusted automatically. This is important because user preference may change over time. By adjusting the weights continuously, our model is adapting to changes in user preference (document genre) for each work task.

Clickthrough data also has its disadvantages. One major disadvantage is that they do not reflect exact document relevance. They are simply user selections for work tasks on the corresponding result page. This does not, by any means, indicate whether users think the selected documents are relevant or not. For instance, a user can select multiple documents from a result list before reaching a document that is relevant. All previously selected documents can either be irrelevant or somewhat relevant. Clickthrough data are only indications on what users believe to be relevant according to the information shown on the result page (e.g. document's title, snippet, and URL).

### 4.2.2 Weighting Document Genres

To determine a realistic estimate of the weight for each document genre, we consider click frequency for each document genre and work task. Each weight should have these properties:

- the weight is one if click frequency is zero;
- the weight increases monotonically with click frequency; and
- the weight increases to an asymptotic maximum.

Given a work task, assume  $cf_{G_j}$  represents click frequency of a document genre  $G_j$ . A rough model for estimating each weight can be formulated as

$$w_{G_j} = |G| * \frac{cf_{G_j} + S}{|C| + |G|S} + 1 \quad (4.8)$$

where  $|G|$  is the number of genres,  $|C|$  is the total number of clicks, and  $S$  ( $= 1.5$ ) is a smoothing parameter.

First, if  $cf_{G_j}$  is zero, then  $w_{G_j} = 1$  (assume  $S$  is relatively small). Second, equation 4.8 is linear, thus,  $w_{G_j}$  increases monotonically as click frequency increases. Finally, if a particular document genre dominates the clicks,  $cf_{G_j}$  would almost be equaled to  $|C|$ , which means  $w_{G_j}$  would have a value close to  $|G| + 1$ . Hence, the weight increases to an asymptotic maximum. Equation 4.8 satisfies all properties listed above.

### 4.2.3 Experimental Results

For our experiments, we employ the same W3C collection used in the previous section. The limitations mentioned in the previous section also apply in these experiments. For simplicity, we classify documents into *document types* rather than *document genres*. In addition, we assume that there is only one task—expert search task—behind the queries submitted to our model.

Table 4.4 shows the number of clicks for each document genre and the estimated weight computed using Equation 4.8. The *lists* genre recorded the most number of clicks and thus,

it carries the largest weight. As a result, for the expert search task, term frequency for documents in the *lists* genre is boosted. Conversely, the *other* genre was never clicked. Hence, term frequency for its documents is unweighted and would not be altered.

Scope	$cf_G$	$w_G$
www	289	2.564632
lists	760	5.101436
dev	32	1.180431
people	9	1.056553
other	0	1.008035
esw	21	1.121185

Table 4.4: Click Frequencies and Genre Weights

Figure 4.5 shows that BM25+CF increases search precision at 5 documents from 0.6286 to 0.7469, a 19% improvement.

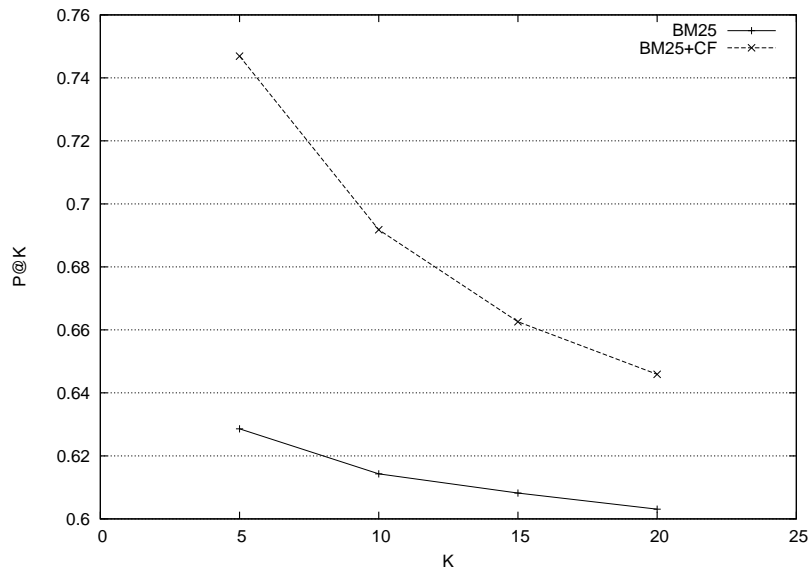


Figure 4.5: Precisions for BM25 and BM25+CF

We run a series of paired t-tests to determine the significance of our model and its results. Table 4.5 shows the p-values for our model. By setting the significance level to

0.05 ( $p = 0.05$ ), our model is statistically significant for precision at 5 and 10 documents.

	P@5	P@10	P@15	P@20
BM25+CF	<b>0.0062</b>	<b>0.0137</b>	0.061	0.0966

Table 4.5: *p-values* for BM25+CF

### 4.3 Chapter Summary

In this chapter, we showed two different approaches to learn the weight for each task-genre pair. There are two reliable resources that we can utilize to learn the weights—relevance judgments and clickthrough data. These resources are provided by users and represent their preferences. Both approaches have shown significant improvement in precision at the top of the search result list.

# Chapter 5

## X-Site

### 5.1 Chapter Overview

In the previous two chapters, we showed how each task-genre relationship can be incorporated into the document relevance scoring function, Okapi BM25, and how each relationship can be weighted using different resources provided by users. As shown in Chapter 4, search precision can be significantly improved using this approach. In this chapter, we will illustrate how these techniques can be deployed in an enterprise search environment.

In order to increase productivity, professionals in the workplace need high-precision search tools capable of retrieving information that is relevant to the task at hand. One approach to identify their search context and to make use of this context in the retrieval process.

X-Site is a contextual search engine for a workplace environment and it is currently deployed at a major technology firm. X-Site uses the relationships between work tasks and document genres to improve search precision for software engineers. Each task-genre relationship is weighted, similar to the techniques shown in the Chapter 4. This chapter first illustrates the implementation of the system and then discusses how it is related to weighting document genres.

X-Site will be demonstrated at the 30th annual international ACM SIGIR conference on Research and Development in Information Retrieval (SIGIR 2007) [YFC07].

## 5.2 The System

X-Site is an enterprise search engine for the software engineering domain that exploits relationships between the user's tasks and the document genres in the collection to improve retrieval precision. The system provides a customized and user-controlled means of refining search results to suit the task context of a user. This is beneficial in enterprise information environments, which need to serve diverse user populations and support a wide range of work and information tasks.



Figure 5.1: X-Site: Contextual Search System

X-Site is currently deployed as a prototype in a real workplace environment. It provides a single point of access to documents from the Internet, a corporate intranet, and Lotus Notes databases, which were crawled using a set of URL seeds tailored to the needs of a group of software engineers.

Figure 5.2 is a screen shot of X-Site's search interface. To search using X-Site, a user enters a query, and selects a work task and an information task from respective drop-down lists. The query is used to retrieve a set of documents from the collection. The task profile is used to determine the genre weights according to the task-genre correlation matrix. The documents are then ranked using our modified BM25 scoring model, which incorporates genre weights (See Figure 5.3).

Figure 5.3 also shows other parts of X-Site. Documents are crawled from many different resources and are transformed into a unified format (XML). These documents are then classified into different genre(s) and merged into a single index. Given a search query and a set of genre weights, X-Site only needs to refer to one single index and be able to retrieve



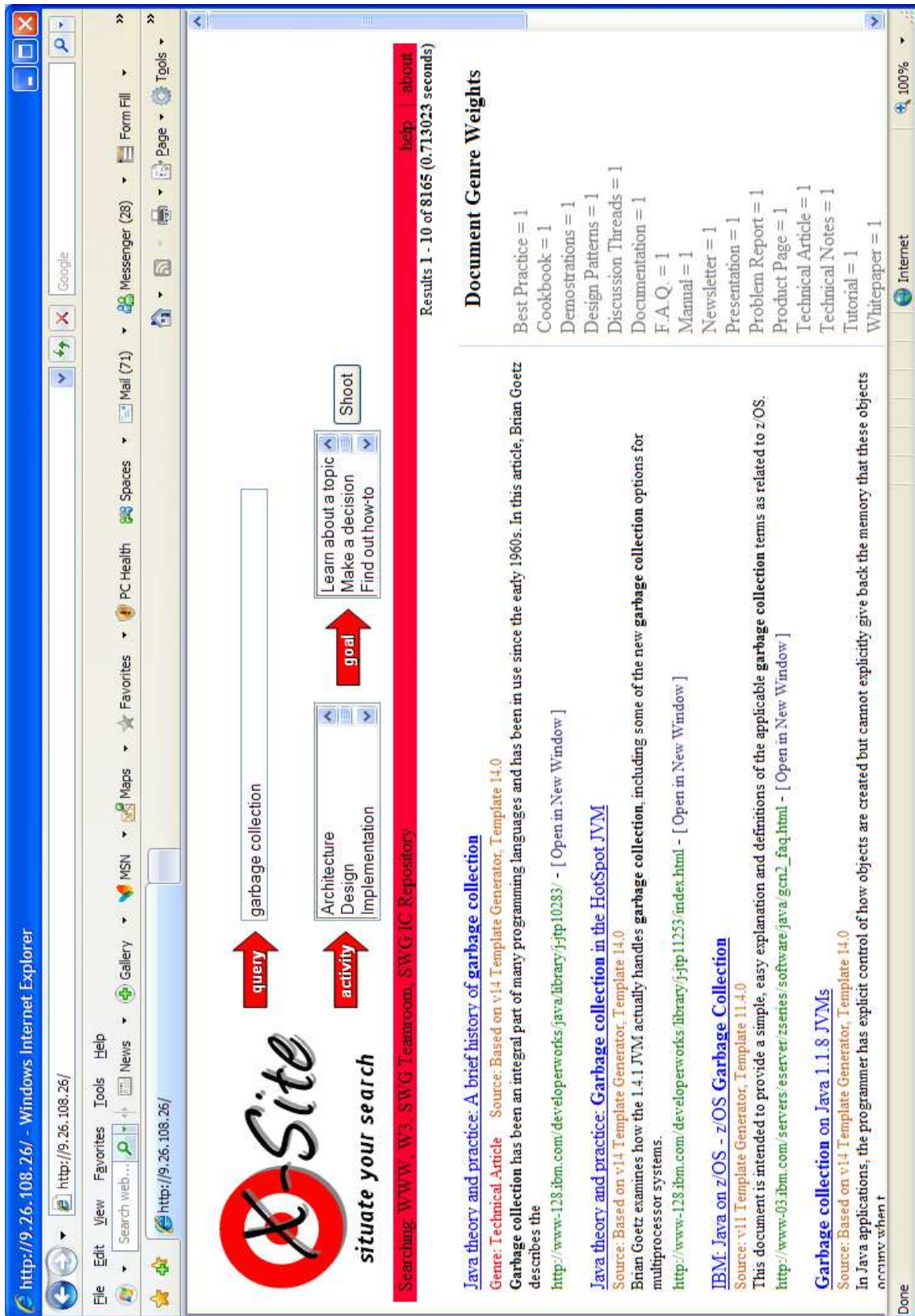


Figure 5.2: Screen shot of X-Site

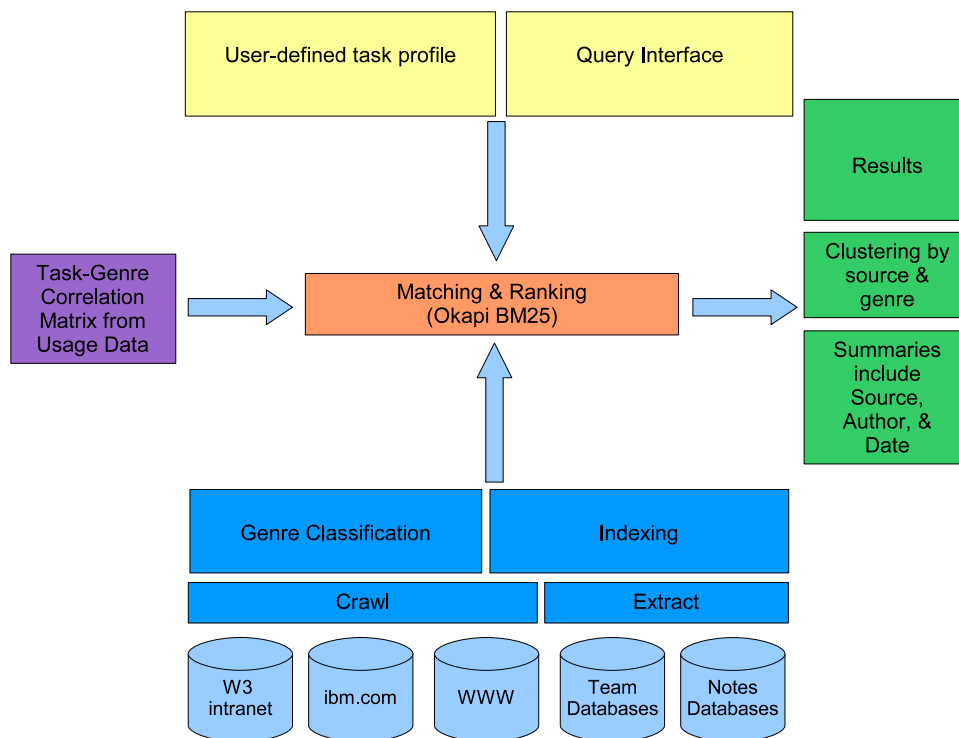


Figure 5.3: Features and Components of X-Site

relevant documents from multiple sources.

### 5.3 Project Motivation

Information sharing has become a major part of a corporation's daily operation. Employees are encouraged to share their knowledge or technical skills through a corporate domain (i.e., a corporate intranet) so that other employees can benefit from their contributions. Since more and more information are being shared and the sizes of corporate intranets are growing rapidly, there is an increasing need to improve search quality for corporate intranets.

One of the problems evolved from information sharing is that existing search tools are no longer capable of performing high quality searches for those rapidly-growing collections. In other words, search tools have not kept pace with the rapid growth of shared information and have become outdated. For instance, it is more difficult to determine which documents, in a collection, are relevant to a search request and, more importantly, which document is most relevant. As a result, search quality has been diminished. Information sharing has introduced concerns over search efficiency, quality of service, and employee satisfaction.

Previous research [FS03] has shown that many employees spent significant amount of their working hours on searching for information relating to their work tasks. The reason is that about 50% of workplace searches failed, which force employees to spend extra effort on refining their search. Therefore, information sharing have introduced other problems for corporations before their employees can enjoy its benefits.

Another problem in enterprise search is that there are many sources for information within a corporation. For a global organization like IBM, there are intranet pages from Asia, Europe, and North America. In addition, there are many different information repositories, web sites, and databases that are used on a regular basis. Information searching would involve many search processes, one for each source. It becomes a complex task to keep track of these repositories, to decide which one to use, and to be able to search effectively within each. In some cases, employees opt to use a general purpose search engine such as Google, which does not search any IBM internal assets.

*Goal of this project: to provide a single point access that searches for the **right***

information *quickly* in an ever-growing corporate data collection.

## 5.4 Anticipated Benefits

Abrol et al. [ALM<sup>+</sup>01] proposed a “business portal” as an ideal solution to all enterprise search problems. Their solution has the following characteristics:

- the need to access information in diverse repositories including HTTP web pages, corporate intranet pages, and Lotus Notes databases;
- the need to respect fine-grained individual access-control rights, typically at the document level;
- the need to index and search a large variety of document types, such as PDFs, Microsoft Word and PowerPoint files, and different languages (such as English, French, Chinese, etc.); and
- the need to seamlessly and scalably combine structured, as well as unstructured, information in a document for search, personalization, and organizational purposes (i.e., clustering, classification, etc.).

X-Site satisfies three of these characteristics (1st, 3rd, and 4th). The only characteristic (2nd) that it does not satisfy can easily be implemented with some modifications.

X-Site converts documents from different types into a single indexable format so that these documents are accessible. These types include PDF, PostScript, Microsoft Word, Excel, and PowerPoint files. In addition, X-Site also provides access to documents from Lotus Notes databases. Finally, X-Site uses a language identifier to differentiate documents in English and other languages. Therefore, X-Site provides a single point access to different repositories and databases.

X-Site uses weighting to combine structured information within a document in the retrieval process. When evaluating the relevance of a document, X-Site computes a weighted term frequency for each query term and calculates a relevance score based on the weighted term frequency. It can assign reasonable weights to each document structure and to each genre classification.

The only characteristic that X-Site does not satisfy is to respect access-control rights for documents. Since X-Site uses a file system search engine, Wumpus, this characteristic can be implemented. Wumpus is meant to be run on a UNIX file system and thus, it supports the security mechanism in UNIX. For instance, a file can be read, write, or execute by its owner or group of owners. Each file can be owned by an individual user, a group of users, or everyone. This security mechanism can also simulate the structure of a corporation. That is, each file can be accessed by an employee, a team of employees, or any employee in the corporation. X-Site can simply use this security mechanism in file system search to reflect the access-control rights for documents in a collection. This characteristic is one of the future enhancements for this project.

## 5.5 The Components

The X-Site concept is based on a domain analysis of the information practices among a community of software engineers in a major technology firm, which identified a strong relationship between the tasks they perform and the document genres they use [FTC05]. This analysis enabled us to identify task-dependent patterns of genre preference, which we incorporated into the ranking algorithm of X-Site. X-Site includes the following contextual components:

- a **genre classifier**, which uses supervised machine learning methods ( $SVM^{light}$ <sup>1</sup>) and textual features to pre-process and tag the document collection by genre;
- a **language identifier**, which uses N-gram-based text categorization (libTextCat<sup>2</sup>);
- a **document type converter**, which converts different document types into an indexable format;
- a mechanism for recording **clickthrough data**; and

---

<sup>1</sup><http://svmlight.joachims.org>

<sup>2</sup><http://software.wise-guys.nl/libtextcat>

- a multi-user **search engine** (Wumpus<sup>3</sup>), which incorporates the weights for task-genre relationships in its retrieval procedure.

### 5.5.1 Genre Classifier

Each document in the collection was classified into genre(s) based on its text and content. We used a text classification tool—Support Vector Machines [Vap95]—to determine which genre(s) a document belongs to. Support Vector Machines are a generalized linear classifier and can be used to classify any textual document.

Support Vector Machines have been shown to perform well in comparison with other classification methods, such as Naïve Bayes, C4.5 decision trees [DVDM01] and neural networks [ES04], and are successful in text classification [Joa98].

The X-Site system uses the  $SVM^{light}$ <sup>4</sup> implementation of Support Vector Machines that was provided by Joachims from Cornell University.

Table 5.1 shows all genres defined in our study [Fre07].

Figure 3.1 is an example of a *cookbook* document and Figure 3.2 is a F.A.Q. document. The genre identified for a document is indicated within the *CLASS* tag.

### 5.5.2 Language Identifier

In enterprise search, one of its main challenges is to identify the language in which a document is written in [Haw04]. Different languages have different structures and patterns. Since  $SVM^{light}$  classifies documents based on the text and patterns within documents, documents written in different languages can confuse and create problems for the genre classifier. Therefore, we made use of *Libtextcat* as our language identifier.

Libtextcat is a library that implements the classification technique described in Cavnar & Trenkle [CT94]. It was primarily developed for language guessing and it can be performed with a high accuracy. The intuition for this classification technique is to calculate a “fingerprint” of a document, whose category is unknown. A fingerprint is a list of the most frequent N-grams within a document, ordered by frequency. The unknown fingerprint is

---

<sup>3</sup><http://www.wumpus-search.org>

<sup>4</sup><http://svmlight.joachims.org>

<b>Document Genre</b>	<b>Description</b>
Best Practice	description of a proven methodology or technique for achieving a desired result, often based on practical experience
Cookbook	step-by-step description of how to implement a technology
Demonstration	automated presentations of products or solutions, usually in multimedia format
Design Pattern	description of a standard solution to a common problem in software design
Discussion Thread	brief, informal posting to a discussion group or forum, usually concerned with technical issues
Documentation	a reference program that contains basic descriptions and instructions on how to use a software program
Engagement Summary	a report describing a particular consulting project and the nature of the services provided
FAQ	Question and answer pairs that provide basic information on specific technical or product issues
Manual	book-length document containing practical instructions, rules, and/or steps for performing a task or using a technology
Presentation	the charts used to accompany a talk or class lecture
Problem Report	record of a reported technical problem together with the details of the customer-support interaction and the steps taken by technical support to resolve it
Product Page	a web page designed to provide basic information and links with respect to a particular software product
Technical Article	a formal essay or report written about a technical subject for publication in a journal or periodical
Technote	brief and informal documents published to share useful technical and product-related information
Tutorial	step-by-step lessons usually including sample code designed for self-study
Whitepaper	an authoritative report on a topic in technology containing advanced technical details and guidance

Table 5.1: Categories for Document Genres in X-Site

compared with fingerprints of documents, whose categories are known. Fingerprints are compared with a simple out-of-place metric. The categories of the closest matches are output as the classification.

The language identifier processes over 100 documents per second on a simple PC, which makes it practical for many uses.

### 5.5.3 Document Type Converter

X-Site converts documents from different types into an indexable format so that documents from a variety of types can be searched. These document types include PDFs, PostScripts, Microsoft Word, Excel, and PowerPoint files.

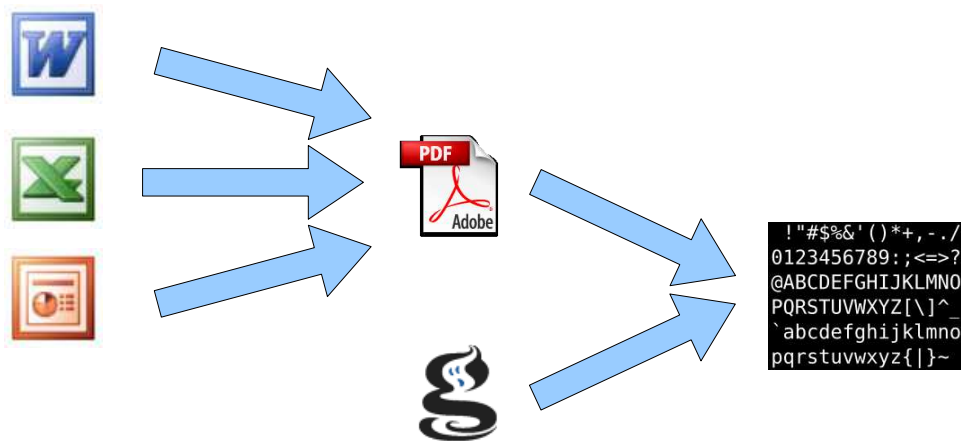


Figure 5.4: Converting Different Document Types into ASCII text

First, X-Site uses a third-party tool, Open Office<sup>5</sup>, to convert any Microsoft Office files to PDF files. A UNIX shell script was written to use Open Office commands to automatically convert documents into PDFs. Then, these PDF files, along with some original PDFs, are converted into plain text files using another third-party tool, Xpdf's pdftotext<sup>6</sup>. For any PostScript files, they are converted into plain text files using Ghostview's ps2ascii<sup>7</sup>.

<sup>5</sup><http://openoffice.org>

<sup>6</sup><http://www.foolabs.com/xpdf/download.html>

<sup>7</sup><http://www.cs.wisc.edu/~ghost/>



Finally, these plain text files are indexed by the search engine and are available to any search request (See Figure 5.4).

#### 5.5.4 Clickthrough Data Collection

One of the proposed methods for estimating the weight for each task-genre relationship is to use clickthrough data. Clickthrough data are collected when a user clicks on a document on the search result page. They need to be coupled with the query that a user submits to the system. Hence, each click is matched with its corresponding query and it is an indication of document relevance with respect to the query.

X-Site records all search requests submitted by users. For each search request, it includes an *activity task*, an *information goal*, and a *query* of terms. X-Site also records additional information (such as date, time, user information, etc.) so that it can identify each search session. Figure 5.5 shows some samples of recorded queries from a log file.

Date	Time	IP Address	Query ID	Activity	Goal	Query
Nov 9	13:55:42	9.26.109.26	562542-2745	installation	howto	MQ Workflow
Nov 9	14:17:59	9.26.109.26	563879-2387			dynamo migration
Nov 9	14:19:39	9.26.109.38	563979-2746			cobol application
...						

Figure 5.5: A query log file

X-Site also records all click-on documents by its users. In order to record these clicks, all documents on the result page have URLs that send users back to X-Site. X-Site then records all required information about this click. Finally, X-Site redirects the users to the documents that they clicked on. Figure 5.6 illustrates the procedure between the time a user clicks on a document and the time s/he gets to view it. This procedure does not cause any noticeable delay or interruption between the users and X-Site.

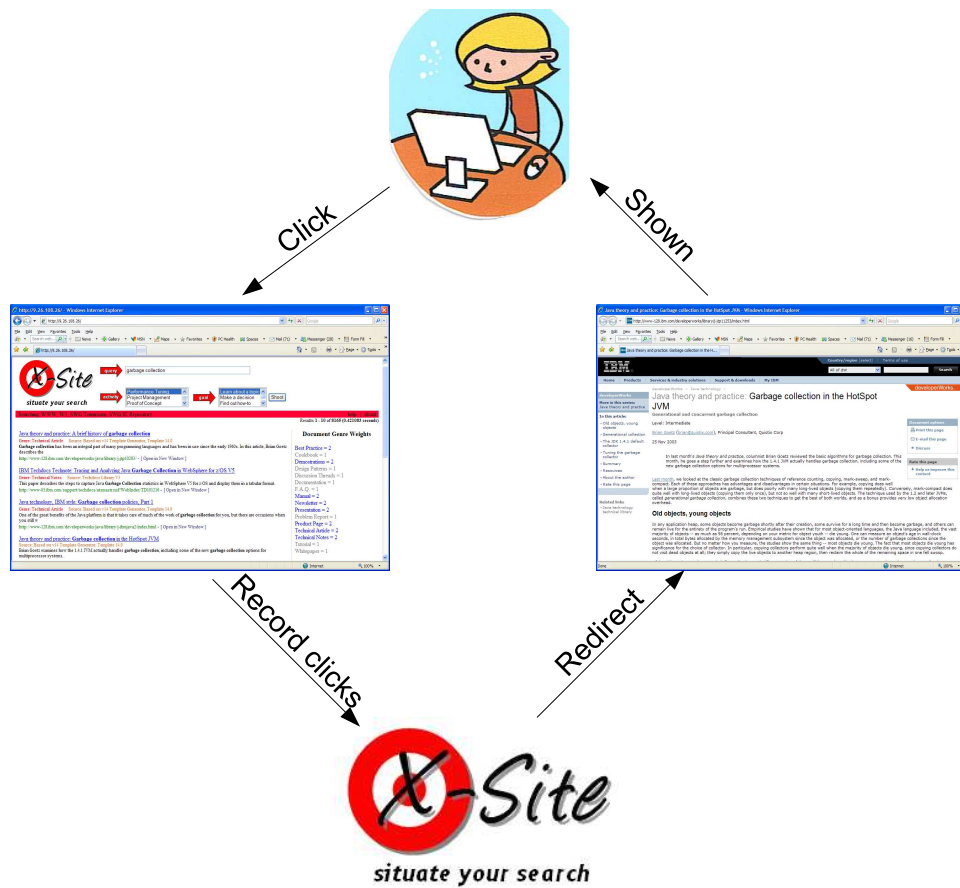


Figure 5.6: Collecting clickthrough data

Figure 5.7 shows samples of recorded clicks from a log file. For each click, the *URL* and *rank* on the result page are recorded. In addition, the date, time, and user information are also recorded. For each recorded query and click, there is a unique *QueryID* assigned to each query submitted to X-Site. This *QueryID* is used to match each click with its corresponding query.

Date	Time	IP Address	QueryID	URL	Rank
Nov 9	14:10:58	9.26.109.26	562542-2745	http://www-128.ibm.com...	15
Nov 9	14:18:02	9.26.109.26	563879-2387	notes://ATE06DB/a_dir/...	1
Nov 9	14:18:26	9.26.109.26	563879-2387	notes://ATE06DB/a_dir/...	4
...					

Figure 5.7: A clickthrough data log file

In addition to providing hints on which document is relevant to a search request, clickthrough data also provides hints on which document is irrelevant. For instance, if a user skipped a document on the result page and clicked on the next document, there is a great chance that the skipped document is irrelevant. Based on the information (e.g., a document's title, genre class, URL, and a short summary) provided on the result page, the user decided that the document is irrelevant or less relevant to the document that was clicked on. Joachims et al. [JGP<sup>+</sup>05] introduced a set of strategies for interpreting clickthrough data to determine document relevance and irrelevance.

Figure 5.8 shows the result page of Figure 5.2 recorded in a log file. Each result page is stored in XML format and is enclosed by the `<PAGE>` and `</PAGE>` tags. Each document is enclosed by the `<RESULT>` tags and it contains its ranking of the result page, title, snippet that was shown on the result page, and URL. If a document is classified into a genre, it is also recorded. By recording these information about the result page, X-Site can see what was shown to the user and understand the reason behind any user behaviour.

```

<PAGE><QUERYID> 562542-2745 </QUERYID>
<RESULT><RANK>1</RANK>
<TITLE> Java theory and practice: A brief history of garbage collection
</TITLE>
<CLASS> Technical Article </CLASS>
<SNIPPET> Garbage collection has been an integral part of many programming
languages and has been in use since the early 1960s In this article Brian Goetz
describes the </SNIPPET>
<URL>      http://www-128.ibm.com/developerworks/java/library/j-jtp10283/
</URL>
</RESULT>
<RESULT><RANK>2</RANK>
<TITLE> Java theory and practice: Garbage collection in the HotSpot JVM
</TITLE>
<SNIPPET> Brian Goetz examines how the 1.4.1 JVM actually handles garbage
collection, including some of the new garbage collection options for multiprocessor
systems. </SNIPPET>
<URL> http://www-128.ibm.com/developerworks/library/j-jtp11253/index.html
</URL>
</RESULT>
<RESULT><RANK>3</RANK>
<TITLE> IBM: Java on z/OS - z/OS Garbage Collection </TITLE>
<SNIPPET> This document is intended to provide a simple, easy explanation
and definitions of the applicable garbage collection terms as related to z/OS.
</SNIPPET>
<URL> http://www-03.ibm.com/servers/eserver/zseries/software/java/gcn2_faq.html
</URL>
</RESULT>
...
</PAGE>
...

```

Figure 5.8: A result page log file

### 5.5.5 Wumpus Search Engine

Wumpus is an open source search system. It was developed with the intent to study research problems that arise in the context of indexing dynamic text collections in multi-user environments. One particular scenario is *file system search* (aka *desktop search*), in which the underlying text collection is dynamic and requires a flexible indexing mechanism.



Figure 5.9: Wumpus: File System Search

There are two main usages of Wumpus. It can be used

- as an ordinary information retrieval system with multi-user support; and
- as an indexing service that automatically keeps tracks of all changes in the collection and updates the index accordingly.

Wumpus provides all of the features that our study requires. For enterprise search, many users are simultaneously performing different search operations on a retrieval system. The system must be able to handle all user requests efficiently. In addition, the data collection is continuously being updated with changes to existing documents or with new documents added. The indexing mechanism must be able to support all changes to the data collection and update the index accordingly. For these reasons, we chose to deploy our system using Wumpus.

In order to employ Wumpus to serve our purpose, some modifications were made. The following components were added in Wumpus:

- a **field weighting** component in Okapi BM25;
- a **task profile**, composed of a work task (e.g. installation) and an information task (e.g. find facts), which are elicited from the searcher at query time;

- a **task-genre association matrix**, which specifies known positive, neutral and negative relationships between task and genre pairs; and
- a **genre weighting** component in Okapi BM25.

### Field Weighting in Okapi BM25

As mentioned in Chapter 3, Robertson et al. [RZT04] introduced a modified version of BM25, Okapi BM25F, for weighting different fields of a structured document and calculating its relevance score by a *linear combination of the scores* for all fields.

For X-Site, these fields include **title**, **abstract**, **keywords**, and **anchor text**. For each document in the IBM corporate domain, it has one title and it is used to describe its content. An abstract is a short description of the content of a document. Keywords are a group of terms that is related to the document. Both an abstract and keywords are not required for each document. Finally, anchor text is a string of terms that appears on the links pointing to the document.

The weights for these fields are set to as follow:

Field	Weight
title	10
abstract	5
keywords	2
anchor text	0.5

Table 5.2: Field weights of a structured document.

Figure 5.10 shows the metadata of a web document. Consider “garbage collection” as the search query, this document was ranked 5<sup>th</sup> by X-Site. However, if the fields of each document were unweighted, this document would be ranked 18<sup>th</sup>. The reason is that since the query appears in both the **abstract** and **keywords** fields, their term frequencies were boosted and thus, the relevance score for this document was increased.

```

<html>
<head>
<meta content="text/html; charset=UTF-8" http-equiv="Content-Type" />
<title>Eye on performance: Referencing objects</title>
<meta content="(PICS-1.1 http://www.icra.org/ratingsv02.html http-
equiv="PICS-Label" />
<link href="http://purl.org/DC/elements/1.0/" rel="schema.DC" />
<link href="http://www.ibm.com/favicon.ico" rel="SHORTCUT ICON" />
<meta content="dW Information/Raleigh/IBM" name="Owner" />
<meta content="en-US" scheme="rfc1766" name="DC.Language" />
<meta content="ZZ" name="IBM.Country" />
<meta content="Public" name="Security" />
<meta name="Abstract" content="Intrepid optimizers Jack Shirazi and
Kirk Pepperdine set their sights on the Java Games Web site to see how
game developers identify and then resolve problems that appear when
their application doesn't release objects for garbage collection." />
<meta name="Description" content="Intrepid optimizers Jack Shirazi and Kirk
Pepperdine set their sights on the Java Games Web site to see how game developers
identify and then resolve problems that appear when their application doesn't release
objects for garbage collection." />
<meta name="Keywords" content="Java Performance, kirk pepperdine,
jack shirazi, garbage collection, garbage collector, allocation, games, java
programming, tttjca" />
<meta name="DC.Date" scheme="iso8601" content="2003-08-26" />
<meta name="Source" content="Based on v14 Template Generator, Template
14.0" />
<meta name="DC.Rights" content="Copyright (c) 2003 by IBM Corporation" />
<meta name="Robots" content="index, follow" />
<meta name="IBM.Effective" scheme="W3CDTF" content="2003-08-26" />
</head>
<body>
...
</body>
</html>

```

Figure 5.10: Metadata of a document

## Task Profiler

Behind each search within a corporate domain, there is a work task related to the search. For instance, if an employee wishes to acquire knowledge on a particular work task that s/he is working on, it is likely that there exists some related documents in the corporate domain. The employee can search for these documents and acquire the necessary knowledge from them. On the other hand, if an employee is searching for information that is not related to any work task, or not related to their work, it is unlikely that this information exists in the corporate domain. In fact, this information would most likely to appear on the Internet and searching through a public search engine (e.g., Google) might be a better choice.

If more information about a search request is given to a retrieval system, then the system can make a more accurate estimation on document relevance for the request. Hence, it can provide a more tailor-made result page to the user. As a result, when a task profile is given to X-Site, retrieval performance can be improved. The task profile used in X-Site consists of two elements: work task and information goal.

Table 5.3 shows a list of work tasks that this project has identified [Fre07]. A work task is the task that a user has at hand, which is related to his/her work responsibilities. For example, if a software engineer wishes to understand the architecture of a software product, then s/he would select *Architecture* in the task profile. X-Site would then return documents that are targeted to show the components of a product.

Table 5.4 shows a list of information goals that this project has identified [Fre07]. Aside from understanding a user's work task, it is equally important to know what the user is trying to accomplish. For example, if the same user is trying to learn about the architecture of a software product, s/he would select *Architecture* and *Learn a Topic* in the task profile. X-Site can then understand that the user is unfamiliar with the project and provide documents that show a general orientation on the components of a product. On the other hand, if the user select *Architecture* and *Find facts*, then s/he is familiar with the topic but wishes to obtain some specific factual information about the product (i.e., s/he wishes to understand the dependence of each API in the system).



<b>Activities</b>	<b>Descriptions</b>
Architecture	To determine the components of a computer system and the way they interact with one another
Design	To research, design, and specify the logical function of an application or process
Implementation	To build the target program
Deployment	To place an application in a distributed environment and make the application available for use
Installation & Configuration	To setup software running on a system and adjust the software settings
Integration	To make separate software/hardware systems and devices function together
Migration	To move a system from one product or technology to another
Performance Tuning	To test and adjust the system to increase its processing speed, load and reliability
Troubleshooting	To use strategies to define and solve problems encountered during the use of computer systems
Project Management	To plan and coordinate a project with the aim of meeting requirements and ensure timely completion, within cost and quality standards
Proof of Concept	To design, implement and demonstrate a working software solution to a business process or problem

Table 5.3: Categories for Work Tasks in X-Site's Task Profile

<b>Information Goals</b>	<b>Descriptions</b>
Learn a Topic	To learn about an unfamiliar topic: to seek general orientation and understanding of concepts
Make a decision	To make a decision: identification and comparison of alternatives in order to determine a course of action or develop a best practice
How-To	To find a procedure or work plan identification of the steps to take and issues that are involved
Find facts	To find specific factual information about products or technologies, for example: APIs, parameter values, supported software
Find a solution	To solve a problem or fix a malfunction; finding information on similar scenarios, problems, bugs and solutions

Table 5.4: Categories for Information Goals in X-Site's Task Profile

### Task-Genre Association Matrix

One of the main intuitions behind X-Site is that there exists relationships between work tasks and document genres [FTC05]. Given a work task, a weight (strength of the task-genre relationship) can be determined for each genre. As a result, a set of weights are obtained and can be used in the retrieval process. When a list of work tasks is defined by the system, a matrix of task-genre weights is essentially constructed by joining all possible combinations of work tasks and document genres (See Table 5.5).

For this association matrix, each element represents the weight for a specific work task and a particular document genre. When a user specifies a work task, X-Site determines the appropriate weight by looking at the association matrix, with respect to the document genre. After the corresponding weight is obtained, it can be applied to our modified Okapi BM25 (Equations 3.1 & 3.3 in Chapter 3) to compute a weighted term frequency, which is then used to determine the document's relevance.

### Genre Weighting in Okapi BM25

Document genre is a weighted structure that is used in combination with term frequency to score documents. The weight is a representation of the strength of each task-genre relationship [YCB07]. For instance, given a work task, X-Site identifies a list of related document genres, which have weights greater than 1, from the association matrix. As described in Chapter 3, these weights are used as a *linear combination of term frequencies*, which then computes a relevance score for each document.

The next section illustrates how the resulting documents could be re-ranked when the user modifies her/his search context. By identifying different work tasks, documents on the result page are ordered depending on the genre(s) that they belong to.

	Best Practice	Cookbook	Demonstration	Design Pattern	Discussion Thread	Documentation	F.A.Q.
Administration/Install & Find facts	1	5	1	1	1	5	1
Administration/Install & How-To	5	5	5	1	1	5	1
Administration/Install & Learn a Topic	1	5	5	1	1	5	1
Administration/Install & Make a Decision	5	5	5	1	5	5	1
Administration/Install & Solve a Problem	1	5	1	1	5	5	5
Architecture & Find facts	1	1	1	5	1	5	1
Architecture & How-To	5	1	1	5	1	5	1
Architecture & Learn a Topic	5	1	1	5	1	1	1
Architecture & Make a Decision	5	1	1	5	5	1	1
Architecture & Solve a Problem	5	1	1	5	5	5	5
Deployment & Learn a Topic	5	5	5	1	1	5	1
Design & Make a Decision	5	5	1	1	1	1	1
Implementation & Solve a Problem	5	1	1	1	5	5	5

Table 5.5: Task-Genre Association Matrix in X-Site

## 5.6 A Walk-through of the Search Process

This section demonstrates a brief walk-through of the search process with X-Site and how a user can refine her/his search request without modifying the search query. All needs to be done is to modify the search context. Documents on the result list would be re-ordered based on the genres that they were classified into.

Consider a case that a user is trying to search for documents related to “*Garbage Collection*”. Without identifying any *work activity* and *information goal*, X-Site returns a list of documents based on Okapi BM25 scores with any weight on term frequencies. In other words, the task-genre relationships are not implied in this situation. Figure 5.11 shows the documents returned on the result page. As shown on the right side of the result page, all genres have the same weight of 1. The top 4 documents are:

1. Java theory and practice: A brief history of garbage collection  
(*Genre: Technical Article*)
2. Java theory and practice: Garbage collection in the HotSpot JVM
3. IBM: Java on z/OS Garbage Collection
4. Garbage collection on Java 1.1.8 JVMs

Only the first document is classified into a genre, *Technical Article*, while the next three documents are not recognized as any defined genre.

The screenshot shows the X-Site search interface in an Internet Explorer browser window. The address bar shows the URL <http://9.26.108.26/>. The search bar contains the query "garbage collection". Below the search bar, there are two dropdown menus: "Architecture Design Implementation" and "Learn about a topic Make a decision Find out how-to". A red arrow labeled "query" points to the search bar, and another red arrow labeled "activity" points to the dropdown menus. A red arrow labeled "goal" points to the "Shoot" button. The search results are displayed in a red bar at the bottom of the page, showing "Results 1 - 10 of 8165 (0.713023 seconds)". The first result is "Document Genre Weights" with a list of items and their weights:

- Best Practice = 1
- Cookbook = 1
- Demonstrations = 1
- Design Patterns = 1
- Discussion Threads = 1
- Documentation = 1
- F.A.Q. = 1
- Mammal = 1
- Newsletter = 1
- Presentation = 1
- Problem Report = 1
- Product Page = 1
- Technical Article = 1
- Technical Notes = 1
- Tutorial = 1
- Whitepaper = 1

The search results are displayed in a red bar at the bottom of the page, showing "Results 1 - 10 of 8165 (0.713023 seconds)". The first result is "Document Genre Weights" with a list of items and their weights:

**Document Genre Weights**

- Best Practice = 1
- Cookbook = 1
- Demonstrations = 1
- Design Patterns = 1
- Discussion Threads = 1
- Documentation = 1
- F.A.Q. = 1
- Mammal = 1
- Newsletter = 1
- Presentation = 1
- Problem Report = 1
- Product Page = 1
- Technical Article = 1
- Technical Notes = 1
- Tutorial = 1
- Whitepaper = 1

The search results are displayed in a red bar at the bottom of the page, showing "Results 1 - 10 of 8165 (0.713023 seconds)". The first result is "Document Genre Weights" with a list of items and their weights:

**Document Genre Weights**

- Best Practice = 1
- Cookbook = 1
- Demonstrations = 1
- Design Patterns = 1
- Discussion Threads = 1
- Documentation = 1
- F.A.Q. = 1
- Mammal = 1
- Newsletter = 1
- Presentation = 1
- Problem Report = 1
- Product Page = 1
- Technical Article = 1
- Technical Notes = 1
- Tutorial = 1
- Whitepaper = 1

Figure 5.11: X-Site: Basic Query with no work task identified

Consider now the user decides to specify **Architecture** and **Learn a Topic** as *activity task* and *information goal* respectively for her/his search request. According to the task-genre association matrix, *Best Practice*, *Design Patterns*, *Manual*, *Newsletter*, *Presentation*, *Product Page*, *Technical Article*, and *Whitepaper* are given more weights (2) than other genres. As a result, documents are re-ordered on the result page because the task-genre weights change their relevance scores.

The top document remains at the top because it belongs to the *Technical Article* genre and this genre is given a weight of 2. Two documents moved into the top four ranking because they belong to *Technical Article* as well. Thus, instead of having only one *Technical Article* document in the top four, the result page moves two more documents because of the task-genre weights.

Finally, ‘Java theory and practice: Garbage collection in the HotSpot JVM’ moved from second to third because it does not belong to any weighted genre and thus, its term frequencies produced a lower relevance score. This is important because if X-Site uses the filtering approach for task-genre relationships, this document would be eliminated from the result page. Instead, X-Site takes the weighting approach, which give the document a chance to appear on the result list. As shown in Figure 5.12, this document is indeed relevant to the search request.

The new top 4 documents are (also see Figure 5.13):

1. Java theory and practice: A brief history of garbage collection  
(*Genre: Technical Article*)
2. Java technology, IBM style: Garbage collection policies, Part 1  
(*Genre: Technical Article*)
3. Java theory and practice: Garbage collection in the HotSpot JVM
4. Java theory and practice: good housekeeping practices  
(*Genre: Technical Article*)

```

<DOC>
<DOCNO> X1124895729-00034-35473 </DOCNO>
<URL> http://www-128.ibm.com/developerworks/library/j-jtp11253 </URL>
<TITLE> Java theory and practice: Garbage collection in the HotSpot JVM
</TITLE>
<SOURCE> Based on v14 Template Generator, Template 14.0 </SOURCE>
<BODY>
...
Old objects, young objects

In any application heap, some objects become garbage shortly after their creation,
some survive for a long time and then become garbage, and others can remain live
for the entirety of the program's run. Empirical studies have shown that for most
object-oriented languages, the Java language included, the vast majority of objects –
as much as 98 percent, depending on your metric for object youth – die young. One
can measure an object's age in wall-clock seconds, in total bytes allocated by the
memory management subsystem since the object was allocated, or the number of
garbage collections since the object was allocated. But no matter how you measure,
the studies show the same thing – most objects die young. The fact that most
objects die young has significance for the choice of collector. In particular, copying
collectors perform quite well when the majority of objects die young, since copying
collectors do not visit dead objects at all; they simply copy the live objects to another
heap region, then reclaim the whole of the remaining space in one fell swoop.

Of the objects that survive past their first collection, a significant portion of those
will become long-lived or permanent. The various garbage collection strategies per-
form very differently depending on the mix of short-lived and long-lived objects.
Copying collectors work very well when most objects die young, because objects
that die young never need to be copied at all. However, the copying collector deals
poorly with long-lived objects, repeatedly copying them back and forth from one
semi-space to another. Conversely, mark-compact collectors do very well with long-
lived objects, because long-lived objects tend to accumulate at the bottom of the
heap and then do not need to be copied again. Mark-sweep and mark-compact col-
lectors, however, expend considerably more effort examining dead objects, because
they must examine every object in the heap during the sweep phase...
</BODY></DOC>

```

Figure 5.12: A non-classified document



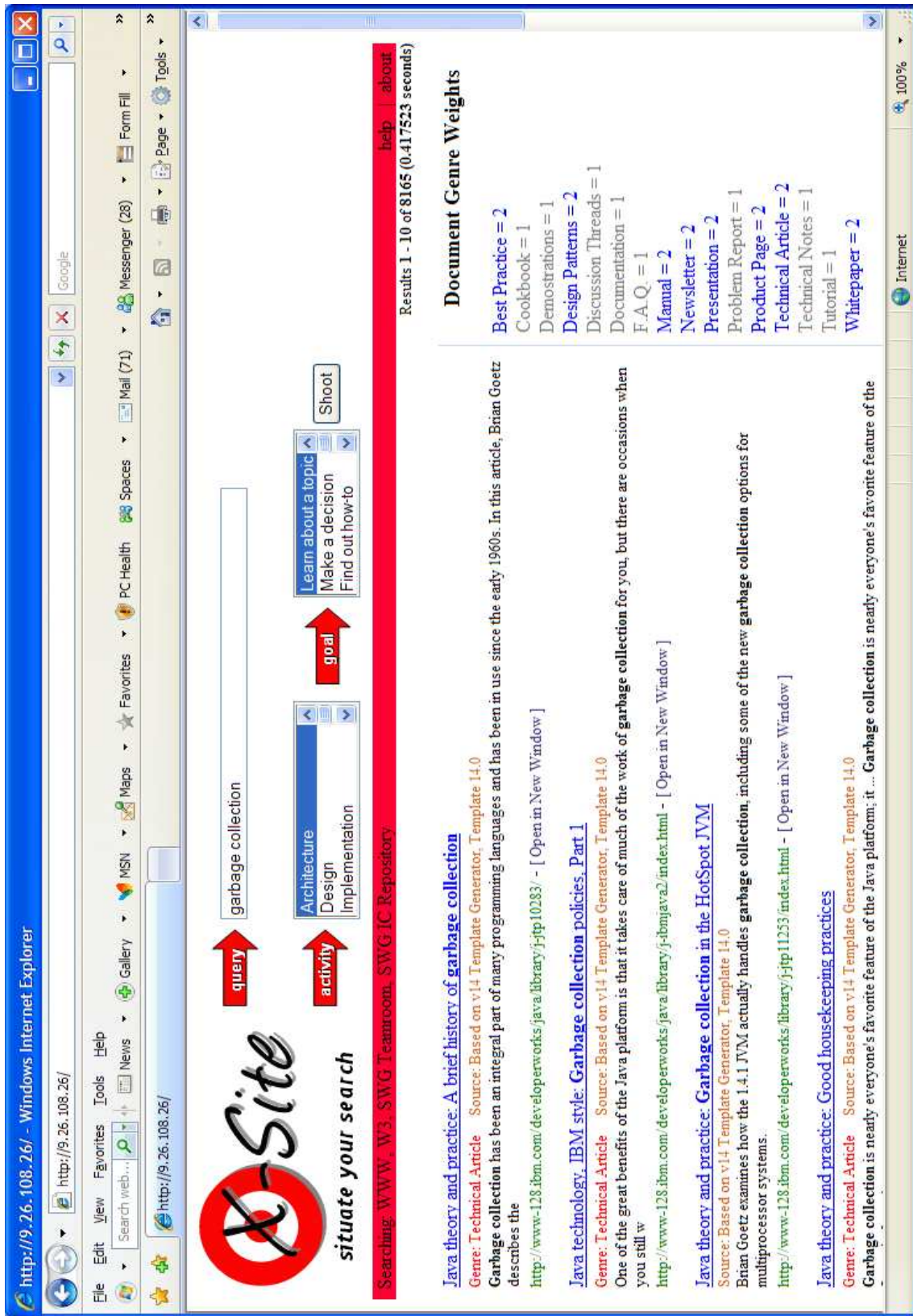


Figure 5.13: X-Site: Architecture and Learn a Topic

Now, assume the user has learned the architecture of garbage collection and wishes to learn about performance tuning with garbage collection. By changing the *activity task* to **Performance Tuning**, documents on the result page are re-ordered.

For **Performance Tuning** and **Learn a Topic**, the following genres are given more weights than other genres:

- Best Practice,
- Demonstrations,
- Manual,
- Newsletter,
- Presentation,
- Product Page,
- Technical Article, and
- *Technical Notes*.

*Technical Notes* is now given more weight. As a result, a *Technical Notes* document appears second in the result list.

1. Java theory and practice: A brief history of garbage collection  
(*Genre: Technical Article*)
2. IBM Techdocs Technote: Tracing and Analyzing Java Garbage Collection in Web-sphere for z/OS V5  
(*Genre: Technical Notes*)
3. Java technology, IBM Style: Garbage collection policies, Part 1  
(*Genre: Technical Article*)
4. Java theory and practice: Garbage collection in the HotSpot JVM

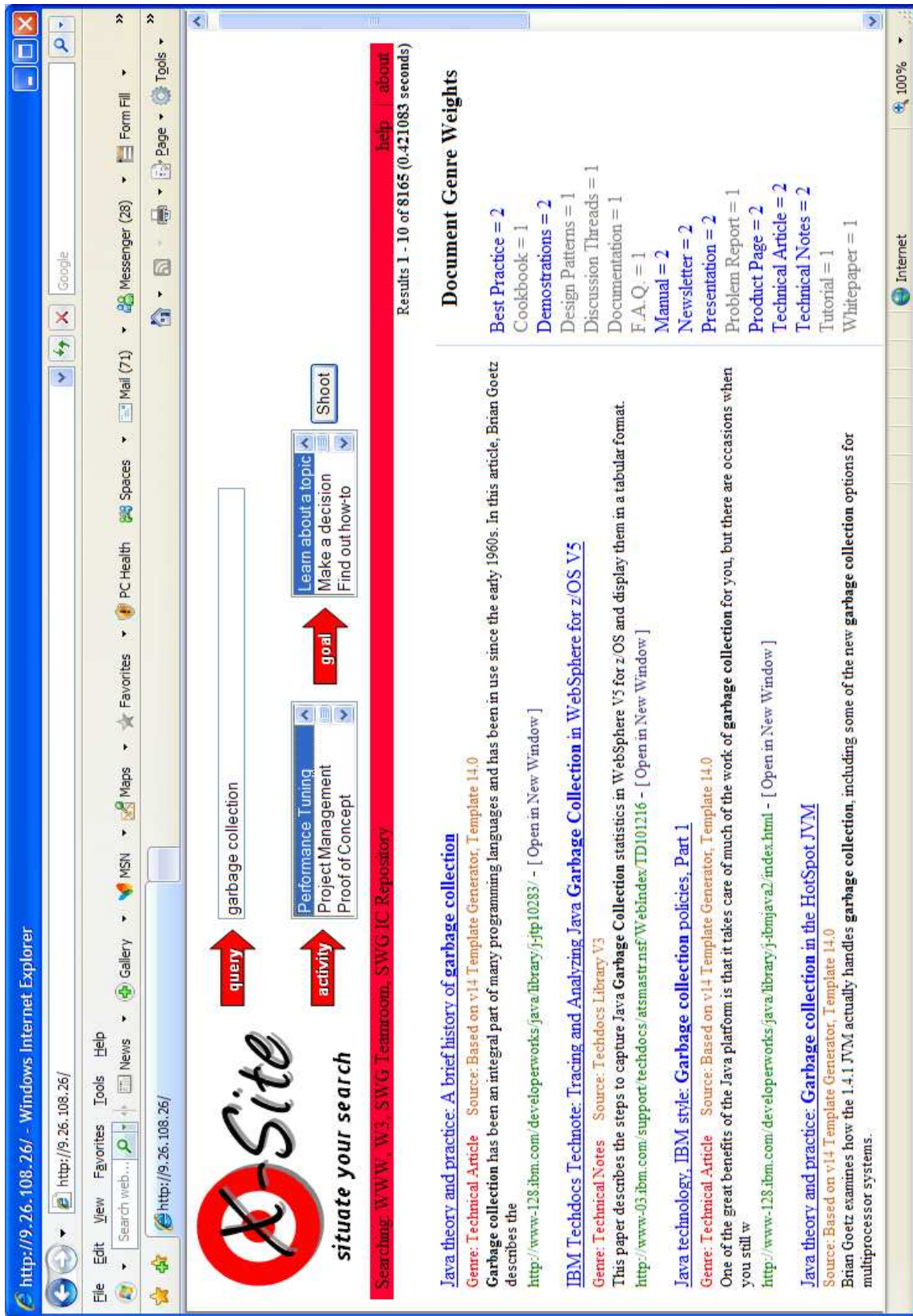


Figure 5.14: X-Site: Performance and Learn a Topic

For the next case, assume the user is trying to learn how to implement the actual work instead of learning the topic. S/he would then select **Performance Tuning** and **How To** for the search context. For **Performance Tuning** and **How To**, the following genres are given more weights than other genres:

- Best Practice,
- Cookbook,
- Demonstrations,
- Discussion Threads,
- Documentation,
- Manual,
- Presentation,
- Technical Notes, and
- Tutorial.

For this particular search context, *Technical Article* is not a related document genre and thus, it is given only a weight of 1. Although *Technical Article* documents are only given a weight of 1, one of its documents still has a high query term frequency and thus, it is ranked second of the result list. Again, the filtering approach would have eliminated this document from consideration.

1. IBM Techdocs Technote: Tracing and Analyzing Java Garbage Collection in Web-sphere for z/OS V5  
(*Genre: Technical Notes*)
2. Java theory and practice: A brief history of garbage collection  
(*Genre: Technical Article*)
3. Java theory and practice: Garbage collection in the HotSpot JVM
4. IBM: Java z/OS - z/OS Garbage Collection



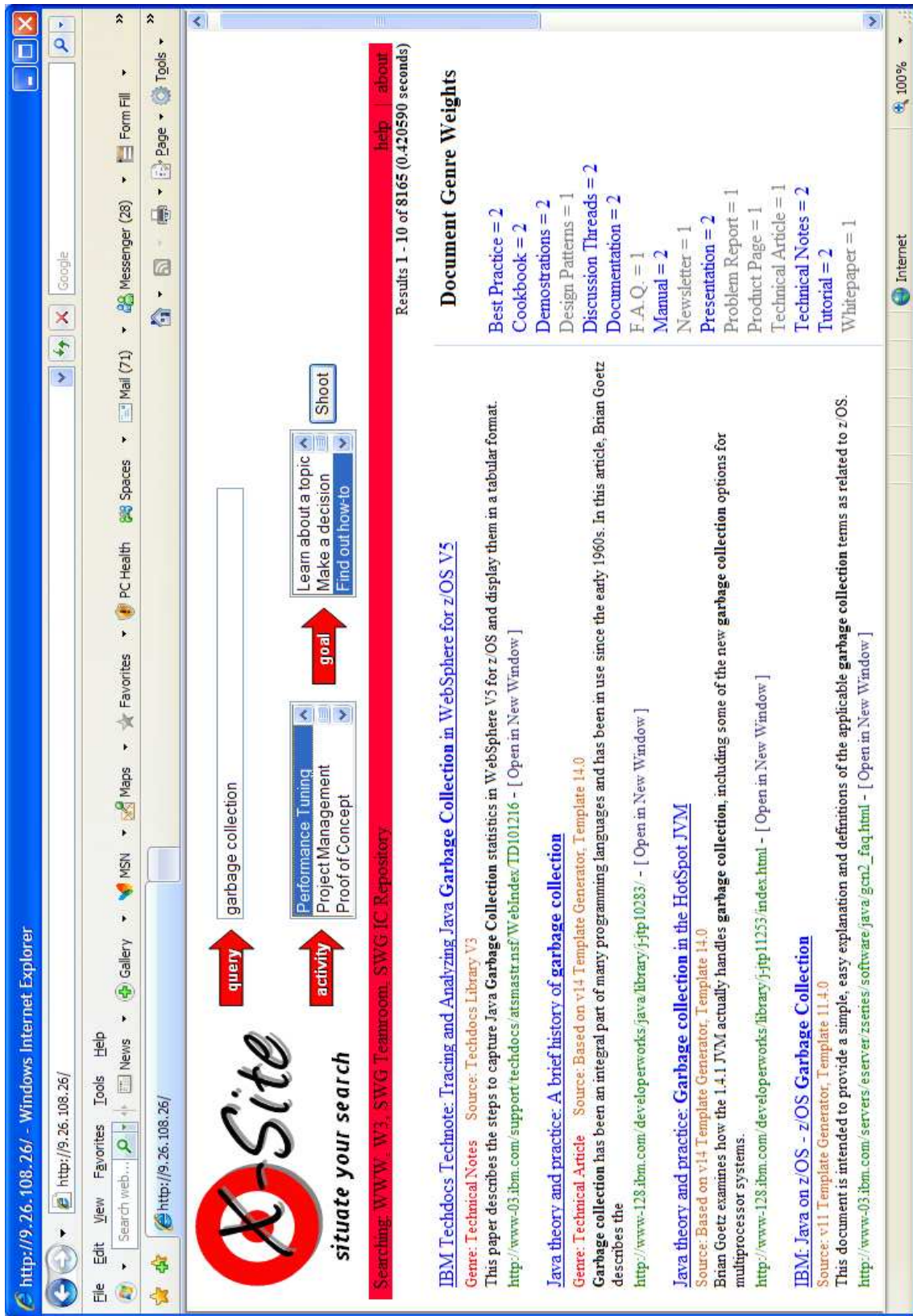


Figure 5.15: X-Site: Performance and How-To

Finally, let's consider a case where *Technical Article* and *Technical Notes* are not given extra weights. By selecting **Proof of Concept** and **How-To** as *activity task* and *information goal* respectively, the following genres are given extra weights.

- Best Practice,
- Cookbook,
- Demonstrations,
- Design Patterns,
- Documentation,
- Manual,
- Product Page, and
- Tutorial.

As shown in Figure 5.16, the ranking on the result list is the same as the first scenario where activity task and information goal are not given. That means, for this particular query, documents from other genres are not as relevant as the ones in *Technical Article* and *Technical Notes*.

1. Java theory and practice: A brief history of garbage collection  
(*Genre: Technical Article*)
2. Java theory and practice: Garbage collection in the HotSpot JVM
3. IBM: Java on z/OS Garbage Collection
4. Garbage collection on Java 1.1.8 JVMs

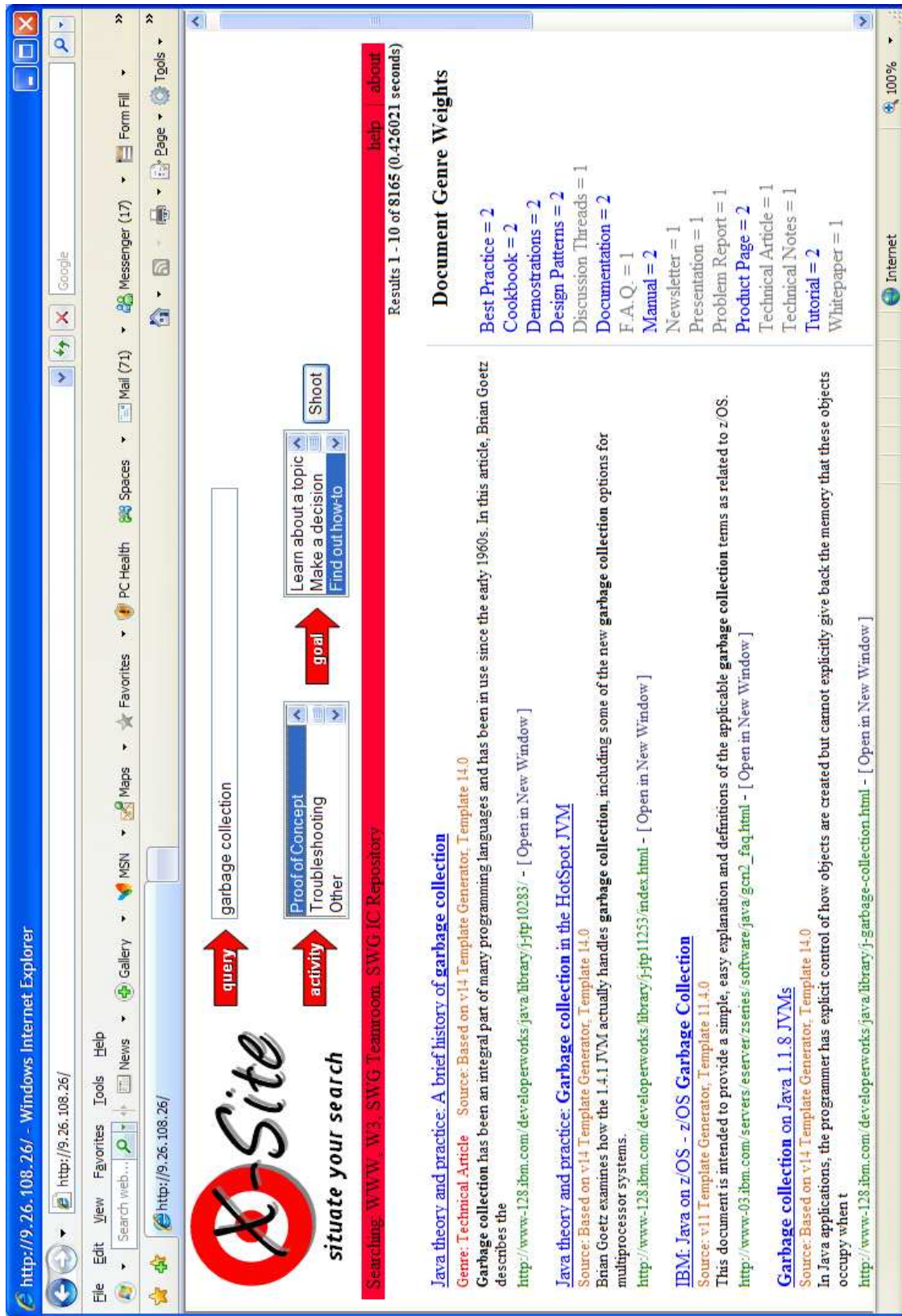


Figure 5.16: X-Site: Proof of Concept and How-To

## 5.7 Chapter Summary

This chapter illustrates how the techniques introduced in the previous two chapters can be deployed in a real workplace environment. In Chapter 3, the task-genre relationships are incorporated into the document relevance scoring function, Okapi BM25, by combining with the term frequencies. In Chapter 4, each relationship are weighted using different resources provided by users.

In this chapter, we introduce a new contextual search tool—X-Site—for an enterprise search environment. X-Site is currently deployed at IBM Corporation and mainly used by software engineers. When a user provides her/his *activity task* and *information goal*, X-Site returns a suitable ranking list of documents depending on the related document genres.

This chapter first illustrates the implementation of the system and then demonstrates how the ranking of documents changes after a user modifies her/his search context. Future improvements of the X-Site system will include a component to monitor implicit measures of document preference during system use in order to tune the task-genre associations in accordance with patterns of user behaviour.



# Chapter 6

## Conclusions

As more and more information are generated within a corporate domain, the task of searching relevant documents from this domain becomes harder and harder. Poor search quality has caused corporations to suffer in the forms of lost opportunities and lost productivity [FS03]. As a result, there is a need to develop high-precision search tools for enterprise search.

For enterprise search, an employee typically uses a retrieval system to seek answers to a problem that s/he has on hand. This problem should relate to some work task that s/he is currently working on. This work task identifies the *purpose* of each search request. An enterprise retrieval system can utilize this information and return documents that are aimed to satisfy this purpose. In order to do so, the system must identify the purpose of each document and match it with the purpose of each search request.

To identify the purpose of each document, a retrieval system can classify documents into groups based on their content, structure, subject, and form. Each document group is defined as *document genre*. Document genre is a distinctive type of communicative action, characterized by a socially recognized communicative purpose and common aspects of form [OY94]. In this thesis, we use document genre to identify the purpose of each document and match each document genre to relevant search tasks.

There are many approaches for incorporating document genre in the retrieval process. In this thesis, we chose to take a simple but effective approach by incorporating genre weight into the popular relevance scoring function, Okapi BM25. Given a specific work task, a set

of relevant document genres can be identified. We can then weight each document genre based on how relevant it is to the work task. Finally, this weight is combined with term frequency to influence the relevance score of a document.

This thesis also proposed two methods for estimating the weight for each task-genre relationship based on user feedback. There are two pieces of user information that we can use to estimate each weight: relevance judgments and clickthrough data. Relevance judgments are direct indications, determined by assessors, on whether documents are relevant or irrelevant to search requests. Clickthrough data are documents that users decided to click on from search result pages and thus, are indirect indications for document relevance. These resources help our search system determine a relatively realistic weight for each task-genre relationship.

The methods proposed in this thesis were tested in a real workplace setting. A new search system—X-Site—was deployed at a major technology corporation for experimental purposes. X-Site is a contextual search engine that uses the relationships between work tasks and document genres to improve search precision for software engineers. This thesis also illustrates each component of X-Site and its implementation in detail.

# Chapter 7

## Future Work

### 7.1 User Study

In Chapter 4, two methods were proposed for estimating the weight for each task-genre relationship. These methods, however, have not been experimented with X-Site and the IBM corporate collection. In order to test these methods in a real workplace environment, a user study needs to be carried out with X-Site and with IBM software engineers.

In order to obtain both resources—relevance judgments and clickthrough data—from IBM software engineers, two versions of X-Site should be implemented. One version is similar to the system described above and it would record the documents that users click on. Another version should have a feedback mechanism on resulting documents so that users can judge the quality of those documents with respect to their search task. Figure 7.1 shows the current feedback mechanism used on the IBM web site<sup>1</sup>.

In addition to using clickthrough data to make an initial estimation of each task-genre weight, X-Site continues to record clickthrough data and update the corresponding weight accordingly. If users continuously click on *F.A.Q.* documents for *Solve a Problem* task, then their corresponding weight should increase until it gets to a maximum value. Conversely, if a document genre is not frequently clicked by users for some work task, the corresponding weight would decrease. Hence, each task-genre weight is updated dynamically.

---

<sup>1</sup><http://www.ibm.com>

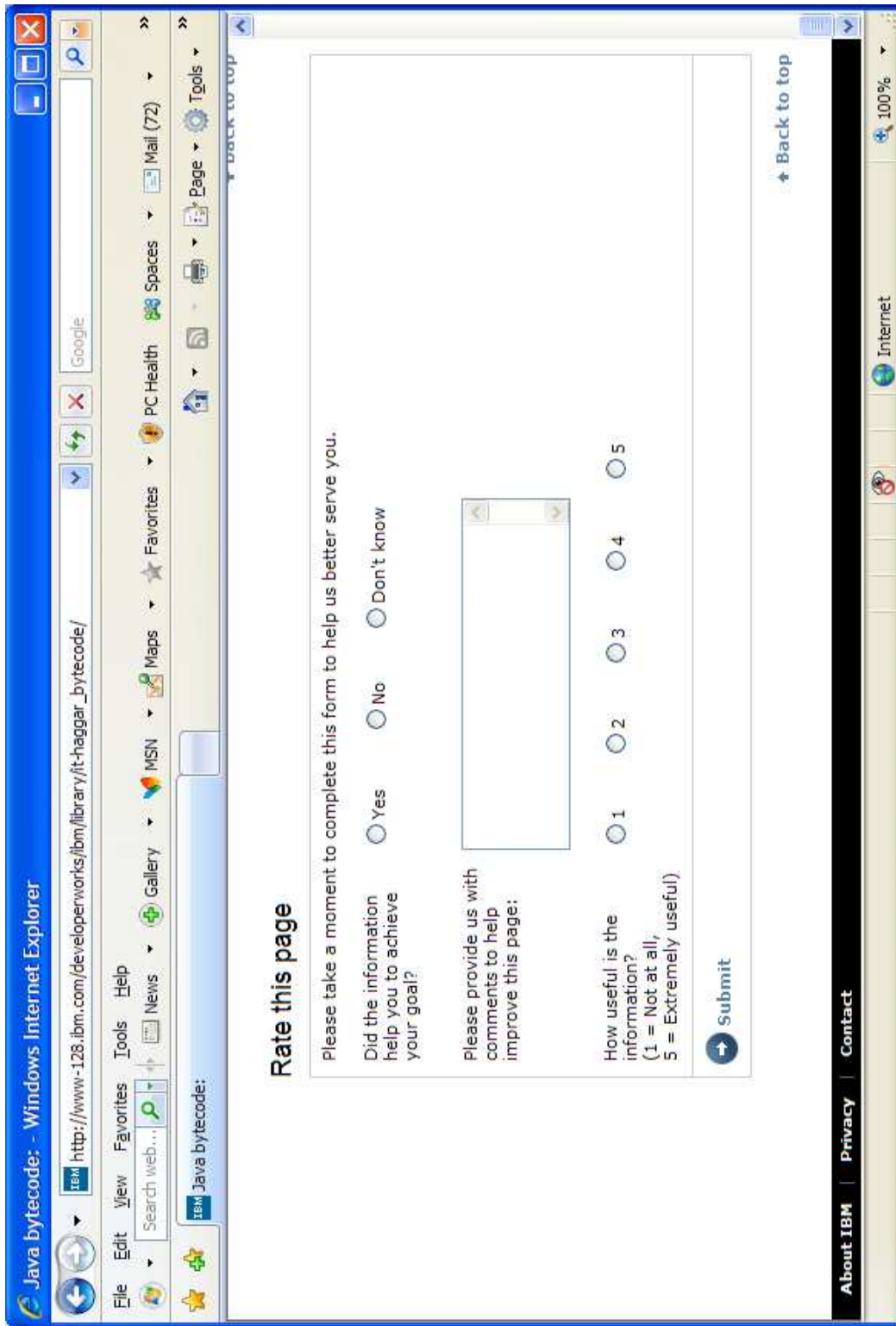


Figure 7.1: Relevance Feedback form on ibm.com

There are two main advantages to this:

- Whether the initial estimations are accurate or not, the weights are being updated to accurate values.
- If user preference changes over time, then the weights adjust to these changes as well.

## 7.2 Using Annotations

Enterprise Search faces many different challenges that were not introduced from Web Search. For instance, anchor text may not have much impact on enterprise search as it does on web search. The reason is that the enterprise search environment is mainly influenced by the structure and policy of a corporation. Each document cannot easily be created or modified by an employee without following the restricted policy enforced by the corporation. As a result, the number of pages within a corporate collection is significantly less than the number of web pages and the quality of anchor text is also worse. Dmitriev et al. [DEFS06] proposed using annotations in enterprise Search to replace anchor text.

For X-Site, using annotations for intranet pages and for pages from Lotus Notes databases can possibly be beneficial. For Internet pages, anchor text can still be used. However, the following questions have arisen involving this procedure.

- Can annotations still be used for Internet pages by appending user-provided text to anchor text?
- Is there any difference between the field weights for annotations and for anchor text?
- What happens when there are links connecting Internet pages, intranet pages, and Lotus Notes documents?

Using annotations in X-Site is a suitable solution for replacing anchor text with intranet pages and Lotus Notes documents. It could possibly solve one of many minor problems for enterprise search. However, the key idea for this thesis is to investigate the relationships between work tasks and document genres, which solves a fundamental problem in enterprise search. Using annotations to represent anchor text can be considered if X-Site were engaged in permanent operation.

### 7.3 SVM Scores

X-Site uses *Support Vector Machines (SVMs)* to classify every document in a collection into different document genres. SVMs output a score for each document, which determines whether the document belongs in a genre or not. Instead of making a binary decision in genre classification, X-Site can take this SVM score as a *confidence weight* and combine it with the actual genre weight to get a new weight for the task-genre pair.

$$w_{actual} = w_G * score_{SVM} \quad (7.1)$$

where  $w_G$  = the weight for each task-genre relationship;  
 $score_{SVM}$  = the score produced by SVMs.

The intuition behind this is to utilize the confidence level of SVM and combine it with the belief that a particular document genre is related to a work task. If SVMs are confident that a document belongs to a genre, then more weight should be given to this document than a document that SVMs are not as confident. If SVM is not quite confident, this should cause the task-genre weight to lower because we are not sure if the document is actually related to the work task. Therefore, we are modifying the task-genre weight for a document depending on how much confident the classifier has with its categorization.

### 7.4 Additional Information on Result Pages

To improve the quality of X-Site's result page, additional information can be added, which might be helpful to the users. These information are independent to the ranking of documents. Figure 7.2 shows two additional pieces of information that can be added to the result page: **Best Links by Product** and **Expert List**. By showing these information in a side column on the result page, users have the freedom to navigate these links and search for answers in a different approach.

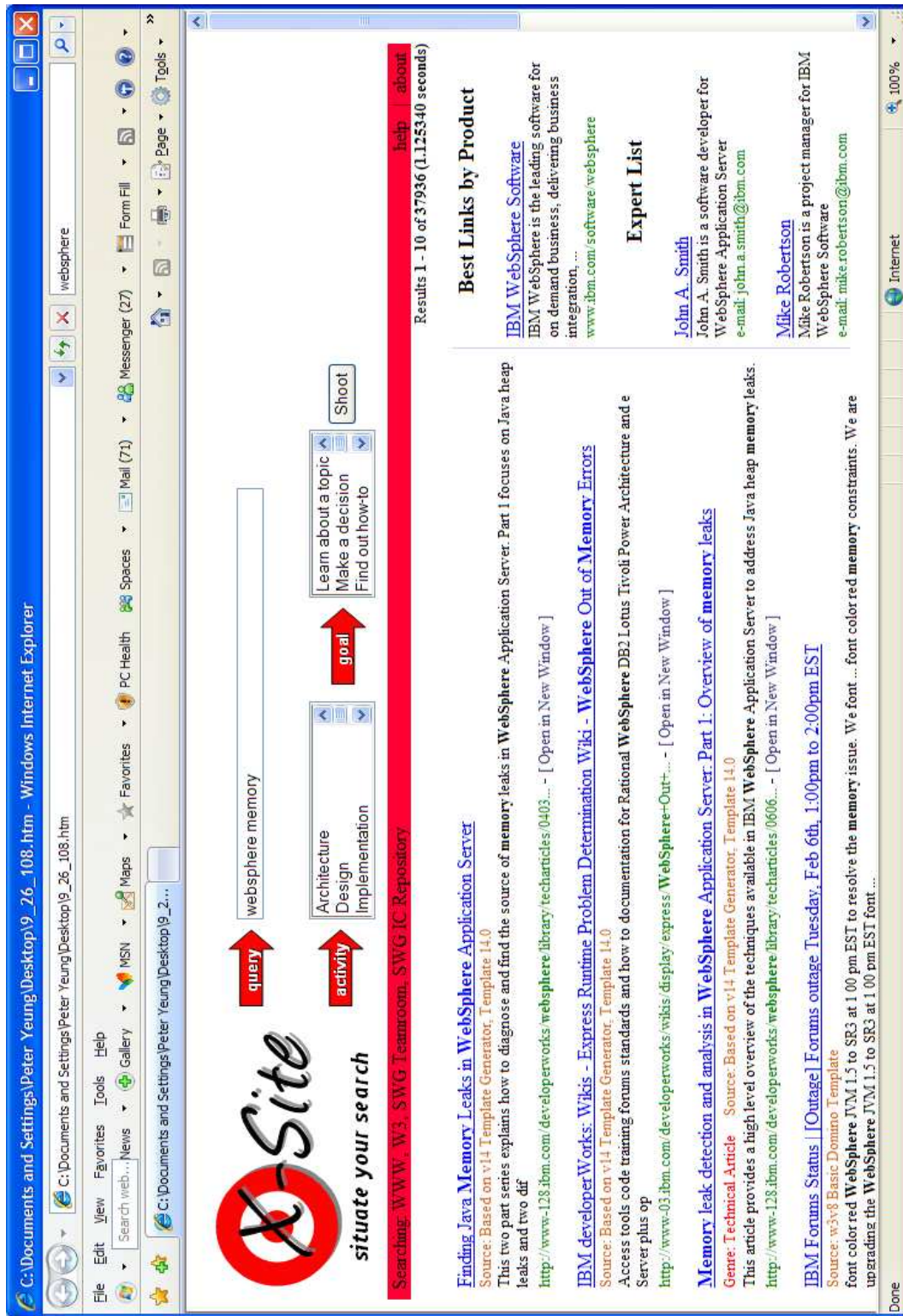


Figure 7.2: Best Links and Expert List on result page

### 7.4.1 Best Links by Project

In order to understand the benefits of displaying links to product pages, we need to understand the *need behind each query*. Broder [Bro02] developed a taxonomy that classifies searches into three different tasks: navigational task, informational task, and transactional task. The purpose of a *navigational task* is to reach a particular site, which the user has previously visited or believes to exist. S/he then navigates the site to acquire information about a topic. *Informational task* is to acquire information, which is believed to be available, from one or more web pages. Here, we only focus on these two tasks.

The approach taken in this thesis assumes all search requests belong to the information task. For each search request, a list of documents is returned to the users and they can acquire information by viewing these documents. However, we ignore the fact that some search requests belong to the navigational task. Many enterprise search requests are often related to a specific product from the corporation. For instance, many search requests at IBM are related to its products like WebSphere, Lotus Notes, Lenovo ThinkPads, etc.. Users would occasionally search for home pages of these products and acquire information by navigation. A direct link to these products' home pages help the users reach a point where they can continue to search for information. Therefore, this allows the users to take the navigational approach if the list of documents returned is not adequate.

### 7.4.2 Expert List

Expert search has been a big component in enterprise search. There are situations where documents do not contain all the information needed to satisfy a search request. It might require users to talk to people who are knowledgeable on these topics and obtain information from them. Hence, finding a group of experts for a search request is a smaller challenge within the enterprise search problem.

In a corporation, each employee can be identified by her/his academic background, knowledge, skills, department, etc.. These information can be accurately organized and be used for identifying experts. In addition, a group of experts can be identified by analyzing the content of documents in the corporate collection. For example, a sender of an email relating to a topic might be an expert. (However, the sender might be posting a question



on a topic and thus, s/he is not an expert.)

If a user cannot satisfy her/his information need from the list of documents shown on the result page, s/he can choose to contact the list of experts directly. Therefore, by providing this expert list, users would know who to contact for a particular topic and can continue in their search quests if desired.



# Bibliography

- [ABD06] E. Agichtein, E. Brill, and S. Dumais. Improving web search ranking by incorporating user behavior information. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 19–26, New York, NY, USA, 2006. ACM Press.
- [ABDR06] E. Agichtein, E. Brill, S. Dumais, and R. Ragno. Learning user interaction models for predicting web search result preferences. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 3–10, New York, NY, USA, 2006. ACM Press.
- [ALM<sup>+</sup>01] M. Abrol, N. Latache, U. Mahadevan, J. Mao, R. Mukherjee, P. Raghavan, M. Tourn, J. Wang, and G. Zhang. Navigating large-scale semistructured data in business portals. In *Proceedings of the 27th VLDB Conference*, pages 663–666, 2001.
- [BCY06] S. Büttcher, C.L.A. Clarke, and P.C.K. Yeung. Index pruning and result reranking: Effects on ad-hoc retrieval and named page finding. In *TREC-15: Proceedings of the fifteenth conference on Text retrieval conference*, pages 237–244, Elmsford, NY, USA, 2006. Pergamon Press, Inc.
- [BCYS07] S. Büttcher, C.L.A. Clarke, P.C.K. Yeung, and I. Soboroff. Reliable information retrieval evaluation with incomplete and biased judgements. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference*

*on Research and development in information retrieval*, New York, NY, USA, 2007. ACM Press.

- [Bro02] A. Broder. A taxonomy of web search. *SIGIR Forum*, 36(2):3–10, 2002.
- [CDR<sup>+</sup>98] S. Chakrabarti, B. Dom, P. Raghavan, S. Rajagopalan, D. Gibson, and J. Kleinberg. Automatic resource compilation by analyzing hyperlink structure and associated text. *Comput. Netw. ISDN Syst.*, 30(1-7):65–74, 1998.
- [CSdV06] N. Craswell, I. Soboroff, and A. de Vries. Overview of the trec-2006 enterprise track. In *Proceedings of the 15th Text REtrieval Conference*. ACM Forum, November 2006.
- [CT94] W.B. Cavnar and J.M. Trenkle. N-gram-based text categorization. In *Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, pages 161–175, Las Vegas, US, 1994.
- [DEFS06] P.A. Dmitriev, N. Eiron, M. Fontoura, and E. Shekita. Using annotations in enterprise search. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 811–817, New York, NY, USA, 2006. ACM Press.
- [DVDM01] N. Dewdney, C. VanEss-Dykema, and R. MacMillan. The form is the substance: classification of genres in text. In *Proceedings of the workshop on Human Language Technology and Knowledge Management*, pages 1–8, Morristown, NJ, USA, 2001. Association for Computational Linguistics.
- [ES04] S.M. Zu Eissen and B. Stein. Genre classification of web pages. In Susanne Biundo, Thom Frühwirth, and Günther Palm, editors, *Proceedings of KI-04, 27th German Conference on Artificial Intelligence*, Ulm, DE, 2004. Published in the “Lecture Notes in Computer Science” series, number 3238.
- [FKM<sup>+</sup>05] S. Fox, K. Karnawat, M. Mydland, S. Dumais, and T. White. Evaluating implicit measures to improve web search. *ACM Trans. Inf. Syst.*, 23(2):147–168, 2005.

- [Fre07] L. Freund. Exploiting task-document relationships to support information retrieval in the workplace. *Unpublished Doctoral Dissertation, University of Toronto*, 2007.
- [FS03] S. Feldman and C. Sherman. The high cost of not finding information. *IDC Technical Report 29127*, 2003.
- [FTC05] L. Freund, E.G. Toms, and C.L.A. Clarke. Modeling task-genre relationships for ir in the workplace. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 441–448, New York, NY, USA, 2005. ACM Press.
- [Haw04] D. Hawking. Challenges in enterprise search. In *ADC '04: Proceedings of the 15th Australasian database conference*, pages 15–24, Darlinghurst, Australia, Australia, 2004. Australian Computer Society, Inc.
- [JFM97] T. Joachims, D. Freitag, and T.M. Mitchell. Web watcher: A tour guide for the world wide web. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, volume 1, pages 770–777. Morgan Kaufmann, 1997.
- [JGP<sup>+</sup>05] T. Joachims, L. Granka, B. Pan, H. Hembrooke, and G. Gay. Accurately interpreting clickthrough data as implicit feedback. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 154–161, New York, NY, USA, 2005. ACM Press.
- [Joa98] T. Joachims. Text categorization with support vector machines: learning with many relevant features. In Claire Nédellec and Céline Rouveirol, editors, *Proceedings of ECML-98, 10th European Conference on Machine Learning*, number 1398, pages 137–142, Chemnitz, DE, 1998. Springer Verlag, Heidelberg, DE.

- [Joa02a] T. Joachims. Evaluating retrieval performance using clickthrough data. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR Workshop on Mathematical/Formal Methods in Information Retrieval*, 2002.
- [Joa02b] T. Joachims. Optimizing search engines using clickthrough data. In *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142, New York, NY, USA, 2002. ACM Press.
- [KF06] D. Kelly and X. Fu. Elicitation of term relevance feedback: an investigation of term source and context. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 453–460, New York, NY, USA, 2006. ACM Press.
- [KT03] D. Kelly and J. Teevan. Implicit feedback for inferring user preference: a bibliography. *SIGIR Forum*, 37(2):18–28, 2003.
- [OY94] W.J. Orlikowski and J. Yates. Genre repertoire: The structuring of communicative practices in organizations. *Administrative Science Quarterly*, 39:541–574, 1994.
- [PBMW98] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998.
- [RWHB<sup>+</sup>94] S.E. Robertson, S. Walker, M. Hancock-Beaulieu, A. Gull, and M. Lau. Okapi at trec-3. In *Proceedings of Text REtrieval Conference*, November 1994.
- [RZT04] S.E. Robertson, H. Zaragoza, and M. Taylor. Simple bm25 extension to multiple weighted fields. In *CIKM '04: Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 42–49, New York, NY, USA, 2004. ACM Press.
- [SB97] G. Salton and C. Buckley. Improving retrieval performance by relevance feedback. pages 355–364, 1997.

- [STZ05a] X. Shen, B. Tan, and C. Zhai. Context-sensitive information retrieval using implicit feedback. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 43–50, New York, NY, USA, 2005. ACM Press.
- [STZ05b] X. Shen, B. Tan, and C. Zhai. Implicit user modeling for personalized search. In *CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 824–831, New York, NY, USA, 2005. ACM Press.
- [TDH05] J. Teevan, S. Dumais, and E. Horvitz. Personalizing search via automated analysis of interests and activities. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 449–456, New York, NY, USA, 2005. ACM Press.
- [Vap95] V. N. Vapnik. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995.
- [Whi06] R.W. White. Using searcher simulations to redesign a polyrepresentative implicit feedback interface. *Inf. Process. Manage.*, 42(5):1185–1202, 2006.
- [XZC<sup>+</sup>04] G.R. Xue, H.J. Zeng, Z. Chen, Y. Yu, W.Y. Ma, W.S. Xi, and W.G. Fan. Optimizing web search using web click-through data. In *CIKM '04: Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 118–126, New York, NY, USA, 2004. ACM Press.
- [YBCK07] P.C.K. Yeung, S. Büttcher, C.L.A. Clarke, and M. Kolla. A bayesian approach for learning document type relevance. In *Advances in Information Retrieval, 29th European Conference on Information Retrieval*, pages 753–756, Berlin Heidelberg, 2007. Springer-Verlag.
- [YCB07] P.C.K. Yeung, C.L.A. Clarke, and S. Büttcher. Improving retrieval accuracy by weighting document types with clickthrough data. In *SIGIR '07: Proceed-*

*ings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, New York, NY, USA, 2007. ACM Press.

- [YFC07] P.C.K. Yeung, L. Freund, and C.L.A. Clarke. X-site: A workplace search tool for software engineers. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, New York, NY, USA, 2007. ACM Press.