Université de Montréal

Context-Sensitive Information Retrieval

par

Jing Bai

Département d'informatique et de recherche opérationnelle

Faculté des arts et des sciences

Thèse présentée à la Faculté des études supérieures

en vue de l'obtention du grade de

Doctorat ès sciences (Ph.D.)

en Informatique

Juin 2007

Université de Montréal

Faculté des études supérieures

Cette thèse intitulée:

Context-Sensitive Information Retrieval

Présentée par:

Jing Bai

a été évaluée par un jury composé des personnes suivantes:

Guy Lapalme, président-rapporteur

Jian-Yun Nie, directeur de recherche

Pascal Vincent, membre du jury

Keith C. J. van Rijsbergen, examinateur externe

Yves Marcoux, représentant du doyen de la FES

Thèse acceptée le 11 octobre 2007

Résumé

La recherche d'information (RI) est devenue plus que jamais un outil indispensable. Avec la croissance en volume d'informations, les besoins d'information des usagers se sont aussi beaucoup diversifiés. Une même requête peut être utilisée pour spécifier des besoins d'information très différents pour des usagers différents. Dans une telle situation, la requête ne peut plus être considérée comme le seul élément qui spécifie le besoin d'information. D'autres facteurs contextuels doivent être pris en compte.

Cependant, les approches traditionnelles n'utilisent que la requête pour retrouver des documents. Ces approches deviennent insuffisantes. Cette étude aborde le problème d'intégration des facteurs contextuels dans les opérations de RI.

Il existe un grand nombre d'études qui révèlent l'importance des facteurs contextuels en RI. Mais peu d'entre eux ont été intégrés dans des systèmes opérationnels. Dans cette étude, nous intégrons plusieurs facteurs contextuels dans un cadre uniforme basé sur la modélisation statistique de langage. Chaque facteur est utilisé pour construire un modèle de langage composant pour la requête, qui définit le besoin d'information d'un point de vue différent. Le modèle de requête final combine tous les modèles composants, qui ont pour objectif de rendre la requête plus complète. Nous considérons une requête plus complète comme une meilleure spécification du besoin d'information, qui permet de retrouver plus de documents pertinents.

Dans cette étude, nous distinguons deux types de contexte pour une requête:

- *Contexte intra-requête,* qui est le contexte spécifié par des termes inclus dans la requête. Ce type de contexte nous permet de mieux comprendre le

sens de termes de la requête, et ainsi déterminer des termes d'expansion plus appropriés;

- *Contexte extra-requête,* qui définit des éléments au delà de la requête, reliés à l'environnement de celle-ci. Dans cette recherche, nous étudions deux contextes de ce type: le domaine du sujet de la requête qui définit le *background* de la requête, et des caractéristiques de la collection de documents qui révèlent les autres sujets avec lesquels le sujet de la requête est développé dans la collection. Ces contextes nous suggèrent des termes reliés à la requête de deux points de vue différents.

Pour exploiter le contexte intra-requête, nous créons des relations entre termes qui déterminent des termes reliés selon des contextes plus riches. Deux modèles de langage sont créés pour modéliser des contextes extra-requête: le modèle de domaine est défini par un ensemble de documents dans le domaine, tandis que les caractéristiques de la collection reliées à la requête sont reflétées par un ensemble de documents de *feedback.*

Nous montrons dans cette étude que les deux types de contexte sont très utiles. La méthode que nous proposons a été testée sur des collections TREC. Nos résultats expérimentaux montrent les faits suivants:

- L'intégration des facteurs contextuels est très bénéfique en RI. Chaque facteur contextuel apporte de grandes améliorations sur la performance de recherche. Le modèle de requête complet qui intègre tous les facteurs contextuels donne la meilleure performance;

- Les relations contextuelles entre des termes, qui exploitent le contexte intra-requête, peuvent déterminer des termes d'expansion plus pertinents que les relations traditionnelles de co-occurrence sans exploitation du contexte intra-requête;

- Les contextes extra-requête peuvent aussi apporter des améliorations importantes sur la performance. En particulier, il est faisable de construire des domaines de sujet et de les utiliser pour renforcer le modèle de requête. Nous avons testé différentes stratégies pour construire et exploiter les domaines de sujet. Toutes ces stratégies se sont avérées performantes;
- Cette étude montre aussi qu'en exploitant les documents de feedback, nous pouvons capter des caractéristiques de la collection reliées à la requête. Ces caractéristiques ont été très utiles.

Cette étude est la première à intégrer plusieurs facteurs contextuels, et à montrer que ceci est faisable et bénéfique sur des collections TREC. Ainsi, elle montre une direction prometteuse pour des développements futurs.

Mots clés: Recherche d'information, Expansion de requête, Contexte de requête, Relation entre terme, Modèle de domaine, Modèle de langage.

Abstract

Information retrieval (IR) is more than ever an indispensable tool in today's society. With the growth of volume of available information, users' information needs have also much diversified. The same query can be used to specify very different information needs by different users. In such a situation, the query can no longer be taken as the only element that specifies the information need. Other contextual factors should be considered.

However, the traditional approaches only use the query to retrieve documents. These approaches become insufficient. This study investigates the integration of contextual factors in IR operations.

There have been many studies showing the importance of contextual factors in IR. However, few of them have been integrated in operational IR systems. In this study, several contextual factors are integrated within a uniform framework based on a language modeling approach. Each contextual factor is used to build a component query model that defines the information need from a different perspective. The final query model combines all the component query models, which make the query more complete. We consider a more complete query as a better specification of the information need, which allows us to retrieve more relevant documents.

In this study, we distinguish two types of context for a query:

- *Intra-query context*, which is the context specified by the terms included in the query. This type of context helps us to understand the meaning of query terms and to determine the appropriate expansion terms;
- *Extra-query context*, which defines elements beyond the query itself and relates to the environment of the query. In this study, we investigate two of them: topic domain of the query which defines background information of the query; and the characteristics of the document collection which reveal the other topics that are developed together with the query topic in the collection.

To exploit intra-query context, we create context-dependent term relations that determine related terms according to richer contexts. Two language models are created to model extra-query contexts: the domain model is defined by a set of in-domain documents, while the query-related collection characteristics are reflected by a set of feedback documents.

We show in this study that both types of context are very useful. Our method has been tested on several TREC collections. Our experiments show that:

- The integration of contextual factors is very useful in IR. Each of the contextual factors brings large improvements in retrieval effectiveness. The complete query model that integrates all the contextual factors performs the best;
- The context-dependent term relations that exploit intra-query context can determine more relevant expansion terms than the traditional co-occurrence relations which do not exploit intra-query context;
- The extra-query contexts can also bring large improvements in retrieval effectiveness. In particular, it is feasible to build topic domain models and use them to enhance the query. We have tested different strategies to build

and to exploit the topic domain models. All the strategies have proven to be quite effective;

- This study also shows that by exploiting pseudo feedback documents, we can capture some query-related collection characteristics. These characteristics are very helpful.

This study is the first one that integrates multiple contextual factors and shows that this is feasible and beneficial on TREC collections. It paves the way for future developments in this direction.

Keywords: Information retrieval, Query expansion, Query context, Term relation, Domain model, Language model

Table of Contents

List of Tables

List of Figures

List of Symbols

D: A document – a set of words

Q: A query – a set of words

θ_D: A statistical language model for document

θ_Q: A statistical language model for query

θ_Q^0: Initial query model

θ_Q^K: Expanded query model by knowledge

θ_Q^{Dom}: Expanded query model by domain

θ_Q^{FB}: Expanded query model by feedback documents

α_i: Mixture weight for a component query model

Acknowledgment

I would like to express my sincere gratitude to my advisor, Professor Jian-Yun Nie, for his constant support and advice. Over the last five years, I have learned a lot from him. His truly scientist intuition and methodology have deeply inspired me and guided me through the path from a student to a researcher. Without his guidance, this work would not be possible.

I am particularly grateful to Professor Keith C. J. van Rijsbergen for accepting to be the extern examiner of this thesis. My special thanks are also extended to Professor Guy Lapalme and Pascal Vincent, who have accepted to be the jury members for the thesis and provided very helpful suggestions on this dissertation.

I was very fortunate to work in the Laboratoire de Recherche Appliquée en Linguistique Informatique (RALI) with all the friendly colleagues – Guihong Cao, Hugues Bouchard, Philippe Langlais, Elliott Macklovitch, Fabrizio Gotti, etc. It has been a great pleasure to work with them.

Finally, I would like to thank my parents for their endless love and encouragement. This thesis is dedicated to them.

To my parents

Chapter 1

Introduction

Everyone recognizes that we are living in a period of information expansion. Information of every kind is more and more available, especially on the Web. According to Netcraft[1], the number of Web sites in May 2007 is more than 118 millions, among which about 54 millions are active. In the following figure, we can observe the rapid increase of these numbers since 1995.

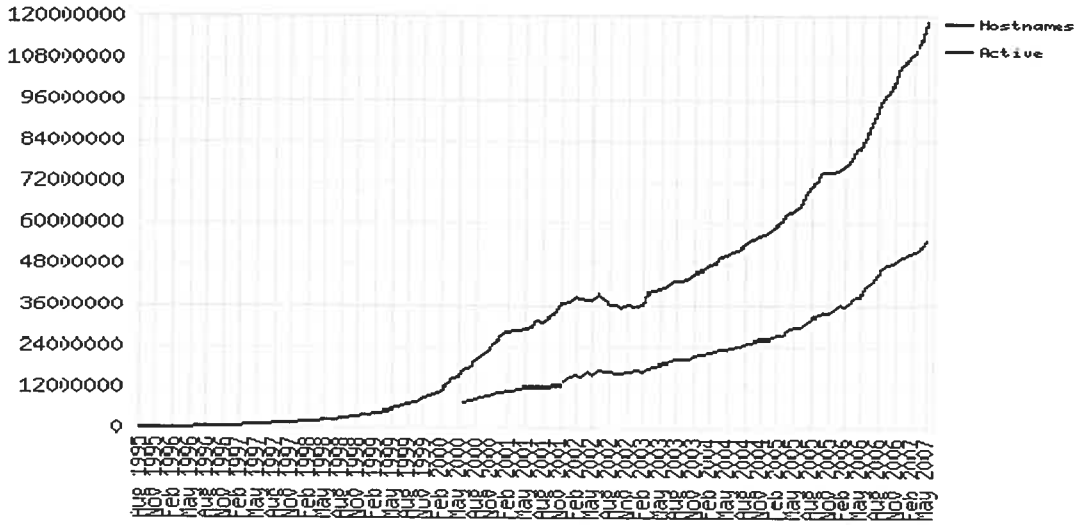

Figure 1. Total sites across all domains in August 1995 – May 2007 (from Netcraft [1])

[1] http://news.netcraft.com/archives/web_server_survey.html

The number of Web pages also increased at the same rate. In 2005, Guli and Signorini [39] estimated that the Web size exceeded 11.5 billion pages. In February 2007, Boutell[2] revised this number to 29.7 billion pages.

The growth in quantity of information can benefit end users by allowing them to find information of any kind on the Web. Indeed, more and more users consider the Web as their primary source of information. However, a piece of information becomes useful only when it is provided to the *right* user for the *right* information need.

The goal of information retrieval (IR) systems is to find relevant documents from large document collections (including the Web) to satisfy the information needs of users. These systems are central to many applications and tasks. They are a means to transform a piece of information into a piece of *useful* information. The utility of IR systems is largely exemplified by the popular utilization of search engines on the Web, which are examples of IR systems in the Web environment.

Despite the popularity of search engines and the progress in IR technologies, users are still faced with the problem of retrieving irrelevant documents and missing relevant ones. In many cases, the retrieved information is not relevant, while much relevant information is not retrieved. There is an acute demand for more accurate IR.

Along with the growth in the volume of information, there is also a sharp increase in kinds of information available, as well as kinds of information needs from users. The traditional basic IR function is to suggest documents which contain potentially relevant information. This basic function has been largely extended in recent years due to the development of the Web. For example, one can use the results of a search engine to calculate the popularity of a term in a language. For example, using *Google*[3], one can estimate that the term "*search engine*" is a much more popular term

---

[2] http://www.boutell.com/newfaq/misc/sizeofweb.html

[3] This estimation is found by *Google* on May 26, 2007.

than "*information retrieval*" since the former appears in about 219 million documents on the Web, while the latter in 24 million documents.

Information needs also diversified. Search engines are not only used to find documents, they are also used as a means to construct meaning [13]. End users may use IR systems and search engines to find various types of information, for example, documents describing a concept or a specific event, the lowest rate for a travel, answers to a specific question, etc. Yet the description about the variety of information needs may be very similar. For example, a user can issue the query "*Java Montreal*" to find information about the "*discussion forum on Java programming language in Montreal*". Another user may want to find information about "*an event in Montreal about Java language*". And a third user may want to find "*travel information from Montreal to Java Island*" with the same query. This is just one realistic example of information search on the Web. It illustrates the typical higher expectation that end users place on search engines: they want search engines to better understand their information needs, instead of just taking the words included in their queries. This also creates and enhances several key challenges for modern IR systems:

- How can we better *understand* the information need from a user?
- How can a system better judge document *relevance* to a user's information need?

These problems have existed since the beginning of the area of IR, even when IR systems were designed to find references in libraries. These issues are related to the following facts:

- A piece of information can be described in different ways by different words in natural languages;
- A user's information need is often described by a short query;

- The judgment of relevance is subjective and dependent on the user and on the context in which the query is issued.

Traditional IR approaches rely on word matching between a document and a query. However, the diversity of information and information needs makes it even clearer than ever that it is insufficient to determine documents solely according to the words that occur in the query and in the documents. IR approaches should go beyond the keywords of the query.

In this thesis, we will address the above questions from a broader perspective: IR is placed within a retrieval context, which contains useful information about the underlying information needs. In addition, to judge if a document is relevant to a query, some form of inference is performed so that a retrieved document does not have to contain the words of the query, but can contain related words. In doing so, a short query can be enhanced by exploiting different types of resources and knowledge.

In the following sections of this chapter, we will further outline the underlying problems of current IR that we will deal with (these problems will be analyzed in more details later). Then we will describe the basic ideas that we propose to deal with them. Finally, we will outline the overall organization of the dissertation.

1.1 Problematics

Besides the problem of understanding and indexing documents, two key problems of IR systems, which we will deal with in this thesis, are:

- The *specification* of a user information need;
- The *judgment* of document relevance.

1.1.1 Information Need and Specification

When a user needs some information, he should formulate a query to describe it. This can be a Boolean expression such as "*information AND retrieval*", or a free text description such as "*information retrieval*". However, the formulation of a good query is not an easy task. This is due to several reasons:

- **Unclear information need**:

  In some cases, the user does not know precisely the information he is looking for. He only has a vague idea about it. This often happens in everyday life, when a person asks someone else for information. Some conversation often takes place before the user himself understands exactly what information is needed. When dealing with an end user of a search engine, the same situation can occur. However, current search engines are unable to engage in a conversation with the user in order to clarify the information need. The user is left on his own to specify his information need as much as he can. As a consequence, the specification may be vague.

- **Inexact query formulation**:

  Even if the user knows exactly what he is looking for, he often does not know how to express it clearly. For example, he may not know the best terms to use. A typical situation is when a user wants to find a replacement car part that he sees on the car, but does not know the name of it. So he may use an inappropriate word to describe it.

- **Partial specification**:

  In many other cases, even if the query contains some correct words, the words may not be the only ones to describe the corresponding concepts. For example, a user may want to shop for laptop computers and can choose to use

"*laptop*" as his query. However, other terms such as "*portable computer*", "*PC*", "*iBook*", etc. can also be used to describe laptop computers in documents. These latter documents can also be relevant to the user's information needs even if they do not contain the word "*laptop*". By including only "*laptop*" in the query, the latter documents are missed.

The problems we described above are common, especially in Web search, as general users only use 2-3 words in their queries on search engines [46] [92][112]. In [46], the following statistics have been found on a set of 54,573 queries collected from the transaction log on 10 March 1997 of the *Excite* search engine:

Table 1. Query length distribution (from [46])

Terms in query	Number of queries	% of all queries
More than 6	1,108	2
6	617	1
5	2,158	4
4	3,789	7
3	9,242	18
2	16,191	32
1	15,854	31
0	2,584	5
Total	51,433	100

We can see in this table that most users use at most 3 words in their queries. Given this situation, an acute problem is to see how the system can better understand the user's information needs behind the query, and to locate the relevant documents as much as possible.

In this study, our goal is to create a better query description for retrieval purposes. Namely, we will try to enrich the user query automatically by exploiting appropriate knowledge and contextual information.

1.1.2 Relevance Judgment

Given a query, most current approaches to IR try to determine a set of relevant documents according to whether the words included in the query also appear in the documents. These approaches are indeed based on word matching and are often called "*bag-of-words*" approaches. Undoubtedly, the results determined using this approach can still be useful. For our earlier example on "*laptop*", one would expect that there may be many documents containing the word "*laptop*", in which the user can find relevant information. In many other cases, the word matching approach is insufficient. This is the case when a query contains ambiguous words such as "*Java*". For example, a traveler may want to find information about "*bus services in Java Island*" prior to a travel. Using the query "*bus service in Java*", the user can be oriented to wrong documents about "*Java language*", in particular, about "*implementing enterprise service bus in Java*". Indeed, in current search engines, most top-ranked results with this query are more related to "*Java language*" than to "*transportation in Java Island*": among the top 100 answers from *Google*[4], 99 concerns "*Java language*" and only one is related to "*transportation*" (however, it talks about bus service in Sri Lanka, which is irrelevant to the query).

The overwhelming number of answers about "*Java language*" may be partly due to the fact that there are much more documents about "*Java language*" than about

[4] Searched on May 28, 2007.

"*transportation in Java Island*" on the Web. Nevertheless, for this particular user, most of the search results are irrelevant.

The above example shows clearly that the word matching approach is insufficient to mimic the relevance judgment expected by the user. In this particular example, it would be useful and necessary to take into account the current situation of the user, i.e., he is "*preparing a travel*".

A strong reason for the ambiguous interpretation of the query is the lack of consideration of the retrieval context: the user is preparing a travel, so he is more interested in information about "*travel*" than "*programming language*". Knowing this fact, an IR system should orient its search towards *travel*-related documents.

In fact, when the user issues a query, he has some goals, which are often hidden behind the query. He also has a background (in our example, "*travel*") or a domain of interest for a particular query. All these factors can help interpreting the query correctly. These factors are called *contextual factors*, which we will further discuss in Chapter 3.

Contextual factors usually are not considered in traditional IR systems and in current search engines (*Google*, with its personalization, is one of the few exceptions). We will show in this study that it is possible and beneficial to integrate several types of contextual factors to enhance the query from different perspectives.

1.2 Proposed Approach – Integrating Different Contextual Factors in IR

In this thesis, we focus on the two problems that we mentioned in the previous section, i.e., the incomplete specification of information need in a query, and the much simplified context-independent relevance judgment by IR systems.

Our hypothesis is that once the query is enhanced to include related words, the query is more capable of retrieving relevant documents. Let us consider again the earlier example on "*bus service in Java*". This query may appear very ambiguous for a search engine. One possible way to make it clearer is to add some related terms. For example, knowing that the user is preparing a travel, we can deduce the words, such as "*transportation*", "*hotel*" and "*flight*" are related to this background. The fact that we add the word "*transportation*" into the query "*bus service in Java*" makes the retrieval results from *Google*[5] appear to be more relevant: among the 20 top results, the documents #10, #11 and #12 are related to "*bus transportation in Java Island*". If we add more related words, the results become even more relevant. When all the above related words are added into the query, 12 results among the top 20 from *Google* deal with "*bus transportation in Java Island*".

This example shows the impact of introducing more related terms into the query: this can render the query less ambiguous and more oriented to the appropriate context. This process is commonly called *query expansion*, which is typically used to increase *recall* (more relevant documents are retrieved). In this study, we will use it as a means to increase *precision* (the retrieved documents are more relevant) as well.

Our goal in this study is to develop methods to determine the appropriate terms to be added to the query. In this study, we will consider the query enhancement from several perspectives:

- By applying appropriate *knowledge* to the query to infer related terms;
- By integrating the *background* information of the query;
- By exploiting some characteristics related to the collection.

We will describe these aspects in more details:

---

[5] Searched on May 28, 2007.

- **Automatically expand the user query description by applying knowledge;**

The problem of incomplete specification of information need has been studied for a long time in IR. Most research work proposes to use query disambiguation or expansion to make the query more complete and less ambiguous. However, many methods for query expansion try to determine related terms to be added according to simplistic relations between terms, which are highly ambiguous. For example, for the query "*hotel in Java*", the two words "*hotel*" and "*Java*" are used separately to suggest two sets of expansion terms. It is possible that the inappropriate word "*programming*" will be added because of its strong relationship with the word "*Java*", even if it is unrelated to this particular query.

In our study, we propose the creation of less ambiguous term relations for query expansion. In particular, we will embed some context words in the relations. For example, in our term relations, the word "*hotel*" will be considered together with "*Java*" to suggest other related terms for the query "*Java hotel*". The addition of such a context word in the relation will limit the applicability of the relation to queries that contain both "*Java*" and "*hotel*". It is expected that the suggested expansion terms are more related to the query. This will lead to an approach of *context-dependent query expansion*. As we will describe later, the context-dependent term relations will be extracted from the document collections.

- **Incorporating more contextual factors into the process of matching documents with the query.**

If we are aware of the context of the user when he issues a query, e.g., his goal, his background, his domain of interests, etc., these factors should also

be taken into account. Our example of "*bus service in Java*" is related to this aspect.

Much effort has been spent to consider these factors, mostly in attempts to personalize search engines and IR systems [18][27][36][48]. Personalized IR tries to favor the documents that are related to the general domains of interests of the user. However, in these approaches, a unique user profile, which mixes up all the domains of interest of the user, is created. For our "*bus service in Java*" example, if the user is interested in both *Travel* and *Computer Science*, then the user profile would be of little help because documents relating to both "*transportation in Java Island*" and "*Java language*" will be favored.

In this study, we will use a different approach: instead of modeling the user as a whole, we will model each of its domains of interest separately. When a query is submitted, the appropriate domain of interest will be determined according to the domain models, and the corresponding domain model will be used in the retrieval process.

The central idea we propose in this study is to exploit query contexts. This idea is not new. However, the exploitation of contexts has been relatively limited. In this study, we will design a different method to integrate contexts into IR operations. In particular:

- **We will distinguish two types of context: *context within query* (or *intra-query context*) and *context around query* (or *extra-query context*);**

  The former means the context specified by the words that occur in the query, which provide useful information for the interpretation of the query or the query terms. The latter means the contextual factors in the environment of the retrieval, such as the user's domain of interest, the characteristics of the document collection, etc. To our knowledge, the concept of context within query is quite new and has not been exploited previously as a contextual factor in IR.

- **Multiple contextual factors will be integrated into a uniform framework;**

  Previous studies usually integrate one contextual factor. We will integrate multiple factors to enhance the query using different resources: general knowledge, domain of interest, collection characteristics related to the query topic (through feedback documents).

- **We will use statistical language modeling (LM) as our basic integration framework.**

  This choice is motivated by the solid theoretical foundation of the framework, its ability to deal with incomplete and noisy data, as well as its flexibility to be extended to integrate more criteria. We will show that LM framework can be extended to take into account more contextual factors. However, this study will also show its limitations. What we will observe in this study is that LM can integrate some contextual factors in a straightforward manner (see Chapter 4). However, for a more refined integration, more sophisticated mechanism is required.

Our approaches will be tested on several TREC collections. The experiments aim to validate the following hypotheses:

- The context-dependent query expansion method that we propose is more effective than the previous context-independent query expansion methods;
- Both contexts within and around query can improve retrieval effectiveness, and they can be combined to produce further improvements.

1.3 Contributions

This study aims to make contributions on the following aspects:

- **Identification of a new type of context – context within query:**

Contextual factors have been revealed in a number of studies in IR. However, the factors are all about the environment of the query, e.g., the goal of the search, the background of the user, etc. In this study, we will show that another type of context exists within the query itself, and it is very useful in IR.

To our knowledge, it is the first time that this type of context is identified and considered in IR.

- **Using multiple domain models – context around query:**

In order to take into account the diverse interests of a user, we propose to model each domain of interest separately. When a user issues a query, the appropriate domain of interest will be determined (either manually by the user or automatically by the system) and used in the retrieval process. In this thesis, we will show that it is possible to classify a user query into related domains automatically, and such an approach using domain models is both feasible and effective. This approach is different from most of previous methods on personalized IR.

- **Multiple contextual factors:**

In previous experiments using retrieval contexts, only one type of context is considered usually. In this study, we investigate the integration of several contextual factors at the same time.

- **Language modeling as an appropriate implementation framework:**

Language modeling has been widely used in IR in recent years. However, few contextual factors have been integrated in it. In this study, we show that language modeling offers an appropriate framework that can be extended to integrate contextual factors of different types.

1.4 Organization of the Thesis

The remainder of this thesis is organized as follows:

In Chapter 2, we will describe the basic notions of IR as well as the main processes.

In Chapter 3, we will revisit the key notions of information need, query, relevance and relevance judgment. This will further motivate our approach using various contextual factors.

In Chapter 4, we will describe the general language model that we propose. Two main components of the general model – context-dependent query expansion and domain model – will be further described respectively in Chapter 5 and Chapter 6.

In Chapter 7, we will describe two simple methods for parameter training in our experiments.

Chapter 8 describes our experiments on several TREC collections.

Finally, general discussions about this study and some conclusions will be given in Chapter 9.

Chapter 2
Traditional Information Retrieval

In order to situate our study in IR environment, let us first describe the basic concepts of information retrieval, as well as the basic retrieval approaches in this chapter.

2.1 Basic Concepts of Information Retrieval

Information retrieval (IR) aims to retrieve relevant information from a large collection of documents in order to satisfy the user's information needs, which is usually expressed as a query in natural language. Salton and McGill defined IR as follows [79]:

"*Information retrieval is concerned with the representation, storage, organization, and accessing of information items. Items found in retrieval systems are characterized by an emphasis on narrative information. Such narrative information must be analyzed to determine the information content and to access the role, each item may play in satisfying the information needs of the users.*"

The ultimate goal of IR system is to retrieve all the relevant documents according to a user query while rejecting non-relevant ones. An information retrieval

system should provide the users with a convenient and effective means to access the information that they are interested in. There are three basic concepts involved in IR: *document*, *query* and *relevance*.

- **Document**:

A *document* can be a text, a segment of text, a Web page, an image, a video, a piece of music, and so on. We call document any unit which can constitute a response to a query of the user. In this thesis, we only deal with the textual documents, and documents designate whole texts (we do not intend to retrieve portions of texts). In this situation, we can also consider a document as a sequence of words (terms) without particular structure, written in a natural language such as English. A *term* is a unit extracted from a document in an indexing process, which represents a part of the document content.

- **Information need and Query**:

Whenever a user needs to find information for whatever purpose, there is an *information need*. The user then needs to interact with an IR system in order to find new information. To do this, the user has to express his information need by a query, which can be a formal expression (e.g., Boolean expression) or a free sequence of terms. In this study, we consider the latter case: a query is a free text expression.

- **Relevance**:

Relevance is the central concept in IR. It expresses the relation between the desired document and the information need. However, the concept of relevance is very complex, because users of IR systems have very different needs and very different criteria to judge whether a document is relevant or not. Therefore, the concept of relevance always covers a very vast range of criteria and relations. As we will see in more details in Chapter 3, besides the query,

relevance judgment also depends on many contextual factors, such as the goal of the search, the background of the user, and so on.

In order to determine the documents to be retrieved, the general approach is to carry out an indexing process on both documents and queries. This process described in more details below, produces a set of weighted indexes to represent their contents. Then a degree or a score of relevance is determined according to the correspondence between the representations of document and query. This score is computed during the retrieval process. *Indexing* and *retrieval* processes are two main steps in IR. We will describe them in more details in the following sections.

2.2 Indexing Process

As a document is usually written in natural language, it cannot be compared directly with a query to estimate its relevance. An internal interpretation needs to be created. Indexing is a key process in IR which converts the document from a natural language into an internal representation. The goal of indexing is to recognize the content of a text, and to represent it by a representation manipulable by computers. Usually, the internal representation is based on keywords.

The main tasks involved in this process are: *term selection* and *term weighting*. These tasks are often performed together.

2.2.1 Term Selection

Not all words are equally significant for representing the contents of a document. It is necessary to process the text of the documents in the collection to determine the terms to be used as index terms. The first step of indexing aims to select a set of

meaningful elements from the text and consider them as indexes in IR systems. The extraction process usually considers the following aspects:

- **Tokenization**:

  This process converts a stream of characters (document text) into a stream of words (the candidate words to be adopted as index terms). Normally, it recognizes spaces and punctuation marks as word separators.

- **Stopword removal**:

  Words which are very frequent in a document collection and function words are not good descriptors of document contents. A set of function words (e.g., prepositions, articles) and domain-specific frequent words (e.g., "*document*" for some collections) are considered to convey no useful semantic information for retrieval. These words are called *stopwords*, and they are filtered out from potential index terms. A stoplist contains the set of stopwords (e.g., "*the*", "*of*", "*and*"). It is built up manually for each language.

- **Word stemming**:

  Words of different forms do not necessarily mean different things. In English and other European languages, words with the same root or lemma typically convey similar semantic information. For example, "*retrieval*", "*retrieve*", "*retrieving*", etc. are all related to the meaning of "*retrieval*". If these word forms are used as different indexes, documents containing each of them will be considered to be completely different. In reality, we would rather consider these words to be similar. Therefore, a stemming process is usually employed to eliminate the meaningless differences in form, and to transform these word forms to the same root form (e.g., "*retriev*").

  Stemming tries to remove the ending (or suffix) of a word according to manually established patterns. For example, the pattern:

$$A + \text{“}ing\text{”} \rightarrow A$$

will remove the ending “*ing*” from “*retrieving*”, resulting in the root form “*retriev*”. The most widely used stemming algorithm in IR is the *Porter* algorithm [70].

In some cases, a simple stemming may wrongly unify words with different meanings, such as “*tube*” and “*tub*”, “*feat*” and “*feature*”, etc. A more complex word transformation can use more linguistically motivated method, by examining the syntactic category of the word, and transforming it to a root form (or citation form) according to its category. For example, Savoy [85] used such an approach for French. The word “*porte*” (door) will be normalized to “*porter*” when it is a verb and remains unchanged when it is a noun.

After these steps, the remaining words are usually considered as potential indexes of a text.

2.2.2 Term Weighting

Once index terms are determined, it is also important to determine an appropriate weight to each term (index). The weight of a term should reflect how representative the term is for the content of a document.

It is generally believed that the more a term appears in a document, the more it can represent the content of the document [60]. This is the *TF* (term frequency) factor. On the other hand, a frequent term in a document may not be specific to that document. It can also appear frequently in many other documents. In this case, the term does not have a high discrimination value allowing us to distinguish a document from the others. Therefore, such a term should not be attributed a large weight. So a second factor *IDF* (inversed document frequency) is added [98]. The best descriptors are words that have both high information value (*TF*) and discrimination value (*IDF*). Therefore, the

common method is to combine the above two factors to measure the important terms for document.

Below is one of the typical *tf*idf* weighting schemes used in IR:

$$w(t,D) = tf(t,D) \times \log\left(\frac{N}{n_t}\right)$$

where $tf(t,D)$ is the frequency of term t in document D, N is the number of documents in the whole collection, and n_t is the number of documents containing term t.

There are several variants of *tf*idf* weighting schemes. For example, the following weight of term t in document D uses the cosine normalization (the denominator factor), which corresponds to the *tfc* weighting schema in the *Smart* system [78]:

$$w(t,D) = \frac{tf(t,D) \times \log\left(\frac{N}{n_t}\right)}{\sqrt{\sum_{t \in V}\left(tf(t,D) \times \log\left(\frac{N}{n_t}\right)\right)^2}}$$

where V is the whole vocabulary.

2.3 Retrieval Models

Once the indexing has been carried out, the next problem is to determine the degree of correspondence between a document and a query. During retrieval process, the query is compared with each document representation in order to estimate the relevance between them. The general approaches usually depend on how many words are shared between the documents and query representations, and how important these common words are.

Below, we will describe some of the existing retrieval models: *Boolean model*, *vector space model*, *probabilistic model* and *statistical language model*.

2.3.1 Boolean Model

In the classical Boolean model, a document is represented as a logic conjunction of terms: $D: t_1 \wedge t_2 \wedge ... \wedge t_n$. The terms are those that appear in the document. This is indeed a binary weighting of terms: terms that appear in the document are weighted 1, and terms that are absent are weighted 0.

A query is a Boolean expression of terms, such as $Q: (t_1 \wedge t_2) \vee t_4$.

To determine if a document should be retrieved, one has to determine if the following logic implication is valid: $D \rightarrow Q$. Documents D which imply the query Q are retrieved.

As we can see, this basic Boolean model has several problems:

- Terms are not attributed with appropriate weights. Binary weights are often too coarse-grained to reflect the importance of terms in a document and in a query;
- The whole evaluation basically examines if the required terms appear in a document. The final result is a set of documents, which imply the query. However, no ranking is made among them. The user can be left with too few documents (with a hard query) or too many documents (with an easy query);
- Even though the model is based on Boolean logic, there is indeed no inference in the retrieval process. For example, a document about "*text retrieval*" will be judged to be irrelevant to a query on "*document* $\wedge$ *retrieval*", since "*text* $\wedge$ *retrieval* $\rightarrow$ *document* $\wedge$ *retrieval*" is

an invalid logic expression. This is because no relation between "*document*" and "*text*" is taken into account.

The classical Boolean model has been extended on the above aspects. For example, term weighting has been integrated by using fuzzy logic [49][74] or *p*-norm [80], and documents can be ranked. Related words can also be viewed as having a logic implication relation; such relation can be used to expand queries (we will see more on this later).

2.3.2 Vector Space Model

Vector space model is one of the most commonly used models to measure the similarity of a document and a query. In this model, the document and query are represented as *n*-dimension vectors, where *n* is the number of all the indexed terms. A vector space is determined by all the index words selected from the entire document collection. A value in a document (query) vector denotes the importance of the corresponding word in that document (query). In other words, given a vector space as follows:

$$\text{vector space: } \langle t_1, t_2, \ldots, t_n \rangle$$

A document and a query may be represented as the following vectors of weights:

$$D: \langle w_{d_1}, w_{d_2}, \ldots, w_{d_n} \rangle$$

$$Q: \langle w_{q_1}, w_{q_2}, \ldots, w_{q_n} \rangle$$

where w_{d_i} and w_{q_i} are the weights of t_i in document D and query Q respectively.

Query matching involves measuring the degree of similarity between the query vector Q and each document vector D. The vector space model evaluates the degree of

similarity $sim(D,Q)$ between each document and the query according to the similarity between their vectors. The document which is the most similar to the query is ranked the highest, and will be considered to be the most relevant to the query. The common way to measure the similarity is by the following *inner product* or *cosine formula*:

$$\text{inner product:} \qquad sim(D,Q) = \sum_{i=1}^{n} w_{d_i} \times w_{q_i}$$

$$\text{cosine formula:} \qquad sim(D,Q) = \frac{\sum_{i=1}^{n} w_{d_i} \times w_{q_i}}{\sqrt{\sum_{i=1}^{n} (w_{d_i})^2 \times \sum_{i=1}^{n} (w_{q_i})^2}}$$

Vector space model has been very popular due to its clear mathematical interpretation and simplicity to implement. It has also proven to be an efficient and effective model.

However, we also notice that the basis of this model is the assumption of a vector space, in which dimensions are mutually independent. In practice, every word (or stem) encountered in the document collection is used as a dimension. This means that each word is assumed to represent a different dimension (and meaning). The similarity function only considers the terms that appear in the query. A term in a document, which is different from the terms in the query, does not have any significant impact (except the normalization effect) on the similarity value. This is indeed a process of keyword matching.

The independence assumption made here is not true in reality. In fact, different terms may have the same meaning (synonymy), and a term may also have different meanings (polysemy). The traditional vector space model is unable to account for these problems. Several approaches have been proposed to solve these problems, including constructing a new vector space (as in LSI – latent semantic indexing [26]) and query expansion (see Section 3.3.2).

2.3.3 Probabilistic Models

Another family of models tries to capture the notion of relevance and irrelevance and to rank a document according to its probability to be relevant and irrelevant. Let us use *Rel* and *NRel* to represent respectively relevance and non-relevance. The key problem is the estimation of $P(Rel|D)$ and $P(NRel|D)$, i.e., the degree of relevance and irrelevance of a document D. Notice that *Rel* and *NRel* are dependent on the query (or the information need), so *Rel* and *NRel* indeed represent Rel_Q and $NRel_Q$; $P(Rel|D)$ and $P(NRel|D)$ are indeed $P(Rel|D,Q)$ and $P(NRel|D,Q)$. However, we will ignore this index in the following discussions.

The quantities $P(Rel|D)$ and $P(NRel|D)$ are often difficult to estimate directly. We then use Bayes rule to transform them as follows:

$$P(Rel \mid D) = \frac{P(D \mid Rel)P(Rel)}{P(D)}$$

$$P(NRel \mid D) = \frac{P(D \mid NRel)P(NRel)}{P(D)}$$

To determine if a document is to be retrieved, we are indeed making a binary decision: whether the retrieval of a document is useful or not. Here we assume that a relevant document is useful while an irrelevant one is not. Related to the decision theory, this decision can be made according to the following Bayes decision rule:

If $P(Rel|D) > P(NRel|D)$ then D is relevant, otherwise, it is irrelevant.

Then the decision can also be made according to the following ratio:

$$ratio(D) = \frac{P(Rel \mid D)}{P(NRel \mid D)} = \frac{P(D \mid Rel)P(Rel)}{P(D \mid NRel)P(NRel)}$$

It is showed that if documents are ranked in the reverse order of this ratio, the effectiveness of the system is optimized (probability ranking principle) [75], or the user can obtain the highest utility.

We assume that *P*(*Rel*) and *P*(*NRel*) are dependent only on the given query and the document collection. They are the same for all the documents in the collection. Thus they do not affect the ranking of different documents and can be ignored in the ranking function. Therefore, document ranking can be made according to the following *retrieval status value* (*RSV*):

$$RSV(D) = \log \frac{P(D \mid Rel)}{P(D \mid NRel)}$$

Below, we will describe two existing probabilistic models: *Binary Independent Model* and *BM25*.

2.3.3.1 Binary Independent Model

We now consider that each document D is represented by a set of binary events, which are the presence or absence of different terms. Let us use x_i to denote the presence (when $x_i = 1$) or absence (when $x_i = 0$) of a term t_i in document D. Then D can be represented as follows:

$$D = (x_1, x_2, ..., x_n)$$

Then $P(D|Rel)$ can be calculated as follows:

$$P(D \mid Rel) = P(x_1 \mid Rel)P(x_2 \mid x_1, Rel)...P(x_n \mid x_1, x_2, ..., x_{n-1}, Rel)$$

However, this non-simplified form is very difficult to calculate because of the dependencies between terms. To simplify the calculation, it is then assumed that the presence or absence of a term is independent from those of other terms. We then arrive at the binary independent model and $P(D|Rel)$ can be simplified as follows:

$$P(D \mid Rel) = \prod_{i=1}^{n} P(x_i \mid Rel)$$

Similarly:

$$P(D \mid NRel) = \prod_{i=1}^{n} P(x_i \mid NRel)$$

The key problem becomes that of estimation of $P(x_i \mid Rel)$ and $P(x_i \mid NRel)$. The traditional way to estimate them is to assume that we have a set of samples of relevant and irrelevant documents for each query. For example, one can acquire such sets of samples through relevance feedback, i.e., the user indicates whether a document in the retrieval result is relevant or not. With a set of sample relevant and irrelevant documents, we can establish the following contingency table for x_i (where we assume that we have N samples, and each of the number represents the number of relevant/irrelevant documents which contain or do not contain the term):

Table 2. Relevant contingency table

	Relevant	**Irrelevant**	**Total**
$x_i = 1$	r_i	$n_i - r_i$	n_i
$x_i = 0$	$R_i - r_i$	$N - R_i - n_i + r_i$	$N - n_i$
Total	R_i	$N - R_i$	N

Then $P(x_i \mid Rel)$ and $P(x_i \mid NRel)$ can be estimated as follows:

$$P(x_i = 1 \mid Rel) = \frac{r_i}{R_i}$$

$$P(x_i = 0 \mid Rel) = \frac{R_i - r_i}{R_i} = 1 - P(x_i = 1 \mid Rel)$$

$$P(x_i = 1 \mid NRel) = \frac{n_i - r_i}{N - R_i}$$
$$P(x_i = 0 \mid NRel) = \frac{N - R_i - n_i + r_i}{N - R_i} = 1 - P(x_i = 1 \mid NRel)$$

We can also represent $P(x_i \mid Rel)$ and $P(x_i \mid NRel)$ as follows:

$$P(x_i \mid Rel) = P(x_i = 1 \mid Rel)^{x_i} P(x_i = 0 \mid Rel)^{1-x_i}$$
$$P(x_i \mid NRel) = P(x_i = 1 \mid NRel)^{x_i} P(x_i = 0 \mid NRel)^{1-x_i}$$

Then we can derive:

$$RSV(D) = \log \frac{P(D \mid Rel)}{P(D \mid NRel)}$$
$$= \log \frac{\prod_{i=1}^{n} (\frac{r_i}{R_i})^{x_i} (\frac{R_i - r_i}{R_i})^{1-x_i}}{\prod_{i=1}^{n} (\frac{n_i - r_i}{N - R_i})^{x_i} (\frac{N - R_i - n_i + r_i}{N - R_i})^{1-x_i}}$$
$$= \sum_{i=1}^{n} x_i \log \frac{r_i (N - R_i - n_i + r_i)}{(n_i - r_i)(R_i - r_i)} + \sum_{i=1}^{n} \log \frac{(R_i - r_i)(N - R_i)}{R_i (N - R_i - n_i + r_i)}$$

The last term in the above expression is independent of the document (i.e., x_i). Therefore, it can be ignored. Documents can then be ranked according to the first term.

As samples are limited (if ever available), a form of smoothing is necessary. Robertson and Sparck Jones [76] proposed the following formula of smoothing:

$$w(x_i) = \log \frac{(r_i + 0.5)(N - R_i - n_i + r_i + 0.5)}{(n_i - r_i + 0.5)(R_i - r_i + 0.5)}$$

Then we have:

$$RSV(D) = \sum_{i=1}^{n} x_i w(x_i)$$

where x_i indicates whether a term appear in the document or not, and $w(x_i)$ indicates its weight.

2.3.3.2 BM25

In the above weighting schema by Robertson and Sparck Jones, a number of factors have been ignored, such as the frequency of the term in the query and in the document, and the length of the document. These latter factors have been shown to be important in IR. In order to integrate them into the weighting schema, Robertson et al. have developed a series of improved weighting schemas, called *BM* (best match). One of the proposed schema, called *BM25*, weights a term as follows [77]:

$$BM25(x_i) = \frac{(k_3+1)tfq_i}{(k_3+tfq_i)} \frac{(k_1+1)tf_i}{(k_1(1-b+b\frac{|D|}{avdl})+tf_i)} \log \frac{(r_i+0.5)(N-R_i-n_i+r_i+0.5)}{(n_i-r_i+0.5)(R_i-r_i+0.5)}$$

where tfq_i and tf_i are the frequency of the term t_i in the query and in the document, $|D|$ is the length of document and *avdl* is the average length of documents in the collection, and $k_1 \in [1.0; 2.0]$, b (usually set at 0.75), and $k_3 \in [0; 1000]$ are constants.

In general, IR users do not provide explicit relevance judgments. It is then often the case that we do not have any sample for the estimation of $P(x_i \mid Rel)$ and $P(x_i \mid NRel)$. In other words, $r_i = R_i = 0$. We use the whole collection to approximate the set of irrelevant documents. This approximation can be made because the collection contains far more irrelevant documents than relevant ones. Therefore, *BM25* becomes as follows:

$$BM25(x_i) = \frac{(k_3+1)tfq_i}{(k_3+tfq_i)} \frac{(k_1+1)tf_i}{(k_1(1-b+b\frac{|D|}{avdl})+tf_i)} \log \frac{N-n_i+0.5}{n_i+0.5}$$

Notice that the term $\log\frac{N-n_i+0.5}{n_i+0.5}$ corresponds roughly to the *IDF* factor we described earlier (i.e., $\log\frac{N}{n_i}$).

Although BM25 has produced good results in experiments, the addition of new factors is made according to heuristics. The way that they are introduced into the above formula can only be justified heuristically or empirically.

There have been several other forms of probabilistic models [31] such as BII (binary independent indexing model) and 2-Poisson model, which assumes a different distribution norm. We can observe that in all these models, terms are always assumed to be independent. An interesting exception on this aspect is the *dependence tree model* [102]. Instead of assuming strong independence among terms, Van Rijsbergen proposed to capture the most important link between a term and another. The whole dependence relations form a tree structure as follows:

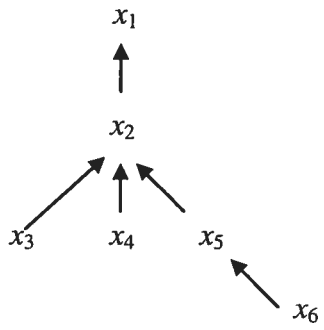

Figure 2. Dependence relations tree structure (from [103])

Then the joint probability $P(x_1, x_2, x_3, x_4, x_5, x_6)$ is estimated as follows:

$$P(x_1, x_2, x_3, x_4, x_5, x_6) = P(x_1)P(x_2 \mid x_1)P(x_3 \mid x_2)P(x_4 \mid x_2)P(x_5 \mid x_2)P(x_6 \mid x_5)$$

Van Rijsbergen proposed to use maximum spanning tree to determine the best term dependencies. Maximum spanning tree is a tree that contains the strongest corrections, which cover all the terms in a set. It is similar to minimal spanning tree.

This dependence model has been successfully used on small collections, but for larger collections, we encounter the problem of complexity to determine such trees for documents [42].

2.3.4 Statistical Language Models

Language models have been successfully applied in many application areas such as speech recognition and statistical natural language processing (NLP). Recently, a number of researches have confirmed that language modeling is also an effective and attractive approach for information retrieval [62][43][50][52][69][97][115]. It not only provides an elegant theoretical framework to IR, but also results in good effectiveness comparable to the state-of-the-art systems.

In the following sections, we will describe the two score functions that are normally used in language modeling approach, and the important process in LM: *model smoothing*.

2.3.4.1 Query Likelihood

The language modeling approach to information retrieval was first introduced by Ponte and Croft in [69] and later explored in different studies. The basic IR approach based on LM constructs a language model for each document D, i.e., a probability distribution $P(t_i \mid \theta_D)$ among all the terms t_i in the vocabulary V. The score of relevance of a document D is determined by its probability to generate the query Q, or the likelihood of Q in the document model. Suppose we have a query:

$$Q = t_1 t_2 ... t_n$$

By assuming independence between query terms, the query likelihood is determined as follows:

$$\begin{aligned} P(Q \mid \theta_D) &= \prod_{i=1}^{n} P(t_i \mid \theta_D) \\ &\propto \sum_{i=1}^{n} \log P(t_i \mid \theta_D) \\ &= \sum_{t_i \in Q} \log P(t_i \mid \theta_D)^{tf(t_i, Q)} \end{aligned}$$

The document model is a unigram model, which does not consider any relation between terms. One can also use a bigram model instead of unigram model. Then the query likelihood can be determined as follows:

$$P(Q \mid \theta_D) = P(t_1 \mid \theta_D) \prod_{i=2}^{n} P(t_i \mid t_{i-1}, \theta_D)$$

This model assumes that a term depends on its precedent term. However, experiments showed that this bigram model only outperforms marginally unigram model, while the model is much more complex [97]. Therefore, the state of the art still uses unigram models.

2.3.4.2 KL-Divergence

Another widely used score function is based on *Kullback-Leibler divergence* (*KL*-divergence) or *cross entropy*, which determines document ranking according to how the document model is close to the query model. The corresponding score function is defined as follows:

$$score(Q,D) = -KL(\theta_Q \parallel \theta_D)$$
$$= \sum_{t_i \in V} P(t_i \mid \theta_Q) \times \log \frac{P(t_i \mid \theta_D)}{P(t_i \mid \theta_Q)}$$
$$\propto \sum_{t_i \in V} P(t_i \mid \theta_Q) \times \log P(t_i \mid \theta_D)$$

where θ_D and θ_Q are the document and the query language model respectively. In the transformation from the second line to the last line, we dropped a constant $\sum_{t_i \in V} P(t_i \mid \theta_Q) \times \log P(t_i \mid \theta_Q)$, which is independent from the document, so does not affect the ranking of different documents.

Besides the direct interpretation of closeness of models, we can also explain the above score in another way as the following *log-Likelihood Ratio* [62]:

$$LR(D,Q) = \log \frac{P(D \mid Q)}{P(D)} = \log \frac{P(Q \mid D)}{P(Q)}$$

The interpretation of the above expression is that a document D is ranked according to the change of its likelihood before and after the query is given (i.e., the change between $P(D)$ and $P(D|Q)$), or to the change of the query likelihood before and after the document is retrieved (i.e., the change between $P(Q)$ and $P(Q|D)$). In both cases, the more there is a change, the more it is believed that the document has an impact on the query or vise versa. So the document is closely related to the query, and should be ranked high.

In the above equation, we can further use a unigram model $P(Q \mid \theta_D) = \prod_{i=1}^{n} P(t_i \mid \theta_D) = \prod_{t_i \in V} P(t_i \mid \theta_D)^{tf(t_i,Q)}$ to estimate $P(Q|D)$, and we use the collection model $P(Q \mid \theta_C) = \prod_{i=1}^{n} P(t_i \mid \theta_C) = \prod_{t_i \in V} P(t_i \mid \theta_C)^{tf(t_i,Q)}$ as an approximation of $P(Q)$. Therefore:

$$
\begin{aligned}
LR(D,Q) &= \sum_{t_i \in V} tf(t_i,Q) \times \log \frac{P(t_i \mid \theta_D)}{P(t_i \mid \theta_C)} \\
&\propto \sum_{t_i \in V} \frac{tf(t_i,Q)}{|Q|} \times \log \frac{P(t_i \mid \theta_D)}{P(t_i \mid \theta_C)} \\
&= \sum_{t_i \in V} P(t_i \mid \theta_Q) \times \log \frac{P(t_i \mid \theta_D)}{P(t_i \mid \theta_C)} \\
&\propto \sum_{t_i \in V} P(t_i \mid \theta_Q) \times \log P(t_i \mid \theta_D)
\end{aligned}
$$

which is the same as the score based on *KL-divergence* or *cross-entropy.*

2.3.4.3 Smoothing

In previous studies, it turns out that *smoothing* is a very important process in building a language model [116]. The effectiveness of a language modeling approach is strongly dependent on the way that the document language model is smoothed. The primary goal of smoothing is to assign a non-zero probability to the unseen words in a document. Without smoothing of the document model, the probability is estimated by *maximum likelihood estimation* (MLE) or relative frequency. This means that a query term that does not appear in a document will have zero probability in the model of that document. As a consequence, such a document will not be retrieved (either according to the query-likelihood formula or KL-divergence score). This is not reasonable in IR. In fact, a document can be relevant even if some query terms are absent from it. For example, the document can contain some similar or related terms. To solve this problem, we have to perform model smoothing, which will increase the probability from 0 to some small value. It is shown that smoothing can naturally incorporate some term weighting factors such as *tf*idf* and document length [117].

Two common smoothing methods are the *Jelinek-Mercer* interpolation smoothing and *Dirichlet* smoothing:

$$\textit{Jelinek-Mercer}: \quad P(t_i \mid \theta_D) = \lambda P_{ML}(t_i \mid \theta_D) + (1-\lambda) P_{ML}(t_i \mid \theta_C)$$

$$\textit{Dirichlet}: \qquad P(t_i \mid \theta_D) = \frac{tf(t_i, D) + \mu P_{ML}(t_i \mid \theta_C)}{|D|_U + \mu}$$

where λ is the interpolation parameter and θ_C is the collection model, $tf(t_i, D)$ is the term frequency of t_i in D, $|D|_U$ is the number of unique terms in the document, and μ is the Dirichlet prior (or pseudo count). Both λ and μ can be tuned empirically using a training collection. The parameter λ can also be tuned automatically so as to maximize the likelihood of a set of feedback documents [116]. We will provide more details on this later.

More smoothing methods are used in statistical NLP [17]. Lafferty and Zhai [116] carried out an empirical comparison on several smoothing methods for IR. It turns out that for short queries, *Dirichlet* smoothing is better than *Jelinek-Mercer* smoothing, while for long queries, the opposite is observed.

Although we have both document and query models in the formulation based on KL-divergence, in the basic language modeling approach, only the document model is smoothed, while the query model is estimated by maximum likelihood estimation i.e., $P_{ML}(t_i \mid \theta_Q) = {tf(t_i, Q)}/{|Q|}$, without any smoothing, where $tf(t_i, Q)$ is the term frequency of t_i in Q. Then we have:

$$score(Q, D) = \sum_{t_i \in Q} P_{ML}(t_i \mid \theta_Q) \times \log P(t_i \mid \theta_D)$$

This score produces the same ranking as the query-likelihood score. In fact, we have:

$$
\begin{aligned}
score(Q,D) &= \sum_{t_i \in Q} P_{ML}(t_i \mid \theta_Q) \times \log P(t_i \mid \theta_D) \\
&= \sum_{t_i \in Q} \frac{tf(t_i,Q)}{|Q|}) \times \log P(t_i \mid \theta_D) \\
&\propto \sum_{t_i \in Q} tf(t_i,Q) \times \log P(t_i \mid \theta_D) \\
&= \sum_{t_i \in Q} \log P(t_i \mid \theta_D)^{tf(t_i,Q)}
\end{aligned}
$$

The last expression is indeed the log likelihood of the query according to the document model.

The reason to use MLE for query model is due to the fact that, in the traditional setting, there is not much information available to define a query model, except term frequency in the query. This is related to the assumption that query is the only element available about information need.

In practice, this MLE model for query also has the advantage of limiting the number of terms to look at during the retrieval process, thus reducing the complexity of the query evaluation process: instead of making a summation over all $t_i \in V$, one can reduce it to $t_i \in Q$.

However, as in the other models, we can also observe in the above formula that the score is still determined solely by the terms that appear in the query ($t_i \in Q$). So the basic operation in the language modeling approach remains that of keyword matching.

The above description concerns the basic language models used in IR. In Section 3.3.4, we will describe several extensions to them, which try to smooth either document model or query model using different strategies and resources.

2.4 Evaluation

It is important to evaluate a retrieval approach or system on their quality. To do this, we should have a test collection and define some measures of quality.

2.4.1 Test Collection

A test collection in IR should contain three types of data:

- A set of documents, called document collection or corpus;
- A set of user queries (or topics) that specify the topics of information needs;
- The relevance judgments of each document with respect to each query or information need.

An IR approach or system to be tested is run on the test collection: the set of documents are indexed then retrieved using each of the queries. The retrieval results are compared against the standard relevance judgments in order to measure the effectiveness.

Several test collections have been created in the TREC [6] (Text REtrieval Conference), CLEF[7] (Cross-Language Experiment Forum) and NTCIR [8] experiments. In this study, we will use the collections from TREC.

---

[6] http://trec.nist.gov/

[7] http://www.clef-campaign.org/

[8] http://research.nii.ac.jp/ntcir/

2.4.2 Performance Measures

The performance of an IR approach or system is measured on two aspects: the quality of the answers, the time and space requirements. In most IR tests, we are basically concerned with the quality of the answers, or the effectiveness of the approach or system. The time and space requirements of a system are believed to be less critical, as computers become more and more powerful, and can eventually satisfy the time and space requirements, provided that the requirements remain in a reasonable range. In our experiments, we will also evaluate the methods with respect to the quality of retrieval results.

With respect to a given query, the collection of the documents can be divided into four groups according to the relevance and the retrieval result.

Table 3. Relevant-retrieved document contingency table

	relevant	**non-relevant**
retrieved	A	B
non-retrieved	C	D

The standard measures of retrieval effectiveness are *precision* and *recall.*

- **Precision**: *Precision* measures the proportion of relevant documents among all the retrieved documents, i.e.:

$$precision = \frac{A}{A+B}$$

- **Recall**: *Recall* measures the proportion of all relevant documents retrieved by the system, i.e.:

$$recall = \frac{A}{A+C}$$

Precision and recall are not independent. There is a strong relation between them: while one increases, the other decreases. Therefore, we obtain a curve of *precision-recall* as shown in the following figure:

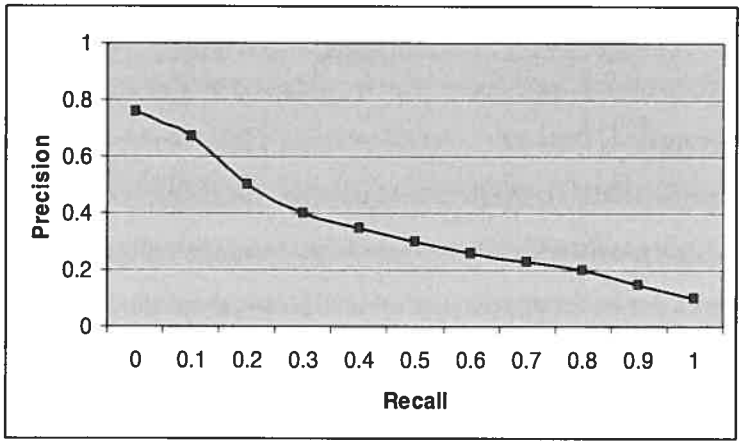

Figure 3. A typical curve of precision-recall

In order to arrive at a single number to measure the effectiveness, one often uses average precision at 11 points of recall, i.e., the average of the precision values at 0.0, 0.1, …, 1.0 recall level. Another average measure is *mean average precision* (*MAP*), which is calculated as follows:

$$MAP = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{|R_i|}\sum_{D_j \in R_i} prec(D_j, Q_i) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{|R_i|}\sum_{D_j \in R_i}\frac{j}{Rank(D_j, Q_i)}$$

where N is the number of test queries (topics) in the collection, R_i is the set of relevant documents to Q_i and $|R_i|$ is its size, and $prec(D_j, Q_i)$ is the precision value at the rank of D_j in the result list of Q_i (by considering all the documents in the result list up to D_j). Both 11-point average precision and MAP can reflect the global effectiveness of the IR system.

In some cases, we also use "*precision at n documents*" and "*recall at n documents*" to measure the precision and recall ratios when the top n documents are retrieved. In particular, when we use a small n, precision at n documents indicates

whether the top few documents are relevant. This measure is often used for Web IR, where it is impossible to know the whole set of relevant documents for a query. The measure of "*precision at n documents*" only requires relevance judgments for the top *n* documents, which is more feasible in practice.

2.5 General Observations on Traditional IR

2.5.1 Keyword as Independent Semantic Representative

In almost all the classical IR systems, each document and query is represented by a set of independent words (terms). No relation has been considered between different terms. This approach is often called "*bag-of-words*" approach. In order to retrieve a document for a query, the document should contain the same words as the query, or at least some of them. For example, in vector space model, the more a document shares words with the query, the higher its similarity is to the query.

We can observe here a gap between the goal of IR and its models: the goal of IR is to retrieve documents of certain meaning, while IR system is usually implemented as retrieving documents based on direct keyword matching. Indeed, there is not a strict 1:1 relationship between words and meanings. A word can denote different meanings (polysemy) in different contexts, and a meaning may be expressed by different words (synonymy). The system may both miss the relevant documents and retrieve the irrelevant ones. For example, a search using the word "*Java*" could return documents on "*coffee*", "*travel*" and "*programming language*" without differentiating the different contexts of the word "*Java*". On the other hand, a document on "*Unix*" may not be returned for a query on "*operating system*" if the words "*operating system*" is absent in that document.

The limitation of classical IR is largely due to the keyword-based representation of documents and queries and the assumption of independence between keywords. The underlying problems with this representation have been known for a long time. Indeed, such a representation and the corresponding retrieval process are unable to handle the polysemy and synonymy problems.

Various approaches have been proposed to address these two problems:

Latent semantic indexing (LSI) [26] is one such approach proposed. In LSI, one tries to construct another vector space from the original document-term matrix, in which each of the dimensions is a linear combination of the dimensions (both document and term) of the original matrix. The new dimensions are extracted through *singular value decomposition* (SVD). The k strongest singular values are kept while the others are trimmed. The strongest singular values correspond to the strongest orthogonal dimensions that best approximate the original document-term matrix from the point of view of Frobenius norm.

Hofmann [45] proposed a probabilistic variant of LSI, called *pLSI* (probabilistic LSI). Instead of using SVD to determine the k best dimensions, Hofmann uses *expectation maximization* (EM) algorithm to tune the k latent variables so as to maximize the likelihood of the documents in the collection. Once the k new dimensions are determined, a document and a query are compared in the new dimensions.

In both LSI and pLSI, the new dimensions are combined from the original ones. It is expected that similar original terms can be grouped into the same latent dimension, while an ambiguous word would correspond to several latent dimensions according to the terms used together with it. The document-query comparison in the new dimensions can then cope with the polysemy and synonymy problems to some extent.

A large number of studies also try to use word sense disambiguation and query expansion to deal with the problems related to the 1:1 correspondence assumption. We

will review some of them in Section 3.3. Our approach falls into the same category as query expansion.

2.5.2 Query as the Only Information about Information Need

Traditional IR approaches assume that query is the only information available about the information need. If a term is not specified in a query, it is assumed to be unimportant for retrieval. For example, in vector space model, terms that do not appear in a query will not have any impact on the similarity measure of a document.

In reality, a query is only an approximate and incomplete description about the information need. As we will describe in Chapter 3, much useful information is not contained in the query.

Many cognitive studies have found that besides query, information need is also described by many contextual factors. However, the consideration of the contextual factors is a difficult enterprise, due to the fact that the contextual factors are complex and have only been described in informal terms in cognitive studies. It is difficult to integrate them into operational IR systems.

Recent studies have put more efforts on integrating contextual factors in various ways, including personalization IR [27][101]. In these studies, external knowledge is used to complement the query, in order to arrive at a better specification of information need.

In order to see the importance of contextual factors in IR, let us review the basic concepts of information need, query, relevance and relevance judgment in the next chapter. This will further motivate our approach that exploits several types of contextual information.

Chapter 3

Information Need, Query, Relevance and Relevance Judgment

As we discussed in the previous chapter, traditional IR approaches try to mimic user's relevance judgments by a word matching process between document and query. To better understand the problems with this simplified approach, let us go back to some fundamental aspects of IR: *information need* and its expression as *query*, *relevance* and its *judgments*. Our review in this chapter aims to show that query should not be considered as a perfect description of the underlying information need. It is not the only element related to the information need, and many other contextual factors exist, which should also be taken into account. This will naturally suggest a broader context-sensitive approach.

3.1 User Information Need and Query

When a user is in a situation where some information is required for whatever purpose, there is an *information need*. Everyone needs information everyday for different purposes in different tasks. One may need information to learn about a particular concept, a special event, a person, etc. Information is needed when the

available information is insufficient to perform a task. Belkin calls such a state *anomalous state of knowledge* (ASK) [9].

To retrieve relevant documents from an IR system, the user has to formulate his information need as a *query*, which can take forms varying from Boolean expression to free text. As we described in the previous chapter, in traditional IR systems, query is usually assumed to be the only input from the user and no other factors is considered during the retrieval process. In reality, there are many other factors important for determining relevant documents.

To better understand this, it is useful to mention some differences between data retrieval and information retrieval. Data retrieval aims to determine which documents in the collection or data items in a database contain the keywords specified in a user query (e.g., a SQL query), while information retrieval tries to determine documents that are relevant to a user's information need [103]. One of the strongest differences between them is in the expectation of the user from the systems. In data retrieval, the user is responsible for formulating a correct query in a formal query language, and the system is responsible for returning the set of data corresponding to the query. If the returned results do not correspond to the user's need, the responsibility is usually placed on the user. However, in IR, the user only submits an informal query and the system is expected to return the answers that correspond to the user's information need. Even if the query does not describe well the information need, the system is still to be blamed for unsatisfactory answers.

We can see here that an IR system cannot take as granted that the user query is perfect. An important aspect in IR is to figure out "*what the information need behind the query is*", rather than simply relying on the keywords contained in the query.

Information needs can concern various topics [109]. They can arise for different purposes [108] and from different users. The nature of information need has been investigated in a large number of studies in cognitive and library sciences [100][109].

In traditional library environment, librarians play the role of intermediary between the library retrieval system and the user. The librarian's task is not always limited to finding a required book or reference for the user. He can also be an assistant to help the user clarify the information need and to specify the best query. For example, when the user wants to find references on a topic (e.g., a scientific topic), the librarian can start a conversation with the user in order to know the background of the user, the purpose of the search, etc. Based on this information, he can advise some specific resources (reference bases) and formulate specific queries for the user. In this scenario, before a request is formulated in any search system, a conversation takes place between the librarian and the user, during which the librarian tries to learn more about the particular information need of the user and about its contexts. This conversation is often necessary to help specify the information need of the user accurately.

The above scenario also shows that user's information need is not always clearly defined. As there is no longer a human intermediary between the user and an IR system or a search engine, the user has to specify a query on his own. In this situation, we can observe several problems in query specification:

- **The user may not know precisely what he is looking for;**

  Even for the user himself, the information need may be implicit and vague. For example, a user may need to find a method to solve a scientific problem, say "*the theoretical formulations of reasoning processes in an information system*", but he does not know what theoretical formulations to request. What he can tell is the problem itself. However, by specifying the problem only, the information need is only partially defined. A more complete specification would also comprise the particular theoretical formalisms, such as "*Bayesian network*", "*Boolean logic*", "*default logic*", and so on. However, the user may not be aware that these formalisms are related to his information need.

- **The user may not know how to formulate his need or he does not know what are the best words to use;**

Another situation is that the user knows what is needed, but does not know how to describe it. For example, a user may be repairing a car problem. He needs a particular car part, but does not know its name. He is thus unable to specify the information need clearly with the correct terms. This problem is often solved by interacting with an intermediary (e.g., a mechanic or vendor of car parts). Without such an intermediary, the user may choose an inappropriate term.

- **The user may put only some of the relevant words and omit others.**

In most cases, the query is usually a very short natural language description. Only the most important terms, from the user's point of view, are included. However, there may be many other words that correspond to the same concepts. These latter words can also be used in relevant documents.

From the experiments in the *STAIRS* system, Blair [11] observed that "*the variability in the words and phrases which the authors of the documents on the system used to discuss the same topic was extraordinary and unpredictable*" (p. 109), and "*the most significant factor contributing to poor retrieval results in information retrieval is the inquirers' inability to predict the words and phrases which represent all and only the relevant or useful documents*" (p. 112). This observation is particularly true in the Web environment, where queries are a largely reduced expression of information need: only a few keywords are input as a query [46]. It is clear that a query is far from being a complete means of expression to cover all the ways that the topic is described in documents.

We have described in Chapter 2 that a good search term is the one that is also discriminative, i.e., it can distinguish a subset of documents from the others. However, the words selected by the user may not be the most discriminative

ones. Previous studies [66] showed that users tend to use frequent words in their queries. Even if these terms are relevant to the information need, they can also appear in many irrelevant documents. Therefore, they are not the best terms to use in a query.

All these factors often make the query a poor expression of an information need. Several common problems occur in user queries:

- A user query contains some terms related to the topic, but these terms are neither necessarily the best ones nor all the possible ones for the retrieval purposes;
- The terms included in the query can be ambiguous.

Many previous studies have tried to deal with these problems, mainly by query disambiguation or query expansion. We will review some common approaches later in this chapter.

3.2 Relevance and its Judgments by the User

3.2.1 The Concept of Relevance

The ultimate goal of IR systems is to retrieve relevant documents for a user's information need. All the IR functions should be designed in a way so as to reproduce the relevance judgments of the user as much as possible. The concept of *relevance* plays an essential role in IR [83]. Relevance is a relationship between a piece of desired information and a user's information need in a given situation. This concept has many dimensions, such as topicality (i.e., whether the document talks about the topic of the query), appropriateness for the user's background, recency, or authority. More than 80

relevance criteria have been discovered in different studies [7][54][86][87][107]. Saracevic [83] gave a review and a framework for an analysis of relevance with emphasis on topic relevance, which is later extended in [84].

Despite the large number of studies devoted to it, the notion of relevance is still poorly defined. Froehlich summarized the current state of our knowledge on relevance as follows [30] :

- The inability to define relevance;
- The inadequacy of topicality as the basis of relevance judgments;
- The diversity of non-topical, user-centered criteria that affect relevance judgments;
- The dynamic and fluid character of information seeking behavior;
- The need for appropriate methodologies;
- The need for more complex, robust models for system design and evaluation.

In most studies, we usually distinguish two types of relevant judgments: (1) *topical relevance*; (2) *user-centered relevance* [28].

- **Topical relevance:**

  Topical relevance is also sometimes called "*aboutness*". It is concerned with the fact that the document is "*about*" the topic of the query. Park [65] defined "*topical relevance*" as follows:

  "*Topical relevance is context-free and is based on fixed assumptions about the relationship between a topic of a document and a search question, ignoring an individual's particular context and state of needs.*"

  The independence from the "*particular context and state of needs*" means that documents are judged in isolation. A document about the query's topic, but described at a level unsuitable to the user (too superficial or too

technical), is still judged topically relevant. The document is judged topically relevant even if the user has already read it before. Topical relevance is used in most IR systems and models.

- **User-centered relevance:**

Users do not judge document relevance solely on topicality: many topically relevant documents can be judged irrelevant by the user for his specific information need. To reflect the relevance intended by the user, the notion of "*user-based relevance*" is defined. Different from topical relevance, relevance from a user's perspective needs to be recognized "*from an individual's subjective contexts and personal needs as an underlying situational force in the information seeking and retrieval process*" [65].

User-centered relevance contains several types of relevancy, as defined by Zhang et al. [118]:

- Direct evidence explicitly gives an answer to a user's question;
- Indirect evidence lets us infer an answer to the question;
- Contextual evidence provides peripheral or background information surrounding an answer;
- Comparative evidence provides a basis for interpretation or inspires some answer through perceived similarity to the question.

With respect to these types of relevancy, traditional IR approaches focus on the first type – direct relevancy. Query expansion that we described earlier is a means to deduce indirect relevancy. The other two types of relevancy have not been much investigated and implemented.

In this study, we focus on the second and the third types of relevancy. We will try to improve the second type of relevancy by creating more precise term relations. We will also integrate an element relating to the third type of

relevancy – contextual and background relevancy, by retrieving documents containing background information related to the query. We will implement this by means of a background domain model.

These types of relevancy also strongly correspond to different contextual factors that influence relevance judgments. We will briefly review these factors in the next section.

3.2.2 Contextual Factors in Relevance Judgments

Relevance judgment is affected by many factors, which are classified into different categories. For example, Barry [7] identified seven categories of criteria (factors) for user-based relevance, respectively pertaining to:

- The information content in documents;
- The user's previous experience and background;
- The user's beliefs and preferences;
- Other information and sources within the information environment;
- The sources of documents;
- The document as a physical entity;
- The user's situation.

Other researchers have established different categories. Despite differences in the categories of factors, some common factors emerge. Indeed, Barry and Schamber [8] confronted their separate classification, and found that the factors that they identified are very similar:

- It turns out in all the studies about relevance and relevance judgments that topicality remains the most important factor: this is the criterion 1 listed by Barry;
- The user's domain of interest: this is the criterion 2 listed by Barry;

The user's domain of interest defines a background of the user and the information need. As we described earlier, user queries are usually short. Users only include a few words into the query while leaving out other words (even related ones). A short query is often ambiguous. By integrating the background information, it can become less ambiguous.

For example, a user working in *Computer Science* area trying to find information about "*communication in Java*" would likely put "*communication*" and "*Java*" in his query. These two highly ambiguous words would lead to many irrelevant documents. However, knowing that the user is interested in the domain of *Computer Science*, the query becomes unambiguous (or at least much less ambiguous).

- The user's knowledge about the subject: this corresponds to Barry's criterion 3 – user's beliefs and preferences.

The user may have some knowledge about the subject. This knowledge influences greatly his relevance judgment of a document. For the same query or information need, two users having different knowledge about the subject may judge its relevance in different ways. For example, if a user knows that "*multithreading*" is related to "*parallel computing*", then for a query on "*parallel computing*", a document about "*multithreading*" could be judged relevant. On the other hand, a user who does not believe that the two terms are related (e.g., a computer novice) would not judge the document to be relevant. Here we can see how the relations between different terms can influence

relevance judgments. The importance of user's knowledge is underlined in several studies [1][57].

As Barry and others showed, there are many other important factors. However, all these factors are not readily usable in IR models because they are very often loosely defined and their impact on relevance judgments is still unclear. Therefore, most previous studies that include contextual factors into operational IR models have focuses on user's profile and background factors. We will review some typical approaches taking into account these factors.

3.3 Previous Attempts to Improve Query Description

In this section, we will review some previous approaches that deal with the two problems we mentioned: short query and contextual factors. We will go into some details in this review since they are strongly related to our work.

3.3.1 Query Disambiguation – Attempt to Recognize Meaning

Words are often poor representation of meaning. In an ideal world, one would desire an IR system that works directly on the semantic level, i.e., to retrieve documents about some meanings. However, this is unfeasible in practice: we only have words or word sequences (sentences) in IR systems.

In order to retrieve documents that are more related in meaning to a query, a possible approach is to try to determine the meaning that a word denotes, then document and query can be compared on the basis of the recognized word senses. The determination of word sense requires disambiguation, i.e., to determine the correct word sense among all the senses that a word can represent. For example, given the word "*Java*" used in the context of *Computer Science*, the goal is to select the sense "*programming language*" and exclude the senses of "*island*" and "*coffee*".

Word sense disambiguation is important for ambiguous words. This is also a key problem in general natural language processing (NLP). Many approaches have been developed for it.

For example, Yarowsky [113] used a semi-supervised method to learn useful context words for disambiguation. The idea is as follows: a small set of seed context words are determined manually for a given ambiguous word, then new context words are learnt automatically. For example, for the word "*plant*", one can select two useful context word "*manufacturing*" and "*grow*" to distinguish the two possible meanings: "*industrial installation*" and "*natural plant*". If the word "*plant*" is used together with one of these words, then its meaning can be determined and tagged. From the tagged texts, we can observe new context words that are strongly associated with one sense or another. These new words are learnt as new seeds, and the process continues. Finally, each word sense will be associated with a set of discriminative context words, which can be used to recognize the meaning of the word in a sentence or text.

Yarwosky showed that this approach can successfully learn to recognize word senses. However, this approach can only be applied to a small set of ambiguous words, because a manual selection of seed words is required at the beginning. This does not seem to be realistic in practice for IR.

Another family of approaches tries to exploit the existing semantic resources, such as a dictionary or thesauri. For example, Lesk [55] exploited the definition of word senses in a dictionary: by comparing the context words of an ambiguous word used in a text with the definitions of senses of that word, one can estimate a similarity of the word usage with each of the definitions. The sense corresponding to the most similar definition can be selected.

Voorhees [105] tried to determine the correct word sense (or synset in *Wordnet* terms) for a query according to the distance of possible synsets of different words in *Wordnet* [61]: the synset that is the closest to the synset of other query term is selected.

Unfortunately, this word sense selection process does not seem to produce satisfactory results: many wrong synsets are selected. Indeed, Voorhees calculates the "semantic" distance between synsets according to the topology of synset hierarchy in *Wordnet*, but the latter does not necessarily encode a semantic similarity. For example, some concepts are developed in more details, thus have a deeper hierarchy, than some others. Then the former tend to have a larger distance to a given synset than the latter. However, this does not mean that the former is *semantically* less similar to the given synset.

What is more problematic is the following result from Voorhees' experiments [106]: she selected the correct synset for each query word manually, and tried to expand the query with the synset ID as well as the related words. The manual selection of synsets aims to simulate a perfect word sense disambiguation. However, in spite of this, the retrieval effectiveness is still not improved from the basic method using vector space model. We notice that in these experiments, only query words have been disambiguated, while document words remain as they are. The experiments seem to indicate that query word disambiguation alone is not sufficient to bring a gain in retrieval effectiveness.

Sanderson [81][82] attempted to simulate different levels of disambiguation accuracy, and to determine the minimal accuracy for the disambiguation required by IR. He showed that in order to be useful in IR, the disambiguation tool should have an accuracy of at least 90%. Unfortunately, the current state of the art in word sense disambiguation is well below this requirement. It seems that word sense disambiguation is not yet ready to be exploited profitably in IR.

3.3.2 Query Expansion – Attempt to Complete a Query

Query expansion (QE) aims to improve query expression by adding related terms to the query. The addition of new terms extends the original query so that it has a wider coverage than the original query. This method can provide a solution to the short query problem in IR. By doing this, a query can become more complete in the sense that

more relevant terms are included. As a consequence, more relevant documents may be retrieved and the *recall* can be increased.

Query expansion also provides a means to relax the 1:1 correspondence between word and meaning assumed in classical IR models. Indeed, once a query is expanded, a document does no longer have to contain the original query terms to be retrieved. A query is now allowed to match documents which contain related terms.

It is generally believed that the overall goal of query expansion is to improve *recall*. We will argue later that it also allows us to improve *precision* indirectly.

Two key questions in query expansion are: (1) *which terms should be added?* (2) *how are new terms weighted and integrated into the query?*

3.3.2.1 How to Select the Expansion Terms

Intuitively, the added terms should have a strong semantic relationship with the existing terms in the original query. That is, a new term should describe a concept which is strongly related to the concepts described in the original query. There are two ways to determine such terms:

- The user may select the terms to be added interactively with the system;
- The system may do it automatically using different resources.

In the first manner, the system usually exploits a thesaurus that stores a set of possible relationships between terms. For example, for each term, a thesaurus may store a set of synonyms, more specific and more general terms. If a term is included in a user query, the system can suggest the related terms to the user. However, this expansion method relies on intensive interactions with the user. In practice, it is often a heavy burden to the user. So it is only used in domains where specialized thesauri are available, and the users are willing to make efforts to cooperate with the system to find the relevant documents. In general IR or Web search, users are not willing to collaborate in this way. Automatic query expansion is often preferred.

Automatic query expansion tries to identify the most closely related terms from the resources and then adds them into the query. A key factor that determines the effect of query expansion is the selection of appropriate expansion terms. Several resources have been commonly used for query expansion: *thesauri*, *co-occurrence statistics* and *pseudo relevance feedback*. We will provide a brief description of them:

- **Thesauri:**

  Thesauri contain manually validated relations between terms, which can be used to suggest related terms. The best known thesaurus is *Wordnet* [61].

  Intuitively, thesauri are good resources for query expansion. However, as suggested by Voorhees' experiments [105][106], their effects in practice can be surprisingly low. Some of the reasons are as follows:

  Although *Wordnet* contains many relations validated by human experts, the coverage is far from complete for the purposes of IR: not only linguistically motivated relations, but also association relations, are useful in IR. For example, there may not be a formal linguistic or semantic relation between "*peace talk*" and "*Middle East*", but this association relation is very useful for IR.

  Another problem is the lack of information about the appropriate context to apply relations. For example, *Wordnet* contains two synsets for "*computer*", one for the sense of "*machine*" and another for "*human expert*". It is difficult to automatically determine the correct synset to expand the word "*computer*" in a query in *Computer Science* area.

- **Statistical co-occurrences:**

  Another often used resource is associative relations extracted from co-occurrences: two terms that co-occur frequently in the same context are thought to be associated to each other [34][47][95][102]. Context of co-occurrences may be document, paragraph or sentence. It can also be within a sub-sentence

structure, such as noun phrases or subject-verb structure. More often we can use windows of fixed size as co-occurrence contexts.

Typically, one extracts co-occurrence relations between two single words, for example, "*york* → *new*". One should notice that such co-occurrence relations are very noisy: frequently co-occurring terms are not necessarily related. For example, in newspaper articles, common words such as "*year*", "*time*" and "*report*" often have strong co-occurrence relations with many words. However, they are not truly related to these latter. On the other hand, they can also miss true relations. For example, one could not extract co-occurrence relation between "*tyre*" and "*tire*", because true synonyms are rarely used together.

Co-occurrence relations have been used in many studies in IR [73][95]. In [37], it is shown that when queries are expanded using term co-occurrence information, worse system effectiveness can be obtained. Smeaton and van Rijsbergen [95] also did not observe the noticeable improvement using co-occurrence relations for query expansion, and this was believed to be due to the limited amount of data.

Overall, the effect of co-occurrence relations seems to be relatively limited. This may be due to several reasons:

First, real semantic relation, in particular, synonymy, can be hardly identified through co-occurrences. In fact, words that are very similar in meaning tend to repulse from each other in continuous portions of text [93].

Second, Peat and Willet [66] observed that statistical relations usually link terms of similar frequency of occurrences in document collection. They have similar degree of generality or specificity to the application area. Adding such a related term into a query does not make the original query more specific or more general. Therefore, it does not bring much new information to the query.

In addition, as users tend to use frequent words in their queries, the added words also tend to be frequent words. They have poor discriminative value to distinguish a document from the others.

Third, co-occurrence relations are highly ambiguous. Most co-occurrence relations are established between single words such as "*Java* → *programming*". This relation is strongly context-dependent: it only applies in *computer*-related context. However, the appropriate application context is not at all specified in this relation. Therefore, when the word "*Java*" is encountered in a query, the relation is always applied. Unavoidably, much noise will be introduced in queries concerning the other meanings of "*Java*".

- **Relevance feedback and Pseudo-relevance feedback:**

An alternative to query expansion is to use the user's relevance feedback. After the system has retrieved a set of documents, the user is asked to judge some of them, indicating whether they are relevant or not. Based on these judged documents, the system can have a more precise idea on what the user's intention is. It is assumed here that the user is more interested in the documents similar to the ones that are judged relevant, and is not interested in those similar to the judged irrelevant ones. By incorporating the words found in the relevant documents (or by increasing their weights), and eliminating those in the irrelevant documents (or by decreasing their weights), it is expected that the new query is closer to the user's intention, thus describes better his information need.

When no relevance feedback is provided by the user, one can also exploit a set of top ranked documents retrieved with the original query: these documents are assumed to be relevant ones, and the same query expansion process as before is used. This method is called *pseudo relevance feedback* or *blind relevance feedback*. This approach is widely used in IR.

It is interesting to compare query expansion and pseudo relevance feedback to see their similarity and difference. These two methods both aim to expand the original query, however from different perspectives. In the first case, we indeed exploit the general relationships between terms, either the relationships are created manually in a thesaurus or extracted from a document collection. In the second case, the extraction of terms is circumvented within a subset of documents that are the most related to the query. The term extraction process can be considered to be similar to a co-occurrence analysis within these documents, since the terms that occur strongly within these documents also have high co-occurrence counts with the query terms in these documents. However, an important difference between them is that co-occurrence only reflects the relation between a term and another term, while a term extracted from the feedback documents has implicitly a relation with the entire query. The contexts from which expansion terms are extracted are also different: in co-occurrences, terms are extracted within a small window, while in pseudo relevance feedback they are extracted from the whole document (or passage).

In [111], Xu and Croft called the two cases *global* and *local* context analysis respectively. They have shown that when each analysis is used alone, local context analysis is more effective. However, the best result is obtained when both types of analysis are combined.

Both co-occurrence relations and pseudo relevance feedback exploit implicitly collection characteristics. The co-occurrence relations extracted from a document collection reflect the way that terms are used in the collection, together with some other terms. The implicit exploitation of collection characteristics is even stronger in pseudo relevance feedback. The subset of documents used to determine expansion terms are those that are related to the query in the given collection. These documents strongly reflect the way that the query topic is developed and described within the collection. The topic can be described together with some other topics. These latter can be extracted as expansion terms from the feedback documents.

For example, in a collection of newspaper articles covering the period of the event of "*9-11*", the term "*New York*" is strongly related to "*terrorism*", "*air hijacking*". Therefore, the feedback documents for the query "*New York*" would likely contain these terms. On the other hand, from another document collection covering a different period of time, the feedback documents would more likely describe "*stock exchange*". Therefore, feedback documents implicitly reflect some collection characteristics, and we can consider them as a portrait of the collection concerning the query topic.

3.3.2.2 How to Integrate New Terms into a Query

A common way to add expansion terms into the query is by using the *Rocchio* formula [79], or a similar one. The original Rocchio formula was developed for manual relevance feedback, with a set of judged relevant documents and a set of irrelevant documents. The new query is formed as follows:

$$new\_query = \alpha \times old\_query + \beta \times R - \gamma \times NR$$

where R and NR are the centroid vectors of the set of relevant and irrelevant documents judged by the user, α, β and γ are the factors that determine the importance of the original query, the relevant and irrelevant documents in the new query. As we can see in the formula, the new query tends to become closer to the relevant documents and far away from the irrelevant documents.

When pseudo relevance feedback is used, we assume the top ranked documents to be the relevant ones. Then some terms are extracted from these feedback documents, and are added into the original query, using a similar formula to Rocchio's:

$$new\_query = \alpha \times old\_query + \beta \times PR$$

where PR is the centroid of the pseudo relevant feedback documents.

The parameters α, β and γ are important in these formulas. They determine the relative weights of the expansion terms with respect to the original query terms. As

exemplified by Voorhees' experiments [105][106], adding expansion terms in a simplistic way may not increase retrieval effectiveness. In some other studies using *Wordnet* [15], it has been shown that with a more appropriate weighting methods, the relations stored in *Wordnet* can improve the retrieval effectiveness.

In an attempt to select better expansion terms and to make a better weighting, Qiu and Frei [73] proposed the following approach to select expansion terms: terms are selected according to their relations to all the query terms, which is calculated as the sum of their relations to each of the query terms. That is, the expansion terms t is weighted according to the following similarity function:

$$sim(t, Q) = \sum_{t_i \in Q} sim(t, t_i)$$

where $sim(t,t_i)$ is a similarity measure between two terms, which is obtained from a statistical thesaurus constructed based on term co-occurrences in their case.

Therefore, a term that is related to several query terms will be favored. In fact, the relationship between two single words does not specify its application context. It is uncertain, in general, whether it applies to the given query. Through the above similarity with the whole query, Qiu and Frei intended to favor expansion terms that are related to several query terms, thus believed to be more appropriate to this query.

In fact, the effect that Qiu and Frei desired is similar to pseudo relevance feedback: they want to determine expansion terms more related to the whole query. However, this effect is limited due to the nature of the relations used, in which an expansion term is suggested only according to one term. Many inappropriate terms will remain after selection and weighting. For example, if "*Java* → *programming*" is a very strong relation, then for a query "*Java hotel*", according to Qiu and Frei's calculation, *sim*("*programming*", *Q*) will still be strong even if "*programming*" is not related to "*hotel*". We can see that this posteriori correction or filtering has very limited effect.

In this thesis, we argue that the key problem lies in the insufficiency of relations based on single term considerations to specify the appropriate application context. The solution thus lies in the construction of richer relations that specify more precise application context. We arrive at a more precise application context by including more terms in the condition part of the relation.

3.3.3 Previous Attempts to Integrate Contextual Factors

Previous studies concerning the utilization of contextual factors in IR mainly focused on personalization. We can find two main approaches to integrate contextual factors: (1) *using user context or profile* [27][48][101]; (2) *using topic domains* [58][108].

3.3.3.1 User Profile

In the first group of methods, a user profile is constructed to contain a set of terms (or a vector, a statistical language model, etc.) corresponding to the user's long term interests. Then a query is enhanced (or the results re-ranked) according to the user profile.

The basic assumption is that a user is usually interested in certain topics and these latter are limited. In a subsequent search, the user usually prefers to retrieve documents that are somehow related to those that he has read previously. For example, a user interested in *Sports* would tend to select sport articles in his future searches. By creating a user profile, which keeps trace of what the user has read or is interested in, a new search can be helped.

There are several ways to define a user profile:

- The user can select a set of words related to the topics of interest;

- There may be documents classified into different directories according to topics (e.g., *ODP*[9] or *Yahoo! Directory*). The user can select the categories of his interests. Then the system collects the documents from these directories and constructs a user profile from them. This approach is used in [18];
- The system can observe the documents that the user has read or browsed in the past or stored on his personal computer, and use them to construct a user profile. This approach is used in [27][48][86][101], as well as in *Google Personalized Search* [36].

In IR, personalization is often concerned with contents, i.e., one tries to capture the topics of interest of the user. In other applications such as recommender systems, more types of user characteristics may be included, for example, the type of movie the user prefers. However, these additional characteristics are highly application-dependent, and they have not been widely explored in general text retrieval.

Once a user profile is constructed, they are usually used to re-rank the retrieved documents: the score of the top retrieved documents are modified according to another score based on the correspondence of the documents to the user profile.

We observe in these studies that one single user profile is usually created for a user. In the document re-ranking step, documents are re-ranked according to the whole user profile. For a user who has stable topics of interest and the new query is in the same areas, such a personalization is useful. Problems arise if the user is interested in a large variety of topics. In this case, a document in a topic domain (say, *Travel*) may be favored because of a different topic domain (say, *Computer science*) in the profile. This influence may be incorrect.

---

[9] http://dmoz.org

Another problem arises when the user submits a query which falls outside of the domains of the user profile. In this case, the user profile is still used to favor some documents, which is also inappropriate.

These problems show that with a unique user profile, it is difficult to determine whether the user profile should be used, and which part of the user profile should be used. Therefore, we propose to model several topic domains instead of one single user profile.

3.3.3.2 Topic Domains

Only few previous studies have tried to exploit topic domains for queries [58][108]. The purpose of this enhancement is to try to incorporate the most frequent terms in a topic domain into the query. This is often necessary because the query only contains a few words, and all the background words in the corresponding domain are not included. For example, the word "*computer*" usually does not appear in a *computer*-related query such as "*Java program*". The addition of the word "*computer*" into the query will provide some background information to the search, and this may make the query more focused.

In [58], Liu et al. defined a set of domains using *ODP* directories. The domains related to a query are identified according to the query. The corresponding domain model (a vector in this case) is used to re-rank the retrieval results. Interesting experimental results are reported. However, only a small scale experiment has been carried out.

A similar approach is used in [12][23][108], where domain models are created using *ODP* categories and user queries are manually mapped to them. However, the experiments showed variable results. In some of the cases, improvements are observed whereas in other cases, no improvement or even degradation is observed. It remains unclear whether domain models can be effectively used in IR.

Our approach will follow the same principle. However, we will incorporate a query classification process to determine the query domain automatically. The addition of this process will make the approach more feasible in practice, since no manual identification of domain will be required.

3.3.4 Attempts in Language Modeling Approach

As our study will be carried out within the language modeling framework, we will pay special attention to the attempts to enhance queries in language models. Our approach will be compared with them.

3.3.4.1 Exploiting Term Relations

We have mentioned that all the traditional IR models consider terms as being independent. In the basic LM approaches, the same assumption is made: we only use independent terms as unigrams.

As terms in reality are not independent, several attempts have been made to take into account term relationships or dependencies to some extent.

Song and Croft [97] extended the unigram model to bigram model. In the latter, a term is considered to be dependent on its precedent term, i.e., give a query $Q = t_1 t_2 \ldots t_n$, its likelihood in a document model is determined as follows:

$$P(Q \mid \theta_D) = P(t_1 \mid \theta_D) \prod_{i=2}^{n} P(t_i \mid t_{i-1}, \theta_D)$$

To deal with the data sparseness problem, the bigram model should be smoothed with the unigram model and the collection model.

In the bigram model, we assume that any word depends on, and only on, its precedent word. This may give rise to several problems:

- Such dependency is very noisy, i.e., many considered dependencies are not true. One hopes that with a large set of data, true dependencies will emerge. However, this is not always the case;
- The dependency is limited to local dependencies between adjacent words. More distant dependencies in a sentence cannot be captured;
- The complexity of the model is largely increased compared to a unigram model;
- More importantly, while texts contain full ordered sentences, users often formulate their queries as a set of words without caring much about ordering.

The experiments of Song and Croft showed that bigram models do not bring significant gain over unigram models: the retrieval effectiveness is only marginally better, but the complexity of the model is much higher. So the state-of-the-art of LM in IR still uses unigram model.

An important characteristic of the traditional *n*-gram models is the importance of word order: "*Java program*" is different from "*program Java*". However, queries in IR are often words in quite free order. For a user, the above two word sequences could mean the same thing. Therefore, Srikanth and Srihari [99] proposed to use *biterms*, in which the order of the words within a bigram is ignored. This produced better effectiveness than bigram model. However, the problem of noisy biterm relations still remains, and biterms are also restricted to adjacent words.

In order to solve the problem of short distance dependency, Gao et al. [32] proposed the following approach to link terms within a sentence: given a link model which specifies the probability of a link, they try to determine the best links that cover the sentence. For example, the sentence "*how has affirmative action affected the construction industry*" can be parsed as follows (where an arc represents a retained link, and stopwords are removed):

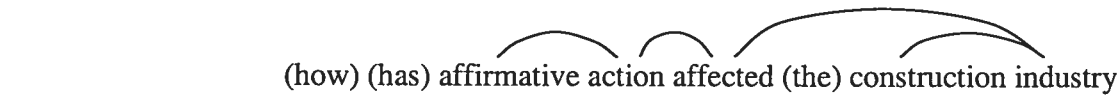

Figure 4. Links in a sentence

The best links are determined using an algorithm similar to maximum spanning tree. In order to create a link model (which is used to parse the above sentence), they used an iterative process looping in the following two steps on the document collection: (1) previous link model is used to parse every sentence in the collection to select links; (2) a new link model is constructed according to the result of this parsing. The initial link model is created using co-occurrence statistics.

To retrieve a document, the document should not only satisfy the term requirements (modeled by both unigram and biterm models), but also the link requirements, i.e., the document should also contain the same links as those recognized in the query. The term dependencies recognized in a query impose a stricter condition on documents to be retrieved. The effect is similar to the utilization of compound terms, but within a purely statistical setting. Gao et al. showed that this method performs better than the classical LM approach.

Another family of models tries to exploit relationships between terms in a similar way to query expansion (or document expansion). By such an expansion, a document is allowed to match a query even if they do not share the same terms: they can just contain related terms.

Berger and Lafferty [10] proposed a *translation model* to extend the unigram language model as follows:

$$P(w \mid \theta_D') = \sum_{w_i \in V} t(w \mid w_i) P(w_i \mid \theta_D)$$

where $t(w|w_i)$ denotes a relationship between two terms w_i and w, which is a translation relation trained on a synthetic parallel corpus, by assuming that a sentence is parallel to the paragraph containing it.

This approach is further extended by Cao et al. in [15], in which several types of term relations are integrated: relations from *Wordnet* or from co-occurrence statistics. It is shown that by integrating multiple types of relation, the retrieval effectiveness can be much improved.

Notice that the above approach tries to expand the document model from $P(w_i|\theta_D)$ to $P(w_i|\theta_D^{'})$. Another possible way to expand document model is by document clustering. In [59], Liu and Croft clustered similar documents and a document model is expanded by (interpolated with) the cluster model.

In our approach, we will work on query expansion. A similar approach exploiting term relations will be used to expand the query model. However, an important difference lies in the type of relation that we use: our relations contain more context information than those used previously.

3.3.4.2 Feedback Model

Several studies proposed to use feedback documents to create a new query model to enhance the original one.

Zhai and Lafferty [115] constructed a distinct language model from the feedback documents θ_F. This model is then combined with the original query model θ_Q to construct a new query model $\theta_Q^{'}$:

$$\theta_Q^{'} = (1-\alpha)\theta_Q + \alpha\theta_F$$

where α controls the influence of the feedback model. Zhai and Lafferty called this a *mixture model.*

A simple way to estimate a feedback model θ_F is to use MLE. However, the feedback documents contain both terms related to the query and common terms in the language. What we desire is a model corresponding to the first part only. The second part should be removed.

Zhai and Lafferty proposed an EM process to extract the feedback model as follows:

A feedback document is assumed to be generated from two sources: the feedback model θ_F (to be extracted) and the general language model (approximated by the collection model θ_C). Therefore, the log-likelihood of all feedback documents is:

$$\log P(F \mid \theta) = \sum_{D \in C} \sum_{t \in D} tf(t, D) \log[\lambda_F P(t \mid \theta_F) + (1 - \lambda_F) P(t \mid \theta_C)]$$

where D is a feedback document in the collection, t is a term belongs to D, and λ_F in this expression is set at a fixed value (i.e., 0.5). Then θ_F can be extracted by using the EM algorithm [25], so that the above log-likelihood can be maximized. The EM updates for $P_\lambda(t \mid \theta_F)$ (where $\lambda = 1 - \lambda_F$) are:

$$w^{(n)}(t) = \frac{\lambda_F P_\lambda^{(n)}(t \mid \theta_F)}{\lambda_F P_\lambda^{(n)}(t \mid \theta_F) + (1 - \lambda_F) P(t \mid \theta_C)} \qquad \text{(E-step)}$$

$$P_\lambda^{(n+1)}(t \mid \theta_F) = \frac{\sum_{D \in F} tf(t, D) w^{(n)}(t)}{\sum_{D \in F} \sum_{t_i \in D} tf(t_i, D) w^{(n)}(t_i)} \qquad \text{(M-step)}$$

By using EM, they are trying to "purify" the document by eliminating some background noise. Thus, the estimated feedback model will generally be concentrated on words that are common in the feedback document set, but not very common in the collection language model $P(\cdot \mid \theta_C)$.

Lavrenko and Croft [52] used feedback documents in a different way. They tried to capture the notion of relevance through feedback documents. They considered the feedback documents as samples of relevant documents. From them, a relevance model is constructed. Despite the difference in principle, the effect of relevance model is similar to the mixture model created by Zhai and Lafferty [115]. Therefore, we will not describe relevance model in details here.

3.4 Summary

The traditional approach to IR usually considers a short query as the only information about the information need. However, a short query cannot describe the information need precisely. The retrieval effectiveness with such a query can be compromised.

In fact, besides the query, we also have much other information: the knowledge or term relations, the retrieval contexts (user's background or topic domains of the query), and some characteristics of the collection. All these types of information can help enhance the query.

Many approaches have been proposed to exploit additional information:

- Knowledge or terms relations can be applied on the query to expand it;
- The user's background can be exploited to direct the search toward documents related to the profile or topic domains;
- Feedback documents can be used to find topics that are developed together with the query topic in the collection.

All the previous approaches (except word sense disambiguation) have produced some degree of improvements in retrieval effectiveness. In particular, the utilization of feedback documents has proven to be highly effective.

We also notice that previous attempts have usually been limited to the consideration of only one of these aspects. No experiment has been performed to integrate all of them. In addition, only simplistic term relations have been used.

In the next chapter, we will propose a model that integrates multiple contextual factors.

Chapter 4
A General Language Model to Integrate Contextual Factors

In the previous chapters, we mentioned various types of contextual factors. However, few contextual factors have been integrated in operational IR systems. In this chapter, we will distinguish two main types of contextual factors that we integrate, and propose the general language modeling framework that we use to integrate them.

4.1 Insufficiency in Previous Approaches

Let us first summarize some of the remaining problems in previous studies, which we will deal with in this thesis.

4.1.1 Ambiguity in Query Expansion

Previous experiments have found mitigated results using query expansion: query expansion has lead to some improvements in retrieval effectiveness in some experiments [73][111][115], but degradation has been observed in some others

[95][105][106]. Beside the particular problems, such as that related to term weighting, a key issue is term and relation ambiguity. Relation ambiguity means we do not know in what context to apply a relation.

When an ambiguous term is included in a query, it is difficult to determine in which sense it should be expanded, and what related terms should be added into the query. For example, given the term "*Java*" in a query, it is difficult to determine whether "*programming*", "*coffee*" or "*island*" should be added into the query. This problem is particularly difficult to solve in most expansion approaches, in which the expansion terms are determined from individual terms in isolation, i.e., from "*Java*" alone. A relation such as "*Java* $\rightarrow$ *programming*" can be applied only in some contexts (e.g., *Computer Science*). The application in a wrong context will produce inappropriate expansion terms (i.e., noise).

Despite the attempts to select the best expansion terms, e.g., by summing up the relations with all the query terms [73], no radical solution to this problem has been proposed. The final weighting of the expansion term is still based on the original term relations, which are created between single words, and no genuine context information is considered in such a solution. The fundamental problem is not solved, but simply alleviated.

4.1.2 Lack of Context in Relevance Judgment

Another underexploited aspect is the query contexts. Query expansion only deals with the query, but not the contextual factors around it.

Some approaches have attempted to capture the user's intent behind a query by personalization. However, a unique user profile is not sufficient to deal with different information needs of the users. For the latter, more refined modeling of contexts is required. Modeling topic domains is a better approach than the unique user profile.

In fact, the two above problems all concern query contexts: when a query is expanded, we would like to make use of term relations that fit the context of the query; when a query is evaluated, the context factors of the query should be considered as much as possible. We will identify two types of context corresponding to these situations: intra-query context and extra-query context.

4.2 An Example

To provide an intuition of what we intend to do in this thesis, let us describe an example to motivate the approach.

Suppose a query about "*air hijacking*" within the domain *Terrorism*. The retrieval is performed on a collection containing documents in the period of Sept. 11, 2001.

With the original query "*air hijacking*", one may retrieve some of the relevant documents, which contain the two words "*air*" and "*hijacking*". This query can be enhanced and expanded in different ways:

- **Knowledge (*K*)**:

  By knowledge, we mean a set of relations between terms. We have some general knowledge about "*air hijacking*": we know that it implies "*airplane*", "*passenger*", "*flight*", and so on. By applying the relations between terms to the query, the latter terms can be inferred, which correspond to concepts that are subsumed by the query.

  In this thesis, we propose to use both terms in the query together to deduce related terms (we call it *context-dependent* term relations). We will have relations such as:

air hijacking → *airplane*, *air hijacking* → *passenger*, ...

In contrast, in the traditional query expansion method, each of the query terms is used to determine related terms separately. Likely, we will have the following sets of related terms:

air: *oxygen*, *gas*, *atmosphere*,... *sky*,... *flight*,... *broadcast*,...

hijacking: *passenger*, *skyjacking*, *airplane*,... *carjacking*, *car*,... *piracy*,...

When all these terms are added into the query, much noise is introduced.

In our study, we will advocate the first strategy using several query terms together to determine a related term.

In general, by applying term relations or knowledge to determine implied terms, we are indeed trying to favor the *indirect relevancy* listed by Zhang et al. [118]: the documents about topics implied by the query are also retrieved.

- **Domain (*Dom*)**:

From the domain *Terrorism*, we know a set of frequently used terms, such as "*attack*", "*terrorist*", "*kidnapping*", "*bomb*", "*threat*", *"kill"*, etc. These terms are considered to be likely implied by any query in the domain of *Terrorism*. As *Terrorism* is the background domain of the query, all these terms can be assumed to be related, and added into the query. The addition of these background terms may favor what is called *background evidence or relevancy* by Zhang et al. [118].

- **Relevance Feedback (*FB*)**:

In a particular document collection, the topic of the query (the event, the person, etc. that the user is looking for) is surrounded by some other topics, i.e., the query's topic is often described together with some other topics.

For example in this particular document collection, the query topic "*air hijacking*" is often developed together with "*New York*", "*world trade center*", "*September 11*", "*terrorist*", "*Al Qaeda*", "*Ben Laden*", and so on. These latter terms are considered to be strongly related to the query in this collection, and it is useful to retrieve documents about these latter topics, such as "*Al Qaeda*", as well. This may favor the *contextual relevancy* of Zhang et al. [118].

The above collection characteristics about the query topic can be reflected by the feedback documents, i.e., the top documents retrieved with the original query. By extracting additional terms from the feedback documents, we can capture the topics related to the query in this collection. Therefore, we will use feedback documents as a means to capture some query-dependent collection characteristics.

The above three expansion processes can add expansion terms from different points of view. They are however not independent. Indeed, the suggested terms on different aspects may overlap. For example, if term relations are extracted from the document collection, then the terms suggested by term relations will have much in common with those suggested by the feedback documents. However, the former is extracted from a wider context (the whole collection), while the latter is restricted to a subset of documents. From this point of view, the former is more general and have a wider coverage, while the latter is more specific to the query. They are complementary.

In the context-dependent relations, we also exploit some context information which exists within the query, i.e., the words that co-occur in the query. So we also consider these relations as exploiting the context information. Therefore, we distinguish two types of contextual factors: the context that is implied in the user query, as used in the context-dependent relations, that we call *context within query* or *intra-query context*; and the context outside the query, such as the background domain and the feedback documents, that we call *context around query* or *extra-query context.*

4.3 Knowledge and Intra-Query Context

Knowledge (term relations) used in almost all the previous IR researches is context-independent, i.e., created between a pair of single terms. A blind application of such relations in query expansion can bring irrelevant terms (i.e., noise).

To solve this problem, Lau at al. [53] defined strong logical relations to encode knowledge in different contexts. For example, in *Computer Science*, we can have "*Java* $\rightarrow$ *programming*" and in *Volcanology*, we have "*Java* $\rightarrow$ *Merapi*" and "*Java* $\rightarrow$ *volcano*". In order to distinguish them, Lau et al. proposed to add stronger conditions to the relations, such as "*Java* $\wedge$ *computer* $\rightarrow$ *programming*" and "*Java* $\wedge \neg$*computer* $\rightarrow$ *Merapi*". This addition will prevent the relation to be applied in a wrong context. However, it is difficult to determine the strong logic condition (especially negation) in such relation, unless they are defined manually.

An alternative is to use domain-specific knowledge to expand queries in the corresponding domain. For example, we can include "*Java* $\rightarrow$ *programming*" in the *Computer Science* domain and "*Java* $\rightarrow$ *Merapi*" in the *Volcanology* domain. For a query, the appropriate relations are applied. This approach has been used in some specialized area, such as medicine, in which term relations in the *MeSH* (medical subject headings) thesaurus are commonly used [91]. However, in many other domains or for general domains, no such relations are available for use.

In an attempt to define user-specific domain knowledge, Croft [21] proposed an approach to define rules in interaction with the user. However, this would require a large amount of manual effort from the user, even though this process can be helped by some tool.

In many situations, one can only have general knowledge, such as those stored in *Wordnet* or extracted from documents. Then a crucial problem is to determine if a

given relation applies to a query or not. We believe that the solution lies in the relations themselves. The crucial problem in the relations currently being used is the lack of context information in them. Therefore, we propose to construct richer term relations with stricter conditions, i.e., containing several words. For example, instead of defining "*Java* → *compute*", we will create "{*Java, program*} → *computer*" and "{*algorithm, program*} → *computer*". The applicability of the relations will be naturally restricted to correct contexts. As a result, "*computer*" will be used to expand queries "*Java program*" or "*algorithm program*", but not "*TV program*". We call these relations *context-dependent term relations*, owing to the addition of context terms in them. This idea used to IR was first proposed in our previous study [5]. A similar idea was also used in [114] for a specific task.

In the above relations, we use a set of terms as condition. A more general form of relation is between two sets of terms. This type of relation has been much investigated in the area of association rule mining [44]. One may think that the approaches developed in association rule mining can be directly used in our case. However, association rules are usually extracted from well structured data from databases. To extract association rules, a complete lattice (e.g., Galois lattice) representing the whole index relation is usually created, in which each node represents the indexing relation between a set of indexes and a set of items. Then relations between sets of indexes are extracted. This mining process requires a large space to store the lattice and the rules to be extracted can be between arbitrary sets of terms, while our relations take a much simpler form (with only one term as condition). Therefore, the methods of association rules are too time- and space-consuming for what we propose here.

There have been utilizations of sets of terms, instead of single words, as index in IR [71]. However, sets of terms are only used to replace the original single index terms and no relation is established between sets of terms. Therefore, this approach only tries

to build a more precise representation for a document, but it fails to find documents described with different terms, which is our goal in this study.

The general context-dependent term relations we desire to extract are of the following form:

$$\{...t_j, t_k...\} \rightarrow t_i$$

which means when we observe the terms $\{...t_j, t_k...\}$ together, we can conclude in t_i. Here $\{...t_j, t_k...\}$ simply means that these terms appear together in a query or within a window. No other constraint is imposed on the relationship between them.

The condition part of the above relation can be arbitrarily long. In practice, however, we do not need to create long conditions. This is because:

- In most cases of ambiguous words, the addition of one useful context word suffices to disambiguate it;
- When the condition of a relation becomes longer, its applicability also becomes more limited.

Therefore, we can limit the condition to only two terms:

$$\{t_j, t_k\} \rightarrow t_i$$

In this study, the above term relations will be extracted from the document collection based on co-occurrences (see Chapter 5). In our experiments (Chapter 8), we will show that it is not useful to include more than 2 terms in the condition of the relation.

This latter form of relation infers a new term from a combination of two terms occurring in the same context. The two terms do not have to be adjacent. This is different from the notion of biterm defined in [99]. However, for convenience, we will still call this type of relation *biterm relation*. In our following description, we will extend the notion of biterm to be "two terms co-occurring in the same window". This

notion is much relaxed from that of [99]. It can be compared to the skip *n*-gram model used in NLP [35], but without word order. In contrast, the traditional relation $t_j \rightarrow t_i$ is called *unigram relation.*

When a biterm in the query matches the biterm of a relation, the relation can be applied. In this case, we are more certain that the deduced expansion terms are more related to the query than that deduced from the traditional unigram relations. For example, when we deduce "*programming*" from "{*Java, computer*}", we are more certain of its correctness than when "*programming*" is deduced from "*Java*" alone. The expansion terms suggested by biterm relations tend to be more relevant.

This type of term relation exploits the word contexts within the query or *intra-query context.* Many queries contain intra-query context. As Jensen et al. showed in their study [46], 64% of the user queries on the Web contain at least 2 words. For this part of the queries, context-dependent term relations can be applied.

In fact, users often do not use a single ambiguous word such as "*Java*" as query (if they are aware of its ambiguity). Some context words are often used together with it. In these cases, contexts within query are created and can be exploited.

Now let us describe the differences between context-dependent term relations and some related work.

- **Differences from compound terms:**

  In the above definition of context-dependent term relations, we deliberately used $\{...t_j, t_k...\}$ to mean co-occurrence of terms. It is not intended to mean stronger relations between these terms so that they can form a compound term, such as "*computer architecture*".

  The idea of using compound terms as the condition of a term relation also seems intuitive. However, in order to see why this does not work in IR, let

us first look at the way that compound terms or collocations are determined. One usually considers the two following criteria (e.g., [64][94]):

- Compound terms should fit in some syntactic structure;
- Meaningful compound terms should occur frequently enough in a document or document collection.

The general approach is to use syntactic patterns to identify candidate compounds first; then those that appear frequently are selected. This approach has been tested in [29] by Fagan. For example, syntactic patterns such as (*NN*, *NN*), (*ADJ*, *NN*) (where *NN* represents a noun and *ADJ* an adjective) are defined manually to identify candidates. However, Fagan also showed that this syntactic approach to determine compound is less effective than a purely statistical approach. By grouping strongly co-occurring words together to form statistical compound, larger improvements were observed in retrieval effectiveness.

Although more sophisticated methods have been employed later, which exploit more sophisticated NLP techniques, such as POS-tagging and sentence parsing, we are still facing the same problems:

- The basic assumption behind the utilization of compound terms to represent documents and queries is that they represent more precise meaning or concept. However, the detection is not 100% accurate and non-compound terms can be wrongly determined. For example, in the segment "*...with... powerful computer, scientists can work faster...*", it is possible that the process detect wrongly "*computer scientist*" as a compound term, while the correct term is "*powerful computer*";
- Compound terms do not always take a fixed form. They are highly variable. For example, the term "*online bookstore*" can also be expressed as "*bookstore on the Web*" or "*virtual bookstore*". It is difficult to recognize all forms of compounds and to relate each of them with others;

- Many queries do not follow a strict syntactic structure. They are only concatenation of words such as "*hotel waterfront*". It is often impossible to recognize correctly a compound term from them.

The approach we advocate is more flexible. It can tolerate different word orders and a larger distance between terms, and it can cover more interesting groups of terms than stricter compound terms.

- **Relations with word sense disambiguation and discrimination:**

Our proposed approach follows the same principle as Yarowsky's study [113], which tried to determine the appropriate word sense according to one relevant context word in the sentence. However, there are two important differences: (1) the requirement for query expansion is less than word sense disambiguation: we do not need to know the exact word sense to make expansion. We only need to distinguish different cases and to determine the relevant expansion terms in each of them; (2) Yarowsky determined word senses, but we determine related terms.

To some extent, our approach is related to word sense *discrimination* of Schütze and Pedersen [88]. In their approach, Schütze and Pedersen considered that an ambiguous word can be discriminated from other meanings by its context words (that co-occur in the same windows). Therefore, each word sense can be defined implicitly by a vector of context words. Given a query (of certain length) and a document, it is then possible to determine if the words denote the same meaning by comparing their context vectors.

To some extent, the approach of Schütze and Pedersen corresponds to a second-order term relations: two terms are considered to be related if they co-occur with the same context words. We will describe and test later an approach (i.e., information flow) to create second-order term relations.

In our approach, although we also exploit a similar idea to word sense discrimination, we use it to derive first-order term relations with other terms. Therefore, the utilization is very different. We will show in our experiments that our relations are more effective than the second-order term relations.

4.4 Extra-Query Context: Domain of Interest and User Profile

When a user issues a query, the query should be interpreted in the corresponding domain of interest. This latter provides a background for the interpretation of a query. One can see at least two types of element in a domain:

- A domain contains a set of domain-specific knowledge (i.e., term relations). For example, in *Computer Science*, "*Java* $\rightarrow$ *programming*" is a valid relation, thus can be included in *computer*-related background;
- A domain contains a set of frequently used specific terms. It reflects a set of specific background terms for a domain, for example "*pollution*", "*rain*", "*greenhouse*", etc. for the domain of *Environment*. These terms are often presumed when a user issues a query in the domain, such as "*waste cleanup*".

These two types of element suggest two possible utilizations of domain: (1) using domain-specific knowledge for query expansion; (2) using domain-specific terms to complement the query.

The first utilization is similar to the utilization of other types of term relations in query expansion. Intuitively, this approach seems to be a reasonable way to deal with ambiguities in query expansion. For example, one can extract co-occurrence term

relations from a specific domain, and use these relations to expand queries in that domain.

However, the application can also be limited in coverage: queries can also contain general terms in addition to domain-specific terms. Domain-specific knowledge does not apply to the latter.

In our experiments presented in a later chapter, we will test both approaches to use domains. Our experiments will show that domain-specific term relations are less effective than general, context-dependent term relations. So we will mainly describe the utilization of a domain as a set of specific terms. This strategy has been used in most previous studies on personalized IR.

One way to take into account the user's domains of interest is by personalization. A user profile is constructed to reflect the domains of the user [68]. However, as we mentioned earlier, a single user profile is created for a user without distinguishing the different topic domains. The systematic application of the user profile can incorrectly bias the results for queries unrelated to the profile.

A possible solution to this problem is the creation of multiple profiles, one for a separate domain of interest [58]. In this study, we will use this second approach and model topic domains. We will propose different ways to construct domain models and to determine the corresponding domain for a query (query classification problem).

In the following discussions, to contrast with the *intra-query context* that we introduced, the topic domain of a query will be called an *extra-query context* since it is an element outside the query.

4.5 Extra-Query Context: Feedback Model and Query's Collection Context

Another extra-query context is collection characteristics related to the query. Many attempts have been made in IR to create query-specific profiles that reflect some collection characteristics. The most common method is based on implicit feedback or blind feedback [22][52][90][111][115].

In [90], Shen et al. exploited the implicit user relevance feedback to re-rank documents. When the user clicks on a document, a relevance judgment is implicitly made. Although the clicked document is not always relevant, most users do click on documents which they think may be relevant. So these documents at least tend to be more relevant than the others. Shen et al. considered the clicked document as a "*relevant*" one, and the document in the result list are re-ranked accordingly. Indeed, Shen's approach is similar to pseudo relevance feedback, except that the feedback documents used are also selected (so somehow judged) by the user.

As we mentioned earlier, by incorporating the blind feedback documents into a query model, we can indeed enhance the query model with some collection characteristics. The feedback documents help us determine the topics related to the queries that are described in the collection. The additional information brought by these documents forms a *query-related collection context*. However, to correspond to the literature, we will continue to use *feedback* model to designate it.

4.6 Context-Sensitive Language Modeling for IR

We have described three types of query context that we try to consider in this study. Of course, there are many more, but we will limit ourselves to these three, which are representative and readily usable.

The next question is how they can be incorporated into a retrieval model. Many previous studies have used heuristics to combine contexts with a retrieval model. For example, user profile is used to re-rank the retrieval results according to a heuristic function. In this study, we try to explore a more principled integration method.

We will use language modeling as our basic modeling framework. This choice is motivated by several reasons:

- **Solid theoretical foundation:**

  The score function is not based on heuristics, but on a theoretical foundation. Even when simplifications are made, no heuristic factors need to be introduced.

- **Robustness to noise:**

  Language models can extract the most important elements from a data set that contains noise (i.e., terms not related to the topic). In our case, a document can contain some words that are not related to the topic of the document, together with the topic words. However, the language modeling framework can tolerate such noise, and the most important elements can emerge.

- **Extensibility:**

  The general language modeling framework can be easily extended to include new components. As we described in Section 3.3.4, the basic language models have been extended to incorporate term relations and pseudo relevance

feedback. The same framework can be further extended to integrate more contextual factors.

Our goal here is to create an enhanced query model by integrating all the available context information about the query. The enhanced query model corresponds to a more complete query description that represents better the information need, and is better suited to the document collection (through the incorporation of a feedback model). This approach was first proposed in our recent study [6].

Let us assume a query model θ_Q^0 before its enhancement. This model is created from the original query Q by using MLE (unigram model). The three contextual factors used to enhance the query are as follows:

- The knowledge or term relations are applied to the query. This will result in additional terms, which constitute a new language model θ_Q^K for the query. We have $P(t \mid \theta_Q^K)$ such that $\sum_{t \in V} P(t \mid \theta_Q^K) = 1$. If the traditional term relation $t_j \rightarrow t_i$ or its probabilistic version $P(t_i \mid t_j)$, is used, then $P(t_i \mid \theta_Q^K)$ is defined as follows:

$$P(t_i \mid \theta_Q^K) = \sum_{t_j \in V} P(t_i \mid t_j) P(t_j \mid \theta_Q^0)$$

If the context-dependent term relation $\{t_j, t_k\} \rightarrow t_i$ is used, then $P(t_i \mid \theta_Q^K)$ is defined as follows:

$$P(t_i \mid \theta_Q^K) = \sum_{t_j t_k \in V} P(t_i \mid t_j t_k) P(t_j t_k \mid \theta_Q^0)$$

In Chapter 5, we will describe in more details the ways to construct these models;

- A domain model is created from a set of documents pertaining to the domain. A domain model is another language model θ_Q^{Dom} created from these documents. We will describe how this model is created in Chapter 6. A domain model is considered as a specification of the background aspects of the query;
- Finally, a set of feedback documents will be used to create a feedback model θ_Q^F, which reflects the way that the query's topic is developed in the document collection.

We consider that each of the above models specifies a different aspect of the query. Therefore, they can be combined to produce a final query model. In this study, we propose the following interpolation for building the final query model:

$$P(t \mid \theta_Q) = \sum_{i \in X} \alpha_i P(t \mid \theta_Q^i)$$

where $X = \{0, K, Dom, FB\}$ is the set of all component models, and α_i (with $\sum_{i \in X} \alpha_i = 1$) is a mixture weight, which controls the importance of each component model (the tuning of the parameter α_i will be described in Chapter 7).

Given the above final query model, we use a score function based on KL-divergence to score documents:

$$\begin{aligned} Score(Q,D) &= \sum_{t \in V} [\sum_{i \in X} \alpha_i P(t \mid \theta_Q^i)] \log P(t \mid \theta_D) \\ &= \sum_{i \in X} \alpha_i \sum_{t \in V} P(t \mid \theta_Q^i) \log P(t \mid \theta_D) \end{aligned}$$

Let us define the following score function according to each of the component models:

$$Score_i(Q,D) = \sum_{t \in V} P(t \mid \theta_Q^i) \log P(t \mid \theta_D)$$

Then the final document score can be rewritten as follows:

$$Score(Q,D) = \sum_{i \in X} \alpha_i Score_i(Q,D)$$

This is indeed a combination of the scores determined by all the component models. It is similar to the document re-ranking strategy used in most previous studies [12][48][101].

The final query model can be illustrated as an inference process through different paths (models) as in the following figure:

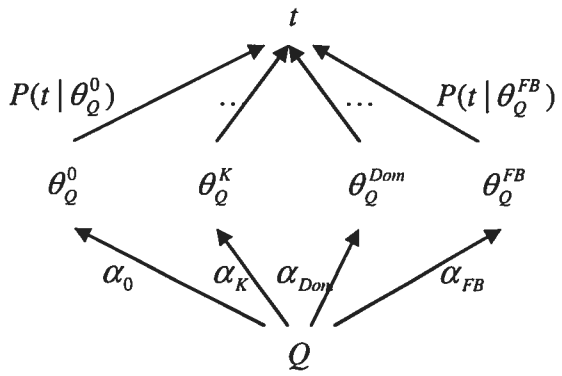

Figure 5. An illustration of general query model: term *t* can be inferred from the query model in several ways

Each of the branches from Q constitutes a choice of the inference path. The value α_i corresponds to the probability that the path is selected. Once a path is selected, the corresponding language model is called to generate the term *t*.

Discussions:

- In the above method, we assume that each component model is considered to describe a different aspect of the query. They are complementary;

- The component models are combined through a simple interpolation. Indeed, this also means that the component models are independent, as illustrated in the above figure. One may question about this combination method. It is indeed not true that each component model acts on the query (or information need) independently. A model can interfere with another model. For example, as we mentioned, the knowledge model can be part of the domain model. However, few studies have addressed the problem of combining different contextual factors. Interpolation seems to be a reasonable way to combine them in such a situation;
- The main focus of this study is to see whether different contextual factors can contribute to improving retrieval effectiveness. We do not seek to define the best way to combine them in this study. The combination method used is simple. It can be improved in the future.

4.7 Logical View of the Generalized Model

The processes that we use to enhance our general query model are based on *inference*: we try to infer related terms from the query in different ways so that the final query model corresponds to a better and more complete representation of the query (or information need). The retrieved documents will not be restricted to those that contain the same terms as the initial query, but may also contain different related terms. This reflects the increased retrieval capability due to inference.

One may also think that the traditional LM has also such a capability: using document smoothing with the collection model, a document also does not have to contain the query terms to be retrieved. However, it is important to understand that the document smoothing process is completely query-independent and it is performed

regardless to the relationships between terms: the probability of the added terms is solely determined according to their distribution in the collection, and they may not be related to the topic of the document.

For example, given a document on "*natural language processing*", when it is smoothed with the collection model, general terms such as "*building*", "*shopping*", "*sea*", etc. may all receive the probability similar to a strongly related term such as "*syntax*". Even if the smoothing process allows a document to match a query without containing all the query terms, this process is not *truly* a logic inference.

Logic inference is usually made according to some basic logic relations, or implications. The model that we propose strongly relies on logic inference. In this section, we will describe our approach from the perspective of logic inference. The goal is to explain the basic idea that guided us to this model.

The basic inference process is specified by the following deduction in classic logic:

$$S|-A \rightarrow B$$

where S is a data set, knowledge or context, A is a logical expression – the premise or condition, and B is another logic expression – the consequence or conclusion. If the above expression is valid, then we say that A implies B, or B is inferred or deduced from A, given the context S.

In the classic logic, we also have the following equation:

$$S|-A \rightarrow B \equiv S \cup A\,|-B$$

That is, given S, if we know that $A \rightarrow B$ is a valid implication relation, then B is also valid in the context $S \cup A$, and vise versa.

This expression can well describe the essence of our approach: a query Q plays the role of A in the above expression, a new term t corresponds to B, and the three contextual factors play the role of S. If t can be inferred in this way, then we can

consider it as a valid expansion term for Q in the context of S. Such an interpretation has been considered in several previous studies [20][104].

The above deduction in classic logic does not take into account uncertainty and non-monotonicity in inference. In order to consider uncertainty in IR, van Rijsbergen [104], proposed the following *uncertainty* principle:

"*Given any two sentences x and y; a measure of the uncertainty of* $y \rightarrow x$ *relative to a given data set, is determined by the minimal extent to which we have to add information to the data set, to establish the truth of* $y \rightarrow x$."

The above process is related to the Ramsey test in conditional logic [56], which states that, to test if a conditional $A \rightarrow B$ is true in a certain situation S, one first has to change the situation S minimally so as to satisfy A. Then we test if B is true in the new situation. This corresponds to the essence of the earlier equation $S|-A \rightarrow B \equiv S \cup A|-B$. However, the difference appears when A and S are inconsistent, i.e., there are pieces of elements in S that are contradictory to A. In this case, $S \cup A|-B$ is valid for whatever B in classic logic. In Lewis conditional (or counterfactual) logic [56], the validity of this expression (with a different type of implication) will be determined in a different way: one first determines a new context S' that is the "closest" to S and in which A is true, then we check if B is true in S'. This process allows produce non-monotonic reasoning, i.e., it is allowed to have $S|-A \rightarrow B$ but not $S|-A \cup C \rightarrow B$, while in classic logic, this may not happen. For example, when C added is inconsistent with S, a part of the statements in S should be removed to accommodate C. This may lead to the failure of deducing B. Therefore, the addition of a new condition C may invalidate the implication.

The uncertainty principle of van Rijsbergen can model similar phenomena. It tries to determine the uncertainty according to ΔS to be added minimally to S so that we

can establish the truth of $S \cup \Delta S | - A \rightarrow B$. The larger is ΔS, the more uncertain is $S | - A \rightarrow B$.

However, the consideration of non-monotonicity in reasoning is beyond the scope of our study. Our aim is more limited: we only aim to model the basic inference process in IR corresponding to that in classic logic. The investigation of non-monotonicity in reasoning in IR requires more sophisticated tools than statistical language modeling. We leave it as a future work.

Going back to the classic deduction and always without taking into account the uncertainty aspect, our goal in this work is to determine t such that $S | - Q \rightarrow t$ holds. The context S is further split into three components: K (user knowledge), *Dom* (topic domain) and *FB* (feedback documents). In this case, the underlying inference process that we propose can be expressed as follows:

$$K \cup Dom \cup FB | - Q \rightarrow t$$

where t represents an inferred term, and $| -$ represents an inference relation. The above expression is interpreted as follows: within the context of K, *Dom, FB,* the term t is inferred from the query Q.

A further simplification that we made is to consider the three contextual factors and original query model separately using the following principle:

$K \cup Dom \cup FB | - Q \rightarrow t$ if $| - Q \rightarrow t$ or $K | - Q \rightarrow t$ or $Dom | - Q \rightarrow t$ or $FB | - Q \rightarrow t$

This principle is implemented within the language modeling framework as follows: each of the expression on the right hand side, i.e., $| - Q \rightarrow t$, $K | - Q \rightarrow t$, $Dom | - Q \rightarrow t$ and $FB | - Q \rightarrow t$, corresponds to a language model. The combination of the "*or*" relation is implemented as an interpolation of these models.

The connection between the generalized language model and the inference relation can be seen as follows: the language model can be viewed as a set of (weighted)

inference relations. If a term has a non-zero probability in a language model for query Q, i.e., $P(t \mid \theta_Q) > 0$, then we can consider that the following inference can be made to some extent: $Q \mid\!- t$. With this interpretation, the final query model we wish to build corresponds to a language model that infers the term t through several paths. Each of the models on the right side of the above expression can be understood in the following way:

$$\mid\!- Q \rightarrow t \equiv Q \mid\!- t \cong P(t \mid \theta_Q^0)$$
$$K \mid\!- Q \rightarrow t \equiv K \cup Q \mid\!- t \cong P(t \mid \theta_Q^K)$$
$$Dom \mid\!- Q \rightarrow t \equiv Dom \cup Q \mid\!- t \cong P(t \mid \theta_Q^{Dom})$$
$$FB \mid\!- Q \rightarrow t \equiv FB \cup Q \mid\!- t \cong P(t \mid \theta_Q^{FB})$$

where $\cong$ means the corresponding element in language modeling approach.

So without considering the uncertainty of probability, the above expression can be expressed as follows:

$$P(t \mid \theta_Q) > 0 \text{ if } P(t \mid \theta_Q^0) > 0 \text{ or } P(t \mid \theta_Q^K) > 0 \text{ or } P(t \mid \theta_Q^{Dom}) > 0 \text{ or } P(t \mid \theta_Q^{FB}) > 0$$

where $P(t \mid \theta_Q)$ is the final query model that takes into account all the contextual factors.

However, we need to consider this uncertainty in a more refined way, and we choose to use interpolation.

Let us look at each of the component model in some more details:

- $\mid\!- Q \rightarrow t$:

  This expression means that term t can be directly inferred from the query Q without any extra condition. This corresponds to the case where t appears in Q. So the expression corresponds to the traditional query model. In terms of language modeling, it corresponds to $P(t \mid \theta_Q^0)$, which can be estimated by MLE.

- $K \mid\!\!- Q \rightarrow t$:

This part of the inference strongly corresponds to the classical inference process in logic: t is inferred from Q by applying a set of knowledge K on the latter. Here we assume that K is a set of term relations of the following form:

$$T \rightarrow t$$

where T is a term or a set of terms and t is another term. This relation means that t can be inferred from T. In terms of language modeling, this relation can be represented as the probability function $P(t \mid T)$. For example, "$\{air\ hijacking\} \rightarrow airplane$" correspond to $P(airplane \mid air, hijacking)$ in language modeling.

When K is applied to a query, say on "*air hijacking*", we can infer the term "*airplane*". Then "*airplane*" can be added into the query. This process is usually seen as query expansion In IR.

The inference $K \mid\!\!- Q \rightarrow t$ is indeed made of two steps: from Q we infer some term or a set of terms T (e.g., a biterm in the query); then from T we infer the term t by applying knowledge K. So we have the following logic relation:

$$K \mid\!\!- Q \rightarrow t \text{ if } \mid\!\!- Q \rightarrow T \text{ and } K \mid\!\!- T \rightarrow t$$

This corresponds indeed to the following relation in classic logic:

$$(A \rightarrow B) \wedge (B \rightarrow C) \mid\!\!- A \rightarrow C$$

In terms of language modeling, by considering all the inference paths, the above knowledge model is then built as follows:

$$P(t \mid \theta_Q^K) = \sum_{T \in Q} P(t \mid T) P(T \mid \theta_Q^0)$$

We will describe this model in more details in Chapter 5. The inference paths in this model can be illustrated in the following figure:

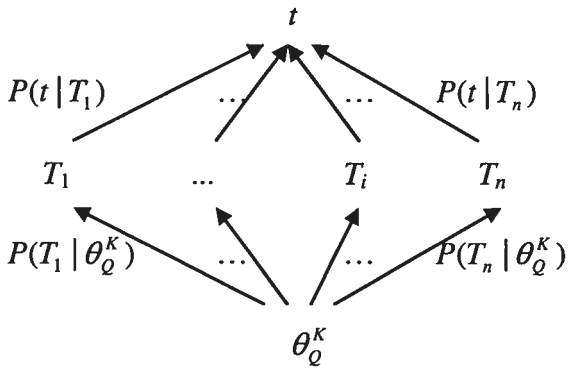

Figure 6. An illustration of knowledge model: term *t* can be inferred through different terms T_n

- ***Dom* |- $Q \rightarrow t$:**

In our modeling, *Dom* corresponds to a set of terms. For example, given the *Terrorism* domain, we may have the following terms: *Dom* = {"*attack*", "*terrorist*", "*kidnapping*", "*bomb*", ...}. These terms are assumed to be related to any query in this domain. We can interpret the terms as having the following relations for any Q in this domain:

$$Dom \mid\!- Q \rightarrow attack$$

$$Dom \mid\!- Q \rightarrow terrorist$$

...

In fact, the role of the query is to determine the correct domain Dom_Q. Once this is done, the subsequent inference from *Dom* is independent of it. So we can write:

$$Dom_Q \mid\!- attack$$

$$Dom_Q \mid\!- terrorist$$

...

Therefore, the inference process using the query domain can be expressed as:

$$Dom_Q \mid\!- t$$

In terms of language modeling, this corresponds to $P(t \mid \theta_Q^{Dom})$, where the index Q only means that the domain is selected for this particular query.

- ***FB* |- $Q \rightarrow t$:**

By the feedback documents, we try to exploit some collection characteristics. So what we intend to have is $C \mid\!- Q \rightarrow t$, i.e., in the given collection C, how the term t can be inferred from Q.

The way we exploit collection characteristics is through feedback documents, which is a subset of documents related to Q. Let us write the feedback documents for query Q as FB_Q. So we have:

$$C \mid\!- Q \rightarrow t \text{ if } FB_Q \mid\!- t$$

Once the feedback model is determined, it acts on the retrieval process in a similar way to the domain model and a language model $P(t \mid \theta_Q^{FB})$ is built accordingly.

In order to correspond to the literature, we will continue to use feedback model (*FB*) to denote this model related to collection characteristics.

At this point, it is interesting to compare with the traditional smoothing with the collection model. Indeed, the smoothing simply assumes:

$$C \mid\!- Q \rightarrow t \text{ if } C \mid\!- t$$

We can see that the smoothing process $C \mid\!- t$ is completely independent from the query Q. Frequent terms in the collection are attributed higher probabilities for whatever query. This utilization is obviously unreasonable from

a logic point of view. This also explains once again the fact that traditional smoothing is not a true inference from the query.

The above logical interpretation of the generalized language model is much simplified. We have not fully considered the uncertainty involved. Our aim is to draw a picture of what we intend to implement. The ultimate goal is to exploit various resources and information to infer what a query (or information need) corresponds. Any additional contextual factor, once properly modeled, can be integrated in a similar way.

Limitations:

However, the above description also shows some limitations of the model that we propose, due to several simplifications:

- The components are considered separately during the inference process. In reality, contextual factors can interfere with each other. For example, the applicability of a piece of knowledge can depend on the topic domain;
- Elements within each context model are considered to be independent. For example, pieces of knowledge are considered to be independent (as can be seen in Figure 6). Terms in a domain model or feedback model (a language model) are also considered to be independent.

The above simplifications are not always reasonable. Indeed, many elements in the contexts are dependent. However, if we take into account all the dependencies, the models will become very complex and inefficient. So the above simplifications are made for the sake of efficiency, in the same line as the traditional language models where terms are assumed to be independent. This is also in contrast with several previous studies on logical modeling of IR [51], which aim to have higher descriptive power, with the detriment of efficiency.

In our modeling, we assumed that different pieces of information cannot be contradictory. Therefore, a modeling in the line of classic logic is used. The assumption is not always true in reality. Pieces of information can be contradictory. To deal with this problem, it would be necessary to employ non-monotonic reasoning. Approaches have been proposed in [51][53]. However, it is still not clear how non-monotonic reasoning can be efficiently managed in IR on large document collections. It is also not clear whether the language modeling framework can be extended to account for this problem. This would be an interesting problem to investigate in the future.

In Section 3.3.4, we have described ways to exploit feedback documents in language modeling. In this study, we will use the mixture model proposed by Zhai and Lafferty [115], which produced good results in previous studies.

In the next two chapters, we will describe in more details how the other two models – *knowledge* model and *domain* model – are constructed and used.

Chapter 5

Implementing Intra-Query Context

In this chapter, we will describe how context-dependent term relations are constructed and how they are integrated into the general language modeling approach. We will start by describing the implementation of traditional query expansion approach in LM, which uses context-independent term relations. Then the approach will be extended to integrate our context-dependent term relations.

5.1 Traditional Query Expansion in Language Modeling

Term relations have been used in several recent language models in IR. Berger and Lafferty [10] proposed a *translation model* that expands the document model. The same approach can also be used to expand the query model. Following [10], we arrive at the *co-occurrence* model for query as follows:

$$P(t_i \mid \theta_Q^{co}) = \sum_{t_j \in V} P(t_i, t_j \mid \theta_Q^{co}) = \sum_{t_j \in Q} P_{co}(t_i \mid t_j) P(t_j \mid \theta_Q^0)$$

where $P_{co}(t_i \mid t_j)$ denotes the co-occurrence relation between two terms.

This is a simple knowledge model. In this model, each original query term t_j is expanded by related terms t_i. The relations between them are determined by $P_{co}(t_i \mid t_j)$. We will explain how this probability is defined later.

Document score according to the co-occurrence model is as follows:

$$score_{co}(Q,D) = \sum_{t_i \in V} P(t_i \mid \theta_Q^{co}) \log P(t_i \mid D)$$
$$= \sum_{t_i \in V} \sum_{t_j \in Q} P_{co}(t_i \mid t_j) P(t_j \mid \theta_Q^0) \log P(t_i \mid D)$$

However, if the query model is expanded on all the vocabulary ($t_i \in V$), the query evaluation will be very time consuming since the query and the document have to be compared on every term in V.

In practice, we observe that only a small number of terms have strong relations with a given term, and the terms having weak relations usually are not truly related. So we can well limit the expansion terms only to the strongly related ones. By doing this, we can also expect to filter out some noise and considerably reduce the retrieval time.

Suppose that we have selected a set E of strong expansion terms. Then we have:

$$score_{co}(Q,D) \approx \sum_{t_i \in E} \sum_{t_j \in Q} P_{co}(t_i \mid t_j) P(t_j \mid \theta_Q^0) \log P(t_i \mid D)$$

Here the expansion terms are determined according to their relations to all the query terms (the sum over $t_j \in Q$). This is the same method as that used by Qiu and Frei [73], but in a LM setting. So our later comparison with this model also reflects the comparison with the method of Qiu and Frei.

The next question is how to compute $P_{co}(t_i \mid t_j)$. We have described two main families of relations (beside relevance feedback): *thesauri* and *co-occurrence*. Thesauri have been used in several studies [15][105][106]. However, mitigated results are obtained. Cao et al. [15] showed that they can help improve retrieval effectiveness,

while Voorhees [105][106] showed that the retrieval effectiveness is rather degraded. Even when thesaurus relations have shown to be able to improve retrieval effectiveness in [15], their impact is smaller than co-occurrence relations. Therefore, we will focus on co-occurrence relations in this study. This does not mean, however, that other types of relation cannot be also applied. Another reason for using co-occurrence relations is that they allow us to compare easily context-dependent term relations to context-independent term relations.

To estimate term co-occurrences, typically, one counts the frequency $c(t_i, t_j)$ that terms co-occur within a certain context such as a window of fixed size. Then the strength (or probability) of term relation is calculated as follows:

$$P_{co}(t_i \mid t_j) = \frac{c(t_i, t_j)}{\sum_{t_l} c(t_l, t_j)}$$

The extraction of such relations from a document collection is simple. The time and space requirements are also not very high, with an upper bound of $O(|V|^2)$, where $|V|$ is the vocabulary size. Typically, for a TREC collection, the vocabulary size is in the order of $100K$.

We show here an example with terms "*space*" and "*program*". The co-occurrence relations are extracted from the collection AP88-89 from TREC (see Chapter 8 for more details about the collection):

***space*:**

***shuttle*:0.0140 *launch*:0.0091 *nation*:0.0078 *soviet*:0.0076 *program*:0.0075 *flight*:0.0065 *year*: 0.0064 *center*:0.0064 *station*:0.0064 *nasa*:0.0060 *administration*:0.0057 *mission*:0.0048 *aeronautic*:0.0043 *astronaut*:0.0042 *agency*:0.0041 *US*:0.0040 *rocket*:0.0040 *office*:0.0038 *orbit*:0.0036 *challenge*:0.0035 *satellite*:0.0034 *system*:0.0033 *time*:0.0032 *discovery*:0.0032 *man*:0.0032 *plan*:0.0030 *defense*:0.0030 *million*:0.0030 *day*:0.0030 *develop*:0.0029 *base*:0.0029 *state*:0.0029 *president*:0.0029 *air*:0.0028 *work*:0.0026**

research:0.0026 earth:0.0026 test:0.0026 ***missil****:0.0025 include:0.0024 american:0.0023 unit:0.0023 office:0.0023 report:0.0022 make:0.0021 engin:0.0021 people:0.0021 build:0.0021 crew:0.0020 …*

***program*:**

year:0.0085 state:0.0055 million:0.0054 govern:0.0047 federal:0.0040 nation:0.0040 billion:0.0037 drug:0.0036 school:0.0034 include:0.0033 house:0.0032 percent:0.0032 people:0.0031 president:0.0030 educate:0.0029 depart:0.0029 work:0.0029 bush:0.0029 time:0.0028 develop:0.0028 office:0.0028 US:0.0028 plan:0.0028 aid:0.0027 call:0.0026 report:0.0026 fund: 0.0026 service:0.0025 administration:0.0025 cut:0.0025 cost:0.0023 company:0.0023 support:0.0023 make:0.0023 television:0.0022 money:0.0022 show:0.0021 part:0.0021 space:0.0021 spend:0.0021 budget:0.0021 congress:0.0020 increase:0.0020 propose:0.0020 student:0.0020 health:0.0020 american:0.0020 soviet:0.0019 country:0.0019 test:0.0019 new:0.0019 …

We can see that the terms co-occurring with "*space*" are surprisingly related to the meaning of "*universe space*". This is because many articles in the AP collection talk about this topic and fewer talk about "*space and time*" in general. For the term "*program*", we see much more diverse co-occurring terms. This term has been used in different contexts, for "*government programs*", "*school programs*", "*space program*", and so on.

For a query on "*space program*", the related terms are determined as follows:

$$P(t_i \mid \theta_Q^{co}) = \sum_{t_j \in Q} P_{co}(t_i \mid t_j) P(t_j \mid \theta_Q^0)$$

We assume $P(t_j \mid \theta_Q^0)$ to be uniform. So for "*space program*", the set of related terms t_i is determined by a weighted union of the above two sets:

***space program*:**

year:0.0074 ***shuttle****:0.0073 nation:0.0059* ***launch****:0.0049* ***soviet****:0.0048 million: 0.0042 state:0.0042 administration:0.0041* ***program****:0.0037 center:0.0037* ***station****:0.0037* ***flight*** *0.0035* ***US****:0.0034 office:0.0033* ***nasa****:0.0032 govern:0.0030 time:0.0030 president:0.0029* ***agency****:0.0029 plan:0.0029 develop:0.0029 billion:0.0028 include:0.0028 work:0.0027 people:0.0026 house:0.0025* ***mission****:0.0025 system:0.0025 federal:0.0024 report:0.0024* ***defense****:0.0024 day:0.0024 percent:0.0023 bush:0.0023* ***test****:0.0022 call:0.0022 make:0.0022* ***aeronautic****:0.0022* ***astronaut****:0.0021 american:0.0021* ***base****:0.0021 school:0.0021 unit:0.0020 research:0.0020 challenge:0.0020 drug:0.0020 depart:0.0019 air:0.0019 company:0.0019 month:0.0019 ...*

The most relevant terms to the query are in bold. As we can see, some of the expansion terms are relevant, such as "*shuttle*", "*launch*", etc., but we also have many irrelevant terms such as "*year*", "*office*", "*time*", "*include*", etc. Indeed, the simple combination of the two sets is unable to determine which expansion term is related to the whole query. The method proposed by Qiu and Frei [73] has the same problem.

5.2 Context-Dependent Query Expansion

Recall that our general context-dependent term relations correspond to the following form:

$$\{...t_j, t_k...\} \rightarrow t_i$$

It is assumed that the more we put terms into the condition, the more strongly the inferred term is related. However, when the condition contains more terms, the complexity of the extraction process also increases: with n terms in the condition, the relation requires a time and space complexity of $O(|V|^{n+1})$. As we said earlier, it is not

very useful to include many terms in the condition. Usually two terms together are sufficient to describe a quite precise meaning. Therefore, we will mainly focus on the relations with two terms (or a biterm) as follows:

$$\{t_j, t_k\} \rightarrow t_i$$

or on its probability $P_K(t_i \mid t_j t_k)$.

5.2.1.1 Extraction of Term Relations

The extraction of biterm relations of form $\{t_j, t_k\} \rightarrow t_i$, or $P_K(t_i \mid t_j t_k)$, can also be performed using co-occurrence analysis. We determine the co-occurrence frequency $c(t_i, t_j, t_k)$ of three terms within the same window. The probability $P_K(t_i \mid t_j t_k)$ is computed as follows:

$$P_K(t_i \mid t_j t_k) = \frac{c(t_i, t_j, t_k)}{\sum_{t_l} c(t_l, t_j, t_k)}$$

The number of relations determined in this way can be very large, with un upper bound of $O(|V|^3)$. However, many relations have very low probabilities and are often noise. In order to reduce space requirement, we can simply consider a subset of strong expansion terms, the relations with low probability are almost never used. Therefore, we further apply the following filtering criteria on relations:

- The two terms in the condition should appear at least certain times together in the collection (10 in our case) and they should be related. There are many measures of term relatedness, such as *mutual information*, *χ-square*, *information gain*, etc. Here we use the following *pointwise* mutual information as a measure of relatedness, which proved to be effective in [19]:

$$MI(t_j, t_k) = \log \frac{P(t_j, t_k)}{P(t_j)P(t_k)}$$

If $MI(t_j, t_k) > 0$, we consider the terms to be related;

- The probability $P(t_i \mid t_j t_k)$ of a relation should be higher than a threshold (0.0001 in our case) to be kept.

By these filtering criteria, we are able to reduce considerably the number of relations. For example, on a collection of about 200*M*, with a vocabulary size of about 148*K*, we selecte only about 137*M* such relations, which remains tractable. Indeed, many of the remaining relations are still never used. Therefore, more severe filtering criteria can be used to reduce the time and space complexity further.

Below is an example showing the terms related to the biterm ("*space program*"):

(*space, program*):

shuttle:0.0174 ***soviet***:0.0146 *nation*:0.0124 ***station***:0.0105 ***US***:0.0098 ***man***:0.0093 *year*:0.0082 ***nasa***:0.0076 ***launch***:0.0069 ***flight***:0.0069 *administration*:0.0065 ***defense***:0.0064 *develope*:0.0063 ***challenge***:0.0055 *billion*:0.0053 *america*:0.0050 *budget*:0.0047 *center*:0.0046 ***aeronautic***:0.0044 *president*:0.0043 ***base***:0.0043 ***mission***:0.0042 ***war***:0.0041 *include*:0.0039 ***missil***:0.0038 ***rocket***:0.0038 *research*:0.0037 *state*:0.0037 ***astronaut***:0.0037 *agence*:0.0037 ***science***:0.0037 *house*:0.0035 ***star***:0.0035 *american*:0.0034 *money*:0.0033 *office*:0.0033 *increase*:0.0032 *spend*:0.0031 ***explore***:0.0031 *work*:0.0031 *reagan*:0.0030 *unit*:0.0030 *support*:0.0029 *fund*:0.0029 *time*:0.0028 *million*:0.0028 *bush*:0.0027 *cut*:0.0027 ***discovery***:0.0027 ***satellite***: 0.0022 ***booster***: 0.0022 ***orbit***: 0.0022 …

Compared to the expansion terms determined by simple co-occurrence relations, this set of terms is more related to the query: the underlined words, such as "*science*", "*explore*", "*satellite*", "*orbit*", etc. are the related terms which do not appear in

traditional co-occurrence relations. We would expect that query expansion with this set of terms is more effective than with the previous one.

Having a set of relations, the corresponding *knowledge* model is defined as follows:

$$P(t_i \mid \theta_Q^K) = \sum_{(t_j t_k) \in Q} P_K(t_i \mid t_j t_k) P(t_j t_k \mid \theta_Q^0)$$

where $(t_j, t_k) \in Q$ means a biterm in the query. Then the document score according to the knowledge model is defined as:

$$Score_K(Q, D) = \sum_{t_i \in V} \sum_{(t_j t_k) \in Q} P_K(t_i \mid t_j t_k) P(t_j t_k \mid \theta_Q^0) \log P(t_i \mid \theta_D)$$

Again, we can keep only a subset E of strongest expansion terms. Then the score function becomes:

$$Score_K(Q, D) \approx \sum_{t_i \in E} \sum_{(t_j t_k) \in Q} P_K(t_i \mid t_j t_k) P(t_j t_k \mid \theta_Q^0) \log P(t_i \mid \theta_D)$$

5.2.1.2 Determining Biterms

Another question is to determine the biterms in a query, i.e., $(t_j, t_k) \in Q$ in the previous expressions. Several approaches are possible:

- One can consider all the possible biterms in the query. Using this approach, we can cover all the biterms in a query;
- As *mutual information* reflects the relatedness of terms, it can also be used to select biterms. Among all the biterms in a query, we can select those whose mutual information is positive: $MI(t_j, t_k) > 0$;
- We can consider only the strongest biterms that cover minimally the whole query. For example, for the query "*US space program*", we would like to

extract "(*US program*)" and "(*space program*)" from it. These two biterms are the strongest ones that cover all the terms in the query. This idea is similar to *maximal spanning tree*: we consider each biterm as forming a link between terms, and the goal is to determine the strongest links that cover all query terms. The approach using maximal spanning tree has been used in a probabilistic model in [102][103]. A similar approach is also used in [32].

In our experiments, we will test all these strategies. However, experiments show that the first naïve method produces the best results.

5.2.1.3 Determining Biterm Probability

The probability of a group of terms in the query $P(q_j q_k \mid Q)$ can also be estimated in different ways:

- We can estimate it according to the relative frequency of the biterm in the query, i.e., compared to the total frequency of all biterms. This is equivalent to assign a uniform probability to each biterm, i.e.:

$$P(q_i q_k \mid Q) = \frac{1}{|Q|_B}$$

  where $|Q|_B$ is the number of biterms in Q;

- It can be weighted by *mutual information*. We can assume that, if two words are strongly associated, their association is more meaningful to the query, thus should be weighted higher. Therefore, a possible way to assign a probability to a biterm in the query is to use mutual information as its weight. Then we use the following normalized weight as biterm's probability:

$$P(q_j q_k \mid Q) = \frac{MI(q_j, q_k)}{\sum_{(q_l q_m) \in Q} MI(q_l, q_m)}$$

All the above weighting schemas are tested in our experiments. The first naïve method produces the best results.

We have described the utilization of context-dependent term relations whose condition is a biterm. If we do not limit the number of terms in the condition to two and allow more terms in the condition, the implementation is similar to biterm relations. But computation time and memory requirement are likely to explore exponentially.

5.3 Second-Order Term Relations

The term relations described so far are first-order relations, i.e., related terms should co-occur within the same windows of text. This type of relation cannot be established between true synonyms in some cases. Therefore, second-order term relations have been proposed [38]. Second-order relation is established between terms that co-occur with other similar terms. For example, "*cloth*" and "*coat*" can co-occur with similar words. Therefore, we can consider them as similar. As we described earlier, the second-order term relations have also been used in [38][88].

Here we would like to compare second-order term relations with context-dependent term relations. In this section, we will describe one particular method based on *information flow*, which is extracted from an analysis in HAL space. It is proposed by Bruza et al. in [14][96] and is believed to be effective. We have tested this type of relation in language modeling approach in [4].

5.3.1 Co-occurrence in HAL Space

HAL (*hyperspace analogue to language*) is a cognitively motivated and validated semantic space model for deriving term co-occurrence relations [14][96].

What HAL does is to generate a word-by-word co-occurrence matrix from a large text corpus via a *l*-sized sliding window: all the words occurring within the window are considered as co-occurring with each other. By moving the window across the text corpus, an accumulated co-occurrence matrix for all the words in a certain vocabulary is produced. The strength of association between two terms is inversely proportional to their distance. This idea is similar to the decaying factor according to distance used in [32].

An example showing the HAL space for the text "*the effects of pollution on the population*" using a 5-word moving window ($l = 5$) is depicted in the following table (stopwords have not been removed in this example).

Table 4. Example of a HAL space

	The	**effects**	**of**	**pollution**	**on**	**population**
the	1	2	3	4	5	
effects	**5**					
of	4	5				
pollution	3	4	5			
on	2	3	4	5		
population	5	1	2	3	4	

For example, the word "*the*" appears before "*effects*" in 5 sliding windows given the window size 5 (shown in bold in the table). The original HAL space is direction sensitive: the co-occurrence information preceding and following a word are recorded separately by the row and column vectors. However, for the purpose of deriving term relations in IR, word order does not seem to be important. Therefore, the HAL vector of a word is represented by adding up its row and column vectors.

Compared to the traditional co-occurrence analysis, HAL measure is more sensitive to distance between terms. For example, the co-occurrence count in HAL space between "*effects*" and "*pollution*" is 4, which is higher than that between "*effects*"

and "*population*", while in the traditional co-occurrence analysis, they are the same (i.e., 1). The effect of this distance factor in HAL is similar to the decaying factor used in [33], where the strength between two terms decreases when the distance between them increases.

For a given word, the dimensions in its HAL vector whose weights are higher than a threshold (set at the mean positive weight in our experiments) are called "*quality properties*" of the word.

To fit in the LM framework, a probabilistic HAL space can be estimated by normalizing a HAL vector by the sum of all the dimension weights:

$$P_{HAL}(t_i \mid t_j) = \frac{HAL(t_i \mid t_j)}{\sum_{t_k \in V} HAL(t_k \mid t_j)}$$

where $HAL(t_i \mid t_j)$ is the weight of t_i in the HAL vector of t_j.

Below are the probability values we obtain for the simple example:

pollution = {*the*: *0.29*, *of*: *0.21*, *on*: *0.21*, *effects*: *0.17*, *population*: *0.12*}

Different words can be combined to form more complex concepts like "*space program*". A vector is computed for this latter by combining the HAL vectors of the individual terms "*space*" and "*program*". However, instead of simply adding two vectors, Bruza et al. [14][96] used some heuristics:

- They consider that between "*space*" and "*program*", one concept dominate another. The dominant concept is the one whose *IDF* is higher (here "*space*"). Therefore, the vector of the dominant concept is attributed higher weight in the combination (see [14][96] for details);
- The weight of a word that appears in the HAL vector of both words is boosted (multiplied by 2). This is done to favor the common terms in both vectors.

The combined concept "*space program*" corresponds to the following vector:

***space program**:*

program:0.40 ***space**:0.38* ***shuttle**:0.37* ***station**:0.33 nation:0.23 center:0.18 administration:0.17* ***soviet**:0.17* ***aeronautic**:0.15 agency:0.15* ***flight**:0.13* ***nasa**:0.12* ***US**:0.11 man:0.10 new:0.09* ***base**:0.09* ***exploration**:0.08* ***rocket**:0.08* ***launch**:0.08 science:0.08 research:0.07* ***defense**:0.07 grant:0.07 billion:0.07 bush:0.07 council:0.06 america:0.06 president:0.06 technology:0.06 official:0.06 development:0.05 kennedy:0.05 million:0.05* ***missile**:0.04* ***mission**:0.04 probe:0.04 aboard:0.04 air:0.04* ***challeng**:0.04 commercial:0.04* ***astronaut**:0.04 budget:0.04 european:0.04 mir:0.04 quayle:0.04 star:0.04* ***unman**:0.04 johnson:0.04 marshall:0.03 director:0.03* ***earth**:0.03 house:0.03* ***war**:0.03 work:0.03 ...*

Compared to the expansion terms extracted from simple co-occurrence relations, this set of terms is also related to the query. This may be due to the utilization of the heuristics in the combination of terms. However, we will show in our experiments that when HAL relations are used to expand queries, we obtain lower retrieval effectiveness than with simple co-occurrence relations.

5.3.2 Deducing Information Flow Relations

Information flow is a mechanism developed to do information inference. We say that there is an information flow from a set of terms (or information items) $t_1,...,t_k$ to another term t_j if the former entails, or "suggests", to some degree, the latter. This is denoted as $t_1,...,t_k|-t_j$.

The extraction of such information flow relations is not performed only according to statistics. Heuristics are used [14], which can be described briefly as follows:

- The initial HAL space is filtered so that for each term, only strong co-occurring terms are kept as "*quality properties*" (QP) of the term;
- The degree of information flow $t_1,...,t_k|-t_j$ is defined as follows:

$$\text{degree } (t_1,..., t_k \mid -t_j) = \frac{\sum_{t \in (QP(t_1...t_k) \cap QP(t_j))} w(t_j \mid t_1,..., t_k)}{\sum_{t \in QP(t_1...t_k)} w(t_j \mid t_1,..., t_k)}$$

which is indeed the proportion of the overlapping quality properties between the premise $t_1,...,t_k$ and the consequence t_j, compared to those of the condition.

For the same query "*space program*", we have the following terms with an IF relation to it:

space program |-

program:1.00 space:1.00 ***nasa****:0.97 new:0.97 US:0.96 agency:0.95* ***shuttle****:0.95 nation:0.95* ***soviet****:0.95 president:0.94 bush:0.94 million:0.94* ***launch****:0.93 call:0.93 thursday:0.93 research:0.92 administration:0.92* ***flight****:0.92* ***rocket****:0.92* ***defense****:0.91 Friday:0.91 project:0.91 system:0.91* ***mission****:0.91 work:0.90 official:0.90* ***station****:0.89 long:0.88 announce:0.88 science:0.88 schedule:0.87 reagan:0.87 direct:0.87 air:0.87 put:0.87 center:0.87 billion:0.87 aeronautic:0.87* ***satellite****:0.87 force:0.86 new:0.86 wednesday:0.86 technology:0.86 america:0.86 budget:0.86 state:0.86 back:0.85 office:0.85 monday:0.85 plan:0.85 people:0.85* ***man****:0.85 ...*

The underlined words are the new ones compared to the HAL relations. We can see that some terms such as "*satellite*" are absent from the HAL vector of "*space program*", but appear in the above IF vector.

Finally, we can define the normalized IF degree as the probability of a term in a query:

$$P_{IF}(t_i \mid \theta_Q^K) = \frac{\text{degree}(Q \mid -t_i)}{\sum_{t_k \in V} \text{degree}(Q \mid -t_k)}$$

This language model can be integrated into the score function as before.

5.4 Discussions

In this chapter, we described several ways to extract term relations. The simple co-occurrence relation between single terms is the baseline model that we will compare to. The second-order information flow relations are developed in [14], which have produced very good retrieval results. The biterm relation is our new type of relation that has not been explored in previous studies. From the implementation point of view, it is only a straightforward extension from the simple co-occurrence relation. However, the more fundamental idea of including more context words into the relation which has not been used before. So our approach will suggest a new research direction to perform context-dependent query expansion. We will show in our experiments that this method is more effective than those proposed previously.

In our approach, the language models we use are unigram models. This type of model has the advantage to be simple and efficient for use. However, it may not always model the documents, a domain, etc. correctly. Therefore, as a future work, it would be interesting to extend it to a model in which dependencies among terms are considered.

For term relations, it would be interesting to infer set of terms instead of single terms. For example, a relation such as "{*information, retrieval*} → {*search, engine*}" is a much more precise relation than "{*information, retrieval*} → *search*" and "{*information, retrieval*} → *engine*". So the idea of grouping terms in the condition part could also be extended to the consequence part. However, more efficient method than the general association rule mining [44] should be developed for this.

Chapter 6

Implementing Domain Context

In this chapter, we will describe our approach to model and to use topic domains of the query.

6.1 Domain-Dependent IR

Let us first describe how the *domain* model $P(t \mid \theta_Q^{Dom})$ is derived. The general query model θ_Q can exploit domains as follows:

$$P(t \mid \theta_Q) = \sum_{Dom} P(t, \theta_{Dom} \mid \theta_Q)$$
$$\approx \sum_{Dom} P(t \mid \theta_{Dom}) P(\theta_{Dom} \mid \theta_Q)$$

In case that we consider several possible domains for a query, we need to make the summation. In this study, we assume that a query only belongs to one domain. Therefore, the selected domain model is assigned the whole probability, i.e., $P(\theta_{Dom} \mid \theta_Q) = 1$ and all the other domains $P(\theta_{Dom} \mid \theta_Q) = 0$.

Let us denote the selected domain model as θ_Q^{Dom}. Then the above part of the general query model becomes:

$$P(t \mid \theta_Q) \approx P(t \mid \theta_Q^{Dom})$$

where $\theta_Q^{Dom} = \arg\max_{\theta_{Dom}} P(\theta_{Dom} \mid \theta_Q)$

The score function according to the domain model is as follows:

$$Score_{Dom}(Q,D) = \sum_{t \in V} P(t \mid \theta_Q^{Dom}) \log P(t \mid \theta_D)$$

$$\approx \sum_{t \in E} P(t \mid \theta_Q^{Dom}) \log P(t \mid \theta_D)$$

In order to use the domain models as above, we have to deal with two problems:

- How to construct a model for each domain?
- How to select the corresponding domain for a query?

6.2 Constructing Domain Models

Recall that we consider a domain model to be a probability distribution over terms, i.e., a language model. We mentioned that it would also be possible to define knowledge (term relations) within a domain. However, we will not exploit this approach, although some tests will be performed in Chapter 8.

The language model for a domain is constructed according to a set of documents included in the domain. For this, we assume that each domain contains a set of documents classified in it. These documents can be identified in two different ways: (1) *using existing domains*; (2) *define one's own domains*.

6.2.1 Using Existing Domains

One can take advantages of an existing domain hierarchy such as *ODP* and *Yahoo! Directory*, and the documents manually classified in them can be used to build domains models.

The *Open Directory Project*[10] maintains a domain hierarchy, and each domain contains a set of manually identified Web pages. Figure 7 shows the top categories defined in *ODP*.

Each of the categories is further separated into sub-categories. At the lowest level, we can find a set of manually classified Web pages. Figure 8 shows a fragment of such a lowest-level category with some of the Web pages included.

Arts Movies, Television, Music...	**Business** Jobs, Real Estate, Investing...	**Computers** Internet, Software, Hardware...
Games Video Games, RPGs, Gambling...	**Health** Fitness, Medicine, Alternative...	**Home** Family, Consumers, Cooking...
Kids and Teens Arts, School Time, Teen Life...	**News** Media, Newspapers, Weather...	**Recreation** Travel, Food, Outdoors, Humor...
Reference Maps, Education, Libraries...	**Regional** US, Canada, UK, Europe...	**Science** Biology, Psychology, Physics...
Shopping Autos, Clothing, Gifts...	**Society** People, Religion, Issues...	**Sports** Baseball, Soccer, Basketball...
World Deutsch, Español, Français, Italiano, Japanese, Nederlands, Polska, Dansk, Svenska...		

Figure 7. Top ODP directories

[10] http://dmoz.org

Top: Science: Environment: Air Quality: Acid Deposition *(13)*

- Acid Rain - Facts, news, children's resources, and links from Environment Canada.
- Acid Rain News - Features links to news articles and related sites. Includes history, scientific background information and FAQ.
- The Acid Rain Report - Student-created site discusses the causes, effects, geographic distribution, and possible solutions for acid rain.
- Acid Rain--A Contemporary World Problem - This website explores the causes and solutions to the acid rain problem.
- Margot's Acid Rain SEA Project - Student's fun and informative site about acid rain, in North Carolina and in general.
- National Atmospheric Deposition Program (NADP) - US federal-state-NGO cooperative effort operating a national precipitation monitoring network to observe geographic and temporal trends in acidity, mercury, and other attributes. Includes data, maps, and meeting announcements.
- PPRP Atmospheric Deposition Measurement and Analysis - Information about and from regional acid deposition monitoring programs in the Chesapeake Bay watershed (USA).
- UK Acid Waters Monitoring Network - Monitors the ecological impact of acid deposition in areas of the United Kingdom believed to be sensitive to acidification.
- ...

Figure 8. Some Web pages in an ODP directory

With such an existing domain hierarchy, one can create a set of domain models by exploiting the Web pages classified in each domain.

6.2.2 Defining One's Own Domains

It is possible that no appropriate domain hierarchy is defined for an application, and the user has to define his own categories. This situation happens when the general categories are insufficient for a user. For example, a user may work in a very specific area and need more fine-grained categories, or the existing categories do not correspond to the user's conception and the user wants to define the categories in his own way. In some cases, the user's own domains can be manually mapped to an existing domain hierarchy. This is the approach used in [12][23][108]. However, this approach is not always feasible.

An alternative way is to ask the user to assign a domain *Dom*(Q) to his queries for a period of time in order to collect example documents for each domain. In this case, we can collect example documents in several ways:

- By user's manual judgments: the user can judge the relevance of the documents to a query. The relevant documents can be classified into the domain *Dom*(Q). This is equivalent to a manual classification. However, the approach is difficult to implement as most users consider the judgment of relevance a burden and are not willing to do it;
- By observing the user's interactions with the system: as an alternative, we can collect the documents that the user chooses to read or browse through for a query and put them into the domain *Dom*(Q). This strategy is similar to those that exploit user logs to guess the intent of the user [24];
- Using top-ranked retrieval results: one can also assume that the top-ranked documents are closely related to the query, thus should be classified into the domain *Dom*(Q). This assumption is equivalent to the pseudo relevance feedback.

Using each of the above approaches, we will be able to collect example documents in each domain. In our experiments, we will compare the first and the third approaches. We will not test the second approach because our test data (from TREC) do not contain user browsing information.

We can now assume that a set of documents is available for each domain. We will describe how to construct a domain LM from them.

The simplest approach that we can imagine is to use MLE to estimate this domain model. The more often a term appears in the documents in a domain, the higher is its probability. However, we also have to notice that the documents in a domain do not only contain domain-specific terms. General terms in a language also occur frequently. Therefore, by MLE, both domain-specific and general terms will be mixed

up. What we desire, however, is a domain model that focuses on domain-specific terms. It is then necessary to purify the domain model so that common terms can be filtered out.

To do this, we employ *expectation maximization* (EM) algorithm to extract the specific part of the domain. This process is the same as that used in [115]. In this process, we assume that each document in the domain is the result of generation from both a domain-specific model (to be extracted) and the general language model (approximated by the collection model). The goal of the EM process is to extract the domain model such that the likelihood of the domain documents can be maximized. This is a classical application of EM.

More specifically, the likelihood of a domain document D is expressed as the following generation from a mixture of the domain model and the collection model:

$$P(D \mid \theta'_{Dom}) = \prod_{t \in D} [\lambda_{Dom} P(t \mid \theta_{Dom}) + (1 - \lambda_{Dom}) P(t \mid \theta_C)]^{tf(t,D)}$$

where $tf(t,D)$ is the term frequency of t in document D, and λ_{Dom} is a smoothing parameter. We fix the parameter λ_{Dom} at 0.5 as in [115], where the same process is used to extract a feedback model.

The EM algorithm is used to extract the domain model θ_{Dom} that maximizes $P(Dom \mid \theta'_{Dom})$ (where Dom is the set of documents in the domain), that is:

$$\begin{aligned} \theta_{Dom} &= \arg\max_{\theta_{Dom}} P(Dom \mid \theta'_{Dom}) \\ &= \arg\max_{\theta_{Dom}} \prod_{D \in Dom} \prod_{t \in D} [\lambda_{Dom} P(t \mid \theta_{Dom}) + (1 - \lambda_{Dom}) P(t \mid \theta_C)]^{tf(t,D)} \end{aligned}$$

During the EM process, the updating functions are as follows:

$$w^{(n)}(t) = \frac{\lambda_{Dom} P^{(n)}(t \mid \theta_{Dom})}{[\lambda_{Dom} P^{(n)}(t \mid \theta_{Dom}) + (1 - \lambda_{Dom}) P(t \mid \theta_C)]} \qquad \text{(E-step)}$$

$$P^{(n+1)}(t \mid \theta_{Dom}) = \frac{\sum_{D \in Dom} tf(t, D) w^{(n)}(t)}{\sum_{D \in Dom} \sum_{t_i \in D} tf(t_j, D) w^{(n)}(t_i)} \qquad \text{(M-step)}$$

where the superscript (n) means the value at step n. To start, we can assign $P^{(0)}(t \mid \theta_{Dom})$ in different ways. In our case, we use MLE as the initial probability value.

One may question about the fixed value for the parameter λ_{Dom}. If we allow the parameter to vary during the EM process, the parameter would tend to become 1. This is because one would obtain MLE when $\lambda_{Dom} = 1$, and this would maximize the likelihood of the documents in the domain. By fixing a value to this parameter, we force a part of the MLE model corresponding to the collection model to be removed. This allows us to purify the domain model.

The effect of the EM process can be observed in the following table, which shows some words in the domain model of "*Environment*" before and after EM iterations (the model converges after 12 iterations):

Table 5. Term probabilities before/after EM

Term	Initial	Final	change	Term	Initial	Final	change
air	0.00358	0.00558	+ 56%	year	0.00357	0.00052	- 86%
environment	0.00213	0.00340	+ 60%	system	0.00212	$7.13*e^{-6}$	- 99%
rain	0.00197	0.00336	+ 71%	program	0.00189	0.00040	- 79%
pollution	0.00177	0.00301	+ 70%	million	0.00131	$5.80*e^{-6}$	- 99%
storm	0.00176	0.00302	+ 72%	make	0.00108	$5.79*e^{-5}$	- 95%
flood	0.00164	0.00281	+ 71%	company	0.00099	$8.52*e^{-8}$	- 99%
tornado	0.00072	0.00125	+ 74%	president	0.00077	$2.71*e^{-6}$	- 99%
greenhouse	0.00034	0.00058	+ 72%	month	0.00073	$3.88*e^{-5}$	- 95%

We can see in the left part that the probabilities of domain-specific terms are much increased, while those of general terms in the right part are largely reduced. The domain model can thus be assumed to reflect the specific terms used in the domain.

6.3 Determining Query Domain

Once a set of domain models has been built, the next question is to assign the appropriate domain to a query. There are two possibilities:

- The user can manually assign a domain to each of his queries. This approach is not always realistic. Indeed, this assignment may be a burden to the user;
- An alternative way is to determine the query domain automatically. This is a problem of query classification. We will describe this approach in more details.

The problem of query classification is defined as follows: we have a set of predefined categories, domains or classes. Each of them contains a set of documents already classified in it. The task of query classification is to assign an appropriate domain to a query. This task is similar to automatic text classification, at the difference that a query is much shorter. It is expectable that short queries cannot be classified as accurately as longer texts. Query classification has been investigated in several studies. For example, Shen et al. [89] investigated the classification of Web queries, which are often short and contain unknown words. In their approach, many Web resources and heuristics have been employed to determine the class of queries. In our study, as we target queries in TREC style, we do not encounter the same problems. Therefore, we will use an approach closer to text classification.

6.3.1 Some Text Classification Approaches

In text classification, several approaches exist. We will describe briefly some of them which have been widely used:

- ***K*-Nearest Neighbor (KNN) Classifier:**

  K-Nearest Neighbor is a well-known non-parametric instance-based learning algorithm. It is based on the hypothesis that the characteristics of members of the same class are similar; therefore, the points (i.e., the documents) that locate closely in the vector space should belong to the same class. Given a new document, its k nearest neighbors are determined according to their similarity to the document (see section 2.3.2 for the formulas of similarity calculation). Each of these neighbors votes for the class of the document. The class with the highest votes is assigned to the document. The following figure depicts a simple example of this method.

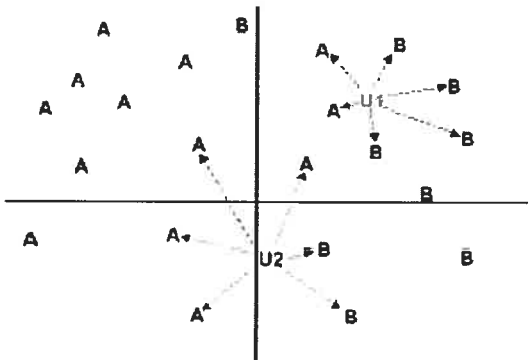

Figure 9. K-Nearest Neighbor illustration

In this figure, arrows represent the distance from the new document to their nearest neighbors, and each neighbor is given equal weight. When k is set to 6, U_1 and U_2 are classified as members of groups B and A respectively, because the majority of their neighbors belong to these classes.

- **Support Vector Machine (SVM):**

The basic classification problem is two-class classification. For this, SVM tries to determine a hyperplane that can maximally separate the instances of the two classes.

Assume we are given a set of training documents $x_i \in R^n$ with $i = 1,...,n$ denoting different dimension of the space. Each document x_i belongs to one of the two classes and it is given a label $y_i \in \{-1, +1\}$. The goal of SVM is to establish the equation of a hyperplane that divides the points leaving all the points of the same class on the same side, while maximizing the distance between the two classes and the hyperplane, i.e., the margin. The support vectors are those vectors (documents) that determine the margin. Informally, they are the hardest data to classify, and the most informative ones for designing the classifier. Here we will only discuss linear SVM.

The equation of the separating hyperplane can be written as $w^T \cdot x+b=0$, where x is an arbitrary data point to be classified, vector w (the weights of the features) and constant b are learned from a training set of linearly separable data.

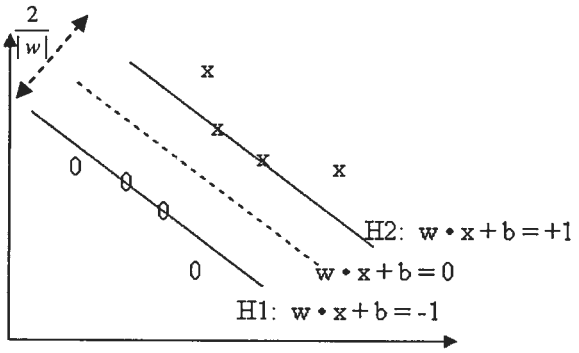

Figure 10. Separating hyperplane and margin

In Figure 10, H_1 and H_2 are the furthest separating hyperplane parallel to the desired separating hyperplane. They are called the margin planes. The separating hyperplane locates in the middle of H_1 and H_2. So margin planes can be considered as $w^T \cdot x+b=-1$ and $w^T \cdot x+b=+1$. The margin between them is $\frac{2}{|w|}$.

The SVM problem is to find w and b that maximizes $\frac{2}{|w|}$, or minimizes the following cost function:

$$\underset{w}{\text{minimize}} \ \frac{1}{2} w^T w :$$

$$\text{Subject to } w^T \cdot x_i + b \geq +1 \quad for\ y_i = +1$$

$$w^T \cdot x_i + b \leq -1 \quad for\ y_i = -1$$

This is a quadratic programming optimization problem. We will not go into details about it.

Once the separating hyperplane is found, the problem of classifying a new data point x is now solved simply by checking which side of the hyperplane it falls on.

SVM has proven to be an effective classification method in many experiments. However, it has higher requirements for time and space than other approaches, in particular Naïve Bayes classifier.

- **Naïve Bayes (NB) Classifier:**

Naïve Bayes classifier is one of the most widely used methods in text classification due to of its simplicity and efficiency.

Given a document D and a set of predefined classes $\{...c_i,...\}$, an NB classifier computes the posterior probability that the document belongs to each particular class c_i, i.e., $P(c_i \mid D)$, and assigns the document to the class with the highest probability value. The posterior probability is computed by applying the Bayes rule:

$$P(c_i \mid D) = \frac{P(D \mid c_i)P(c_i)}{P(D)}$$

The denominator $P(D)$ in above formula is independent from classes; therefore, it can be ignored for the purpose of class ranking. Thus:

$$P(c_i \mid D) \propto P(D \mid c_i)P(c_i)$$

In Naïve Bayes, it is further assumed that words are independent given a class, i.e., for a document $D = t_1,...,t_m$:

$$P(D \mid c_i) = \prod_{j=1}^{m} P(t_j \mid c_i)$$

We then have:

$$P(c_i \mid D) \propto \prod_{j=1}^{m} P(t_j \mid c_i)P(c_i)$$

The class probability $P(c_i)$ can be estimated by the percentage of the training examples belonging to class c_i:

$$P(c_i) = \frac{N_i}{N}$$

where N_i is the number of training documents in class c_i, and N is the total number of training documents respectively. $P(t_j \mid c_i)$ is usually determined by:

$$P(t_j \mid c_i) = \frac{1 + tf(t_j, c_i)}{|V| + |c_i|}$$

where $tf(t_j, c_i)$ is the term frequency of the term t_j within the training documents of class c_i, $|V|$ is the total number of vocabulary, and $|c_i|$ is the total number of words in class c_i. This estimation uses the *Laplace* (or *add-one*) smoothing to solve the zero-probability problem.

Once the class model $P(t_j \mid c_i)$ and class probability $P(c_i)$ are estimated, the classification of a new document is very efficient.

6.3.2 Query Classification Using Language Models

The Naïve Bayes classification is very similar to language modeling approach. In fact, the probability $P(t_j \mid c_i)$ defines a unigram language model for class c_i, and $P(D \mid c_i)$ is the likelihood of the document to be classified into this language model.

The only minor difference is that *Laplace* smoothing is often used in Naïve Bayes classification, while more sophisticated smoothing methods can be used in language models.

It is then intuitive to extend the Naïve Bayes classifier by using a language modeling approach. This extension has been considered by Peng et al. [67]. In our previous studies [2][3], we have also investigated the utilization of LM for classification, and we have attempted to integrate compound terms in it as well. The experiments showed that the classification accuracy can be improved by replacing the *Laplace* smoothing with other smoothing methods, such as *Dirichlet*.

Here we use a similar approach for the following reasons:

- The class language model is easy to build and to update. This is important for us because new documents will be added along with the

utilization by the user. An approach such as SVM would be more difficult to be updated;

- Compared to KNN, the classification process of this approach is faster. This is because we do not have to calculate a similarity of the query with each of the example documents, but only have to compare with a small number of class models;
- Finally, in previous studies, NB and language models have shown good classification results.

Given a set of domain models, we select the closest one with which the KL-divergence score of the query is the lowest, i.e.:

$$\theta_Q^{Dom} = \arg\max_{\theta_{Dom}} \sum_{t \in Q} P(t \mid \theta_Q^0) \log P(t \mid \theta_{Dom})$$

As we showed earlier, this is equivalent to choose the class model, from which the query can be best generated. This approach has an additional advantage that we do not need to construct additional model for query classification. The domain models used here will be used later in our retrieval process.

6.4 Sub-Domain Models

Although domain models are more refined than a single user profile, the topics in a single domain can still be very different. For example, some domains cover a large variety of topics, such as *Science and technology* (a domain defined for TREC queries). Using such a large domain model as the background can also introduce noise terms.

In order to create a domain model that is more related to the query, we can construct a sub-domain model as follows: the query is used to retrieve a subset of documents within the domain. These documents are closely related to the query's topic. Then we construct a sub-domain with these documents.

This approach is indeed a combination of domain and feedback models. In principle, one can expect higher effectiveness with such a sub-domain model than with the entire domain model.

In our experiments, we will see that this further specification of sub-domain is useful in some, but not all cases.

Chapter 7
Parameter Tuning

There are several parameters in our model: λ for smoothing the document model with the collection model, and the mixture weights $\alpha_i (i \in \{0, K, Dom, FB\})$ for combining different component query models. As our work is mainly concerned with query model, in our study, we will use a fixed manner to smooth document models. Therefore, we determined a value ($\lambda = 0.5$) that maximizes the effectiveness on a training collection: TREC queries 1-50 and documents on Disk 2, and use the same value for testing data throughout our experiments.

In this chapter, we will describe the methods to tune the mixture weights α_i of the component query models. We will propose two methods: (1) using a training dataset containing documents, queries and relevance judgments; (2) using an unsupervised training without relevance judgments.

7.1 Tuning with Relevance Judgments

Using a training dataset with relevance judgments, we can directly try to determine the best parameters that maximize the final objective function, which is the *mean average precision* (MAP) (see Section 2.4.2) on the training dataset.

Assume a set of parameters $\overline{\alpha} = \langle \alpha_0, \alpha_K, \alpha_{Dom}, \alpha_{FB} \rangle$, the query model is described as follows:

$$P(t \mid \theta_Q, \overline{\alpha}) = \sum_{i \in X} \alpha_i P(t \mid \theta_Q^i)$$

where $X = \{0, K, Dom, FB\}$. Then the document score is determined as:

$$Score(Q, D, \overline{\alpha}) = \sum_{t \in V} \sum_{i \in X} \alpha_i P(t \mid \theta_Q^i) \log P(t \mid \theta_D)$$

$$= \sum_{i \in X} \alpha_i Score_i(Q, D)$$

The average precision (AvgP) for a query Q can be defined as a function of the parameters $\overline{\alpha}$ as follows:

$$AvgP(Q, \overline{\alpha}) = \frac{1}{|R_Q|} \sum_{D_j \in R_Q} \frac{j}{Rank(D_j, Q, \overline{\alpha})}$$

where $Rank(D_j, Q, \overline{\alpha})$ is the rank of the j-th relevant document for the query Q. Then the MAP for a set of N test queries is:

$$MAP(\overline{\alpha}) = \frac{1}{N} \sum_{i=1}^{N} AvgP(Q_i, \overline{\alpha})$$

Finally, the parameter tuning problem is defined as follows:

$$\overline{\alpha} = \arg\max_{\overline{\alpha}} MAP(\overline{\alpha})$$

This maximization problem cannot be solved using the methods such as *gradient descent* because the objective function $MAP(\overline{\alpha})$ is not smooth. One suitable method is *line search* (or *coordinate descent*). This method has been used previously in machine translation [63] and IR [33], and it produced very good results. We follow this approach here.

The MAP function can be viewed as a multi-dimensional function. For the latter, line search is a common method for its maximization [72]. The basic idea of line search is to try to maximize each of the parameters in turn, while keeping the other parameters unchanged in each iteration, until reaching a maximum.

Suppose a set of free parameters. Line search works as follows:

(1) A grid is defined by a set of values for each free parameter;

(2) A start point is chosen at random on the grid;

(3) Each parameter is selected in turn. This parameter takes on all the possible values, while the other parameters remain unchanged;

(4) The value of the parameter which gives the best value to the objective function is chosen;

(5) The steps (3) and (4) are repeated on each parameter in turn until no change is made on the values.

An illustration is shown in Figure 11, where we assume two free parameters varying from 0 to 1. The objective function corresponds to the vertical axis. The path indicated in the figure corresponds to the changes of the two parameters during 2 iteration loops.

In our case, we have four parameters which should satisfy the constraint that their summation equals to 1. Therefore, we vary three parameters while the fourth is set to the complement of the others. For each of the three parameters, the possible values that it can choose in step (4) of the algorithm should also satisfy the constraint that $\alpha_i \leq$ 1 - (sum of the two other parameters).

It is known that this algorithm can be trapped in a local maximum. In order to avoid this situation to some extent, we repeat the line search 10 times, each from a random point. The best values among the 10 runs are kept.

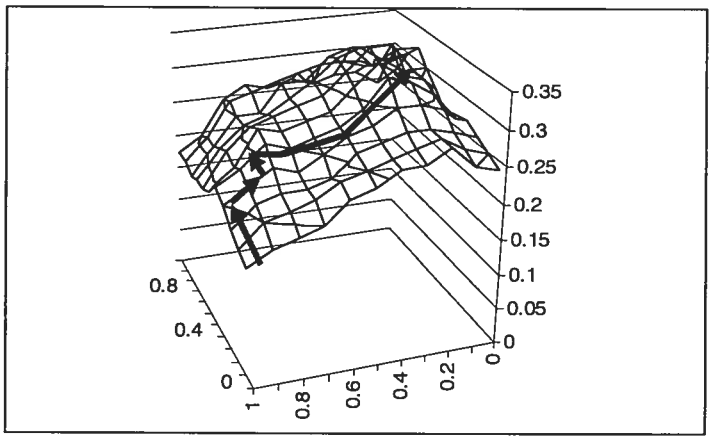

Figure 11. Illustration of line search algorithm

For the purpose of our tests, we define a relatively coarse-grained grid. Each parameter is allowed to vary among the following values: 0, 0.1, 0.2, …, 1.0.

With such a limited number of values, it would be possible to perform an exhaustive search on the whole grid, which contains less than 11^3 nodes. However, when the number of values grows, the complexity also grows quickly. Using line search, we can refine the values and the search complexity only grows linearly to the number of values in each direction.

The above training process determines the values of the parameters on a training dataset. Then we assume that the same parameters can be used on a different dataset. In our experiments, we will see that the values we obtain in this process are reasonable for a new dataset. These values are often close to the optimal values. So the parameters seem to be quite stable across document collections and queries.

7.2 Tuning without Relevance Judgments

In many cases, we do not have relevance judgments. The previous line search algorithm (or any supervised learning method) cannot be used to train the parameters.

In this situation, it is a common practice in IR to use a set of feedback documents as "relevant" documents. Therefore, a possible way is to determine the parameters $\overline{\alpha}$ such that they maximize the likelihood of these feedback documents.

This principle has been widely applied in IR, especially in the language modeling framework, for determining parameters. For example, the mixture model used by Zhai and Lafferty [115], the relevance model by Lavrenko and Croft [52] all use this principle. We follow the same principle here.

Given a set of feedback documents *FB*, its likelihood according to the query model is defined as follows:

$$L(FB) = \prod_{D \in FB} P(D \mid \theta_Q)$$
$$= \prod_{D \in FB} \prod_{t \in V} \sum_{i \in X} \alpha_i P(t \mid \theta_Q^i)^{tf(t,D)}$$

where $tf(t,D)$ is the term frequency of t in document D, and θ_Q^i is a component query model.

However, the above likelihood cannot be maximized because of the zero-probability problem. Indeed, in the query model, many terms will have zero probability, even when it has been extended. To solve this problem, we assume again that the feedback documents are generated from two sources: the query model and a general language model, which is approximated by the collection model. Then the log-likelihood of the feedback documents is as follows:

$$LL(FB) = \sum_{D \in FB} \sum_{t \in V} tf(t,D)[\sum_{i \in X} \alpha_i P(t \mid \theta_Q^i) + \alpha_C P(t \mid \theta_C)]$$
$$= \sum_{D \in FB} \sum_{t \in V} tf(t,D) \sum_{i \in X'} \alpha_i P(t \mid \theta_Q^i)$$

where $X' = X \cup \{C\} = \{0, K, Dom, FB, C\}$ and $\sum_{i \in X'} \alpha_i = 1$.

The determination of the best mixture weights to maximize *LL(FB)* is a classical problem, which can be solved using EM algorithm. Here we will leave the derivation of the updating functions in Appendix 1. These update functions derived are as follows:

$$Z_{i,t} = \frac{\alpha_i' P_i(t \mid \theta_Q^i)}{\sum_{i \in X'} \alpha_i' P_i(t \mid \theta_Q^i)} \qquad \text{(E-step)}$$

$$\alpha_i = \frac{\sum_{D \in FB} \sum_{t \in V} tf(t,D) Z_{i,t}}{\sum_{D \in FB} \sum_{t \in V} tf(t,D)} \qquad \text{(M-step)}$$

where α_i' is the previous parameter value, and α_i is the new parameter value. To start the EM process, we set all the parameters at equal value, i.e., 0.2.

The above EM process may still have some problems. In fact, we allowed the collection model to play different roles in the above training process. Its mixture weight can vary largely. The component query models will share the remaining part. However, in the score function, we only use the component query models, but not the collection model, for the query. Therefore, the relative importance of the component query models resulted in this training process may not be always reasonable. In order to solve this problem, we can fix the mixture weight of the collection model at 0.5, and let those of the component query models vary, so that they will always share a total mixture weight 0.5.

This EM process is performed for each query. The training process is done online (during the retrieval time). Therefore, time is critical. Fortunately, the EM

process converges very quickly. On average, it stops after about 12 iterations (the maximum number of iteration is set at 50).

We also note that the number of feedback documents is limited: we use the top 20 retrieved documents. So the calculation of the likelihood is also very fast. In all, the EM process takes about 5 seconds.

In both training processes, we assign one mixture weight to a component query model. One may question if a more refined weighting depending on the term should be made. For example, a component model may not cover all the topics (terms) equally well. It can cover some topics better than some others. Therefore, it may be reasonable to assign a mixture weight according to the term that we are considering. A similar approach has been tested in another study [16]. We will leave the utilization of this approach to a future research.

Chapter 8
Experiments

In this chapter, we will test the approaches we proposed earlier on several TREC *ad hoc* retrieval collections. We will integrate several contextual factors by using a language modeling approach. The experimental results show that it is feasible and effective to use contextual factors in IR systems.

8.1 Experimental Questions

Our experiments aim to investigate several aspects concerning the utilization of contextual factors:

(1) For *inter-query context – knowledge* model, in addition to testing its effectiveness, we also want to compare the context-dependent relations with context-independent relations and the second-order (i.e., information flow) relations. This will show whether the consideration of *intra-query context* is important.

(2) For *extra-query contexts – domain* model and *feedback* model, we examine the following questions:

- Is it useful to use domain model and feedback model to complete the query?
- When both models are incorporated, are the effects of them complementary?
- We described two ways to gather documents for a domain: either by using documents manually classified, or by using documents retrieved for the queries in the domain without relevance judgments. How do they compare?
- There are two ways to determine the domain for a query: manually by the user or automatically by the system. How do they compare?

(3) Finally, we will see the results of the complete model when all the contextual factors are combined.

8.2 First Experiments: Comparing Different Types of Term Relations for Query Expansion

This first series of experiments aim to compare query expansion with context-dependent relations and with context-independent relations.

8.2.1 Test Collections

The experiments are carried out on three documents collections: AP, SJM and WSJ, which are used in TREC Disks 1-3. The queries we use are also in TREC Disks 1-3 [40][41]. We choose to use these queries because each topic (query) has manually specified domain. This will allow us to carry out experiments on domain models later.

The statistics of the test collections are given in Table 6:

Table 6. Collection statistics of first experiments

Collection	Description	Size (MB)	Vocabulary	# of Doc.	Query
AP	*Associated Press* (1988-98)	491	196,933	164,597	51-100
SJM	*San Jose Mercury News* (1991)	286	146,514	90,257	101-150
WSJ	*Wall Street Journal* (1990-92)	242	121,946	74,520	51-100

Below is an example of full TREC topic (#55):

<top>

<head> Tipster Topic Description

<num> Number: 055

<dom> Domain: ***International Economics***

<title> Topic: ***Insider Trading***

<desc> Description:

Document discusses an insider-trading case.

<smry> Summary:

Document discusses an insider-trading case.

<narr> Narrative:

A relevant document will discuss an insider-trading case, identifying the accused/the defendant(s), as well as the government doing the investigation. It will also mention at least one of the following specifics of the case: the alleged illegal activity, whether charged with providing or using insider information, possible conspirator's role in the scheme, the monetary amount of illegal profit and of damage; if pronounced guilty, then the sentence terms, e.g., monetary penalty (cash payment), time in prison, cooperation with the government, probation, community service, being barred from the industry for a certain time period.

<con> Concept(s):

1. *insider-trading case*
2. *insider-trading scheme, illegal insider-trading activity, illegal securities transactions, white-collar crime, securities-law violations*
3. *insider-trading investigation, insider-trading probe*
4. *Securities and Exchange Commission, SEC*
5. *investment banking, investment banker, arbitrage, arbitrager, securities analyst*
6. *inside information, advance knowledge of corporate takeovers*
7. *leaking information, misappropriating information, leaking word, leaking news, passing a tip*
8. *pre-bid market activity, takeover speculation, matched-book transaction*

<fac> Factor(s):

<def> Definition(s):

Insider Trading - Dealing in shares with the advantage of inside information. The owners of shares in a public company are all supposed to be equal - so if one of them knows that the company is about to go broke before the others and sells his shares while their price is still good, that is unfair. In many countries, including the U.S. and the U.K., it is also illegal. However, it is very difficult to prove that an investor is buying or selling shares on the basis of inside information.

</top>

We choose to use only the "*<title>*" of topic as our query because this is a more realistic situation, although using the other fields of the topics may result in better retrieval effectiveness.

Our approaches described in the previous chapters have been integrated with the *Lemur* toolkit[11], which is developed jointly by the Language Technologies Institute of

[11] http://www.lemurproject.org/

Carnegie-Mellon University and the Center of Intelligent Information Retrieval (CIIR) in University of Massachusetts at Amherst. This is a toolkit for IR based on language modeling approach. The basic retrieval models as well as various smoothing techniques have been integrated in *Lemur*. We have extended this toolkit to build different component query models and to integrate them.

In these experiments, as well as all the following experiments, we perform the standard preprocessing as follows:

- Terms are stemmed using the *Porter* stemmer [70];
- Stopwords are removed. The stoplist is the one included in the *Lemur* Toolkit.

In this series of tests, the query model only combines the basic unigram language model with different *knowledge* models, which are defined below.

8.2.2 Comparing Context-Dependent and Context-Independent Relations

In this series of experiments, we compare the impact of using different types of term relations for query expansion. The following types of term relations are compared:

- Context-independent - traditional co-occurrence relations of the form $t_j \rightarrow t_i$;
- Context-dependent - biterm relations of the form $\{t_j, t_k\} \rightarrow t_i$.

In all the experiments, we will use the same documents models, which are smoothed with *Jelinek-Mercer* and *Dirichlet*. These methods have shown good results in other studies [116], and they are proven to be robust. In our experiments, the parameters have been tuned so that we can obtain the best effectiveness for the baseline model. For example, the mixture weight λ in *Jelinek* smoothing is set at 0.5, and the

Dirichlet prior μ in *Dirichlet* smoothing (see Section 2.3.4) is set at 1000, following the experimental results of Zhai and Lafferty [116]. This baseline method is the one that produces the state-of-the-art retrieval effectiveness which is comparable or higher than reported in other studies on the same collections.

Below, we will remind the query models that we compare:

- **Basic LM (UM):**

  In our baseline model, we use the basic unigram language model as our query model, which is estimated by MLE.

- **Context-independent query expansion using co-occurrence relations (CIQE):**

  We extract co-occurrence relations from each document collection. The co-occurrence relations are weighted as described in Section 5.1. Co-occurrences are considered within windows of fixed size. In this experiment, we use a window size of 10 words. This window size has produced good results in a previous study [14]. In fact, when the window size increases, we do not observe much difference, but more relations are extracted. Therefore, the size 10 seems to be a good compromise between the performance and efficiency.

  The co-occurrence relations are extracted from the respective document collection. 80 strongest expansion terms are selected and incorporated into the co-occurrence model. Again, this number of expansion terms has produced good results. Increasing this number does not lead to any meaningful increase in retrieval effectiveness, as we will see later in the experiments.

  The mixture weight α_K is manually tuned on a different training collection: AP89 collection (TREC Disk 1) and queries 1-50. The value is $\alpha_K = 0.6$ (i.e., the mixture weight for the original query model is 0.4). The two other parameters α_{Dom} and α_{FB} in this series of experiments are 0.

- **Context-dependent query expansion (CDQE):**

This expansion uses context-dependent term relations to create another knowledge model (see Section 5.2). These relations only apply to queries of at least two words, and no impact is produced for queries of one single word. As for the previous query expansion experiments, we use a window size of 10 words, and the 80 strongest expansion terms are selected and integrated into the knowledge model. Again, the mixture weight α_K is tuned using the same training collection ($\alpha_K = 0.7$ in this case).

In this series of experiments, we assign a uniform probability $P(t_j t_k \mid Q)$ to any biterm (t_j, t_k) in the query, i.e., any combination (t_j, t_k) of two words in the query is considered as a valid biterm, and it is used to suggest expansion terms. This strategy produces the best effectiveness among all those we tested (see a later test on this aspect).

8.2.3 Experimental Results

The main experimental results are described in Table 7 (with *Jelinek-Mercer* smoothing for documents) and Table 8 (with *Dirichlet* smoothing for documents). **UM**, **CIQE** and **CDQE** use the basic query model and the two expanded query models respectively. In these tables, we also indicate whether the improvement in average precision obtained is statistically significant by *t*-test: */+ indicates that the improvement is significant at the level of $p < 0.05$ and **/++ indicates it is significant at the level of $p < 0.01$; (.) is compared to **UM** and [.] is compared to **CIQE** respectively.

Table 7. Query expansion by CIQE vs. CDQE (*Jelinek-Mercer*)

Collection	Measure	UM	CIQE (Co-occurrence)	CDQE (Biterm relation)
AP Q51-100	AvgP	0.2524	0.2844 (+12.68%)**	**0.3325 (+31.74%)** [+16.91%]++
	Recall / 6 101	3 535	3 862	**3 991**
	P@10	0.3878	0.4102	**0.4837**
SJM Q101-150	AvgP	0.1781	0.2122 (+19.15%)**	**0.2435 (+36.72%)** [++14.75%]++
	Recall / 2 559	1 579	1 761	**1 863**
	P@10	0.3087	0.3391	**0.3935**
WSJ Q51-100	AvgP	0.2332	0.2404 (+3.09%)	**0.2714 (+16.38%)** [+12.90%]+
	Recall / 2 172	1 554	1 653	**1 740**
	P@10	0.3104	0.3125	**0.3438**

Table 8. Query expansion by CIQE vs. CDQE (*Dirichlet*)

Collection	Measure	UM	CIQE (Co-occurrence)	CDQE (Biterm relation)
AP Q51-100	AvgP	0.2767	0.2902 (+4.88%)*	**0.3383 (+22.26%)** [+16.57%]++
	Recall / 6 101	3 677	3 897	**4 029**
	P@10	0.4408	0.4551	**0.5082**
SJM Q101-150	AvgP	0.2017	0.2225 (+10.31%)**	**0.2448 (+21.37%)** [+10.02%]+
	Recall / 2 559	1 641	1 761	**1 873**
	P@10	0.3152	0.3630	**0.3870**
WSJ Q51-100	AvgP	0.2373	0.2393 (+0.84%)	**0.2710 (+14.20%)** [+13.25%]+
	Recall / 2 172	1 588	1 626	**1 737**
	P@10	0.3292	0.3313	**0.3625**

Let us analyze these results in more details:

8.2.3.1 CIQE and CDQE vs. UM

It is interesting to observe that query expansion, either by **CIQE** or **CDQE**, consistently outperforms the basic **UM** on all the collections. In all the cases except **CIQE** for WSJ, the improvements in average precision are statistically significant. At the same time, the increases in the number of relevant documents retrieved are also consistent with those in average precision.

The improvement scales obtained with **CIQE** are relatively small. These improvements correspond to the typical figure using this method that is shown in other studies using language modeling approach.

On the other hand, query expansion with context-dependent relations (**CDQE**) produces much higher improvements. In all the cases, the differences between them are statistically significant. The gain of context-dependent relations compared to the traditional co-occurrence relations stems from the fact that the relations applied to queries are more appropriate by using the additional context word. Indeed, a word that co-occurs with a biterm is more strongly related to the query than a term that co-occurs with only one query term. The results that we obtain strongly confirms our hypothesis that context-dependent term relations can identify better expansion terms than context-independent unigram relations.

We have shown an example earlier to compare the two types of relations. Let us show one more example. The expansion terms (stemmed) for the query #55 "*insider trading*" suggested respectively by **CIQE** and **CDQE** are as follows:

CIQE: *stock:0.0141 market:0.0113 US:0.0112 year:0.0102 exchange:0.0101 trade:0.0092 report:0.0082 price:0.0076 dollar:0.0071 1:0.0069 govern:0.0066 state:0.0065 future:0.0061 million:0.0061 day:0.0060 office:0.0059 people:0.0059 york:0.0057 issue:0.0057 ...*

CDQE: ***secure****:0.0161 charge:0.0158 stock:0.0137* ***scandal****:0.0128* ***boeski****:0.0125 inform:0.0119 street:0.0113 wall:0.0112 case:0.0106 year:0.0090 million:0.0086* ***investigate****:0.0082 exchange:0.0080 govern:0.0077* ***sec****:0.0077* ***drexel****:0.0075* ***fraud****:0.0071 law:0.0063 ivan:0.0060 …*

In the case of **CIQE**, some suggested expansion terms are quite related to the query such as "*exchange*", "*market*", but many others are common words in English such as "*report*", "*future*", "*dollar*", etc. These latter terms do not help retrieve relevant documents. On the contrary, they can even hurt it. Therefore, it is very important to assign an appropriate mixture weight to the **CIQE** model.

For **CDQE,** we observe that most of the suggested words are relevant. What is interesting to observe is that not only the general term related to "*insider trading*" such as "*secure*", "*sec*", "*scandal*", "*fraud*", etc. are suggested, but also specific names: "*boeski*" – the name of a person involved in an insider trading scandal, and "*drexel*" – a company involved in this scandal. This example shows clearly the advantages that we can have with context-dependent term relations.

8.2.3.2 Query Expansion can also Increase Precision

What is interesting to see is that the addition of expansion terms does not only improve recall. Precision of top-ranked documents (*P@10*) is also improved. This can also be seen in Figure 12 and Figure 13, where we compare the full *precision-recall* curves for the AP collection using the three query models. In particular, we can see that at all the recall levels, the precision values always follow the following order: **CDQE** > **UM**. The same observation is also made on the other collections. This shows that the **CDQE** method does not increase recall to the detriment of precision, but both of them. The reason that can explain this is that, by adding terms into the query, the documents that contain these additional terms are favored. If the added terms are strongly associated with relevant documents, then the documents promoted to the top tend to be more relevant. Therefore, adding strongly related terms into the query can also be a

means to improve precision. This result also shows that context-dependent relations are capable to suggest strongly related terms.

In contrast, **CIQE** increases precision at all but 0.0 recall points: the precision at the 0.0 recall point is 0.6699 for **UM**, but 0.6565 for **CIQE** (in *Dirichlet* smoothing). This shows that **CIQE** can slightly deteriorate the top few documents, although the *P@10* is a little bit increased. This figure corresponds more to the general observation and belief using query expansion: it is better suited to improve recall than precision. However, with **CDQE**, precision is also increased.

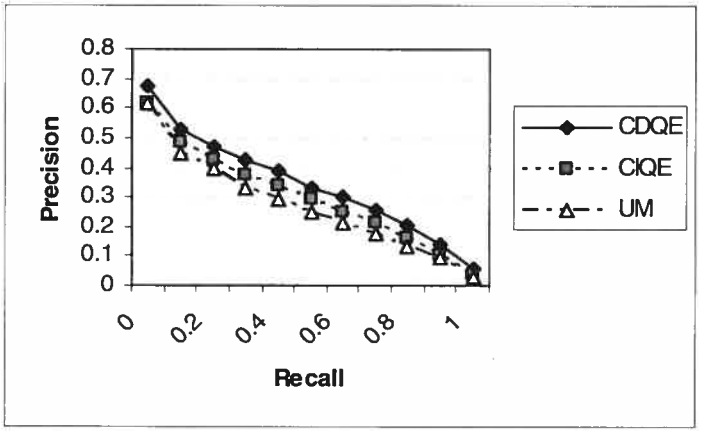

Figure 12. Comparison of query expansion on AP (*Jelinek-Mercer*)

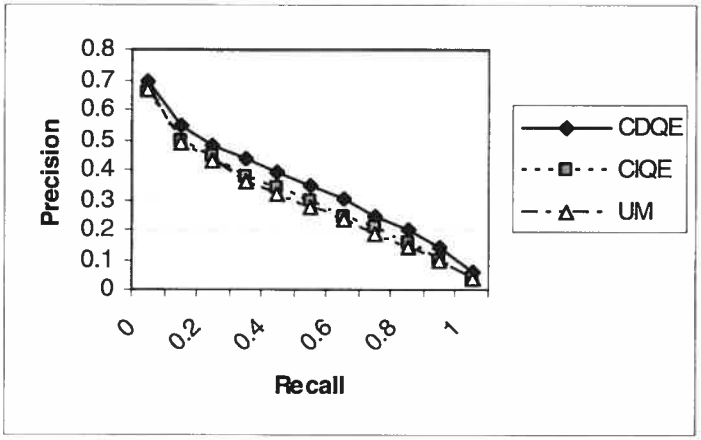

Figure 13. Comparison of query expansion on AP (*Dirichlet*)

8.2.3.3 Effect of Smoothing Parameter

In the previous experiments, we have fixed the smoothing parameters according to a training collection. In this series of tests, we analyze the effect of the smoothing parameters on retrieval effectiveness. The following figures show the change of average precision (AvgP) using **CDQE** along with the change of the parameter α_K (**UM** is equivalent to $\alpha_K = 0$, and the previous results are obtained with $\alpha_K = 0.7$).

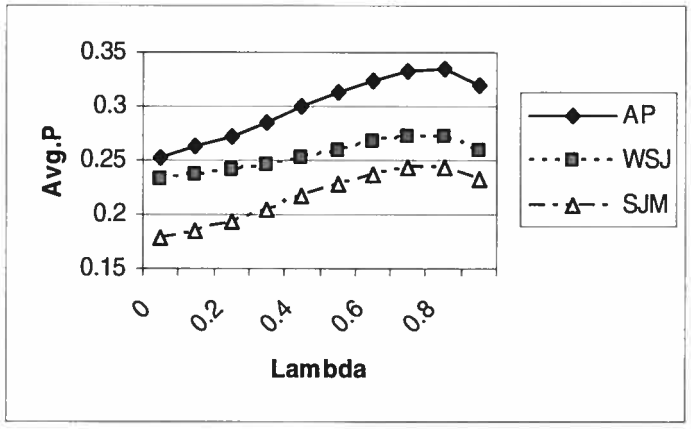

Figure 14. CDQE effectiveness w.r.t. α_K (*Jelinek-Mercer*)

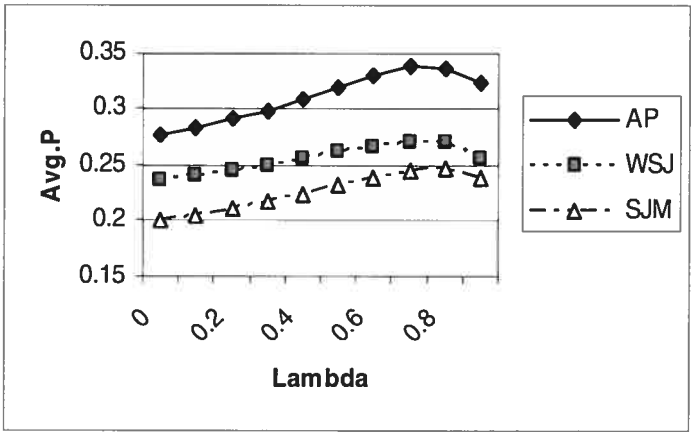

Figure 15. CDQE effectiveness w.r.t. α_K (*Dirichlet*)

We can see that for all the three collections and for both document smoothing methods, the effectiveness is good when the parameter is set in the range of 0.6-0.8. The best value for different collections remains stable at 0.7-0.8.

The effect of α_K on **CIQE** is slightly different, as we can see in the following figures:

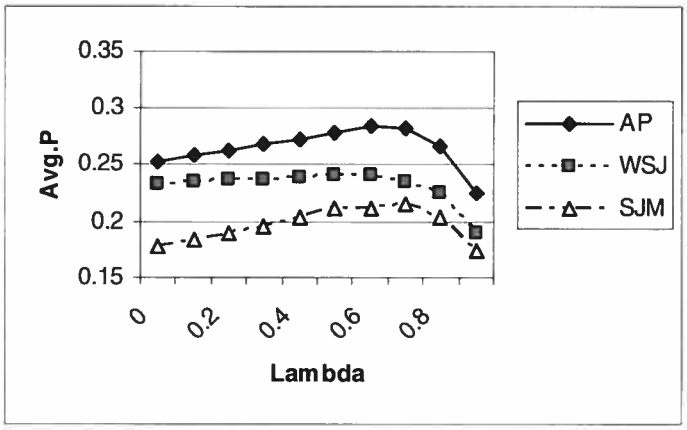

Figure 16. CIQE effectiveness w.r.t. α_K (*Jelinek-Mercer*)

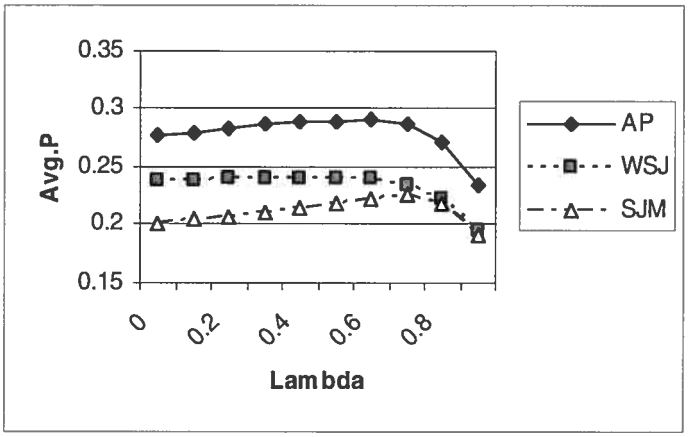

Figure 17. CIQE effectiveness w.r.t. α_K (*Dirichlet*)

We can see that the increases in retrieval effectiveness are less than with **CDQE**. In particular, for the WSJ collection, only slight improvements are observed. This

comparison shows that context-dependent term relations can make larger impact than context-independent term relations.

8.2.3.4 Number of Expansion Terms

In the previous tests, we limited the number of expansion terms to 80. When different numbers of expansion terms are used, we obtain different effectiveness results. The following figures show the variation of average precision (AvgP) with different numbers of expansion terms, using **CDQE** method.

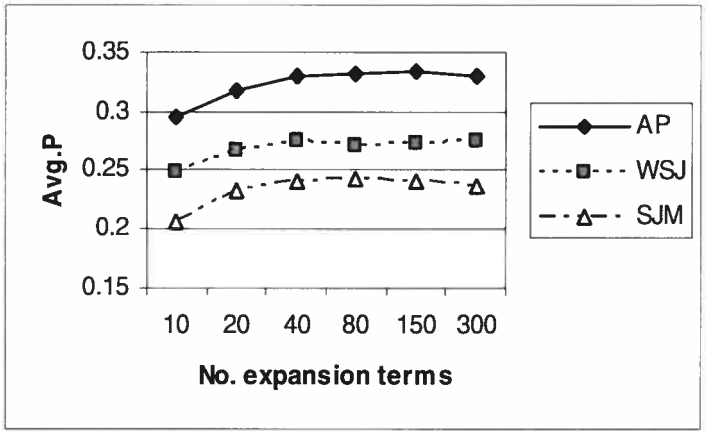

Figure 18. CDQE effectiveness w.r.t. # of expansion terms (*Jelinek-Mercer*)

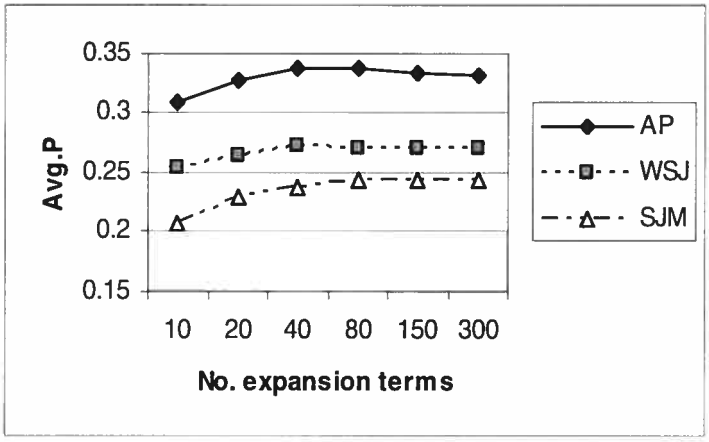

Figure 19. CDQE effectiveness w.r.t. # of expansion terms (*Dirichlet*)

We can see that when more expansion terms are added, the effectiveness does not always increase. This is because when more terms are added, there is a larger chance to introduce noise terms. In general, a number around 80 produces good results. In some cases, even if better effectiveness can be obtained with more expansion terms, the retrieval time is also longer. Therefore, a number around 80 seems to be a good compromise between effectiveness and retrieval speed: the retrieval time remains less than 1 second per query.

8.2.3.5 Combining Biterms with Co-occurrence Relations

As both **CDQE** and **CIQE** can improve retrieval effectiveness to some degree, an intuitive idea would be to combine them. This corresponds to the idea of relation smoothing. In addition, some of the test queries only contain one term, thus **CDQE** cannot apply. In this case, we can still use **CIQE**.

We tested two strategies:

- For each query, we use both **CDQE** and **CIQE**. They are combined using a smoothing factor. The smoothing parameter is tuned to its best in order to see the maximum effectiveness we can obtain;
- We have observed that for all the queries to which **CDQE** is applicable, the effectiveness with **CDQE** is always higher than with **CIQE**. Therefore, an alternative strategy is to choose to use **CDQE** when it can apply, and use **CIQE** when **CDQE** does not apply (for single word queries).

Table 9 shows the results with both strategies. We can see in the table that both combinations do not improve much the retrieval effectiveness. Globally, the second strategy is slightly better than the first one. This shows that the systematic introduction of **CIQE**, when **CDQE** is also used, is not very helpful. On the other hand, if we choose to use **CIQE** only when **CDQE** does not apply, the former can add some additional

impact on single-word queries. Still, the overall impact is very small. So our conclusion is that such a combination is not very useful.

Table 9. Combine biterm and co-occurrence relations (*Dirichlet*)

Collection	Measure	UM	CDQE	CDQE + CIDE	CDQE or CIDE
AP Q51-100	AvgP	0.2767	0.3383	0.3415 (+0.95%)	0.3411 (+0.83%)
	Recall / 6 101	3 677	4 029	4 074	4 061
	P@10	0.4408	0.5082	0.4980	0.4959
SJM Q101-150	AvgP	0.2017	0.2448	0.2448 (--%)	0.2481 (+1.35%)
	Recall / 2 559	1 641	1 873	1 873	1 897
	P@10	0.3152	0.3870	0.3870	0.3826
SJM Q101-150	AvgP	0.2373	0.2710	0.2710 (--%)	0.2725 (+0.55%)
	Recall / 2 172	1 588	1 737	1 737	1 750
	P@10	0.3292	0.3625	0.3625	0.3583

8.2.3.6 Selecting Biterms from the Query

The previous tests on **CDQE** used all the biterms included in a query and all biterms are weighted equally. However, some biterms may not be correct, in the sense that they may not represent the correct dependency in the query.

In an attempt to select and weight the best biterms, we tested the following two approaches:

- Using mutual information (MI) as the weight of biterms;
- Using maximum spanning tree (MST) to select the strongest biterms that cover all the query terms. Again, we use MI as the weight of biterms.

The following tables compare these methods with the previous one:

Table 10. Biterm selection in short queries (Title only, *Dirichlet*)

Collection	Measure	UM	CDQE	CDQE (MI)	CDQE (MST)
AP Q51-100	AvgP	0.2767	0.3383	0.3210 (-5.11%)	0.3183 (-5.91%)
	Recall / 6 101	3 677	4 029	3 890	4 024
	P@10	0.4408	0.5082	0.4531	0.4776
SJM Q101-150	AvgP	0.2017	0.2448	0.2643 (+7.97%)	0.2566 (+4.82%)
	Recall / 2 559	1 641	1 873	1 921	1 868
	P@10	0.3152	0.3870	0.4152	0.4022
WSJ Q51-100	AvgP	0.2373	0.2710	0.2614 (-3.54%)	0.2612 (-3.62%)
	Recall / 2 172	1 588	1 737	1 686	1 678
	P@10	0.3292	0.3625	0.3563	0.3521

As we can see, the two selection and weighting methods are not better than the previous simple method (i.e., **CDQE** in the table). We observe some improvements only on one collection, while there are decreases on the other two collections.

The following reason may explain this result: the biterms removed by these methods are not always noise biterms. The correct biterms can also be removed. In fact, MI is only a statistical measure. A biterm with strong MI value may be a wrong biterm in a query, and vise versa. This is because a query can contain two terms that do not connect with each other, while they have a strong relationship in the collection. For example, from the phrase "*computer used in education science*", it is possible that this method will extract the biterm "*computer science*" because of their statistical relationship in the collection, but it is not a correct one in this example.

One may think that these selection and weighting methods can be more effective on longer queries, since there may be more need to remove noisy biterms. To test this,

we use long queries which contain both "*<title>*" and "*<description>*" fields of the topics. The table below shows the results:

Table 11. Biterm selection in long queries (Title + Description, *Dirichlet*)

Collection	Measure	UM	CDQE	CDQE (MI)	CDQE (MST)
AP Q51-100	AvgP	0.3001	0.3403	0.3108 (-8.67%)	0.3171 (-6.82%)
	Recall / 6 101	3 860	4 080	3 998	4 018
	P@10	0.4531	0.5143	0.4592	0.4816
SJM Q101-150	AvgP	0.2538	0.2786	0.2662 (-4.45%)	0.2717 (-2.48%)
	Recall / 2 559	1 869	1 983	1 983	1 965
	P@10	0.3870	0.4109	0.4087	0.4022
WSJ Q51-100	AvgP	0.2522	0.2722	0.2696 (-0.96%)	0.2701 (-0.77%)
	Recall / 2 172	1 670	1 731	1 683	1 678
	P@10	0.3292	0.3542	0.3479	0.3583

Surprisingly, we see that the effectiveness is even worse in most cases. This indicates that, on the one hand, the methods we used to select and weight biterms are not effective; on the other hand, there may not be needed to filter out biterms.

To understand the reason, we have to go back to the fundamental goal of biterms: the biterm relations aim to suggest related terms with two words that co-occur in the same window. They are not designed to capture stricter relations such as syntactic or semantic relations. In the above filtering process, we were trying to use biterm relations to capture the latter, which is not appropriate. Indeed, the notion of "*correctness*" of biterms does not apply. Any biterm that we can encounter are legitimate biterms. So no filtering is required. By removing part of the biterms, we are indeed reducing the

potential impact of context words: some context terms will not be considered after the filtering. Our experimental results suggest that we should consider all the context terms instead.

This result also indicates that biterm relations can be widely applied without further conditions: whenever we encounter a biterm in a query, the corresponding relation applies.

8.2.3.7 Suitability of Relations across Collections

In many real applications (e.g., Web search), we do not have a static document collection from which term relations can be extracted. The question is whether it is possible and beneficial to extract relations from one text collection and use them to retrieve documents in another text collection. Our intuition is that this is possible because the relations (especially context-dependent term relations) encode general knowledge, which can be applied to a different collection. In order to show this, we extract term relations from each collection, and apply them on other collections. The following tables show the effectiveness produced using unigram and biterm relations respectively.

Table 12. Cross-utilization of term relations (*Dirichlet*)

Rel. / Coll.	Unigram relation			Biterm relation		
	AP	**SJM**	**WSJ**	**AP**	**SJM**	**WSJ**
AP	**0.2902**	0.2803	0.2793	**0.3383**	0.3057	0.2987
SJM	**0.2271**	0.2225	0.2267	0.2424	0.2448	**0.2453**
WSJ	**0.2541**	0.2445	0.2393	**0.2816**	0.2636	0.2710

From this table, we can observe that relations extracted from any collection are useful to some degree: they all outperform **UM**. In particular, the relations extracted from AP are the best for almost all the collections. This can be explained by the larger size and thus possibly wider coverage of the AP collection than the two other collections. This result suggests that we do not necessarily need to extract term relations from the same text collection on which retrieval is performed. It is possible to extract relations from a large text collection, and apply them to other collections. This opens the door to the possibility of constructing a general relation base for various document collections.

8.2.3.8 Adding More Terms into Conditions

We have compared term relations with one term and two terms in the condition part. The latter produced better retrieval results. A legitimate question is whether further enhancement of the condition can help suggest even better expansion terms.

In order to test this, we have produced *triterm* relations of the following form:

$$\{t_j, t_k, t_l\} \rightarrow t_i$$

These relations have been extracted in a similar way as biterm relations.

We have compared three types of relations on queries with 3 or more terms. Table 13 shows the results.

We can see that increasing the number of terms in the condition of relations does not lead to better results. The overall effectiveness of triterm relations is comparable to the simple co-occurrence relations. Both are well below the effectiveness of biterm relations.

This comparison validates our earlier hypothesis that by adding one additional term into the condition, the two terms can mutually precise the meaning of the other term and one additional context term is often sufficient.

Table 13. Comparing co-occurrence, biterm and triterm relations (*Dirichlet*)

Collection	Measure	UM	CIQE (Co-occur.)	CDQE (Biterm rel.)	CDQE (Triterm rel.)
AP Q51-100	AvgP	0.1998	0.2045	**0.2569**	0.2067
	Recall / 3 581	1 931	2 039	2 178	1 988
	P@10	0.3485	0.3515	0.4212	0.3576
SJM Q101-150	AvgP	0.2113	0.2320	**0.2630**	0.2168
	Recall / 1 849	1 183	1 278	1 401	1 214
	P@10	0.3359	0.3897	0.4128	0.3436
WSJ Q51-100	AvgP	0.1679	0.1628	**0.1810**	0.1710
	Recall / 1 315	888	946	1025	933
	P@10	0.2406	0.2500	0.2719	0.2437

Triterm relations could also suggest strongly related terms. However, as we add more terms into the condition, we can observe fewer occurrences of co-occurring terms. Therefore, we are more exposed to the problem of data sparseness. The extracted triterm relations may not have a good coverage. This may explain why triterm relations perform worse than biterm relations.

8.2.4 Evaluating Second-Order Term Relations

We mentioned that some strong relations (e.g., synonymy-like relations) cannot be extracted directly from co-occurrences. Second-order relations can be used to cover this type of relation. In the following experiments, we compare the second-order relations with the two previous ones. They are carried out on a TREC collection: the documents from *Associated Press* (AP88-89) contained in TREC Disks 1 and 2. We use two sets of queries that have been used respectively in TREC 2 and 3 *ad hoc* tracks [40][41]: 101-150 and 151-200. Only the titles of the topics are used as queries. Some statistics are shown in Table 14.

Table 14. Collection statistics for testing second-order relations

Collection	Description	Size (MB)	Vocabulary	# of Doc.	Query
AP	*Associated Press* (1988-89)	491	196,933	164,597	101-150 151-200

We also use a slightly different dataset for training: AP89 collection (TREC Disk 1) and topics 1-50 for setting the mixture weights.

The previous models will be compared with the following ones:

- **HAL-based query expansion (HAL):**

  This model is similar to the co-occurrence relations, except that we replace the traditional co-occurrence relations by the **HAL** co-occurrence relations. We remind that the main difference between them is that **HAL** co-occurrences implicitly consider the distance between the words and use it as a decaying factor (the more distant are the words, the less strong is their relation), and that heuristics are used to combine terms in the query.

- **IF-based query expansion (IF):**

  From **HAL** co-occurrence relations, information flow (**IF**) relations are extracted between a combination of terms (query) and another term. These are second-order relations.

Table 15 and Table 16 show the experimental results. The percentages in the table are the relative changes with respect to the baseline LM without query expansion. In general, the methods under comparison perform similarly with both smoothing methods. The general trend is: **CDQE > IF > CIQE > HAL > UM**.

Table 15. Query expansion with different relations (*Jelinek-Mercer*)

Coll.	Measure	UM	CIQE	HAL	IF	CDQE
AP Q101-150	AvgP	0.2042	0.2392 (+17.14%)	0.2163 (+5.93%)	0.2686 (+31.54%)	**0.2800 (+37.12%)**
	Recall/ 4 805	3 021	3 337	3 118	3 691	**3 756**
	P@10	0.3600	0.4040	0.3720	0.4340	**0.4560**
AP Q151-200	AvgP	0.2876	0.3234 (+12.45%)	0.3057 (+6.29%)	0.3377 (+17.42%)	**0.3636 (+26.43%)**
	Recall/ 4 933	3 367	3 551	3 472	3 555	**3 777**
	P@10	0.4900	0.5220	0.5220	0.4960	**0.5420**

Table 16. Query expansion with different relations (*Dirichlet*)

Coll.	Measure	UM	CIQE	HAL	IF	CDQE
AP Q101-150	AvgP	0.2304	0.2568 (+11.46%)	0.2379 (+3.26%)	0.2744 (+19.10%)	**0.2886 (+25.26%)**
	Recall/ 4 805	3 157	3 393	3 227	3 722	**3 812**
	P@10	0.3880	0.4140	0.4000	0.4200	**0.4460**
AP Q151-200	AvgP	0.3132	0.3337 (+6.55%)	0.3235 (+3.29%)	0.3503 (+11.85%)	**0.3715 (+18.61%)**
	Recall/ 4 933	3 425	3 630	3 488	3 624	**3 847**
	P@10	0.5180	0.5160	0.5240	0.5160	**0.5340**

Observations:

- **Query expansion by HAL vs. by co-occurrence relations:**

When **HAL** relations are used for query expansion, we can obtain improvements of around 3-6% for each of the smoothing methods. However, compared to the traditional co-occurrence relations, **HAL** relations do not seem to have any advantage.

Their effectiveness is even lower. This may show that the consideration of the distance between the terms (the decaying factor) and the heuristics used to group terms are not very useful. The statistics of raw co-occurrence frequency can be better.

- **Query expansion by IF relations vs. by biterm relations:**

In contrast, when we use **IF** to expand queries (LM with IF), the effectiveness is much improved compared to **HAL** relations. This experiment shows that **IF** combined with LM can indeed add interesting terms into queries, which cannot be added using raw co-occurrence and **HAL** relations.

However, compared to the context-dependent term relations (**CDQE**), we see that **IF** relations are still less effective. In fact, this second-order relations can only take into account context words indirectly. This is less effective than a direct solution to deal with context words. In addition, the computation cost to obtain **IF** relations is also much higher. In conclusion, it is better to use context-dependent term relations than second-order term relations.

8.3 Second Experiments: Integrating Contextual Factors

In the second experiments, we investigate the impact of integrating different contextual factors: **knowledge**, **domain** and **feedback**.

8.3.1 Test Collections

The main dataset are those from TREC Disks 1-3 *ad hoc* tracks, including queries 1-150, and documents including AP (*Associated Press*), SJM (*San Jose Mercury*

News), WSJ (*Wall Street Journal*), FR (*Federal Register*), Ziff (*Information from Computer Select disks by Ziff-Davis*, PAT (*U.S. Patents*), DOE (*Department of Energy Abstracts*), FBIS (*Foreign Broadcast Information Service*), LAT (*Los Angeles Times*) and FT (*Financial Times Limited*). The queries 1-150 contain manually specified domain for each query, as we can see in the query topic example given earlier (see section 8.2.1). This allows us to compare with an approach using automatic domain identification. Again, we only use topic titles for the queries in all our tests. In these experiments, queries 1-50 are used for training and 51-150 for testing.

13 domains are defined and assigned to these queries: *Environment*, *Finance*, *Science and technology*, etc. Figure 20 shows their distributions among the training and test queries. We can see that the distribution varies strongly between domains and between the two query sets.

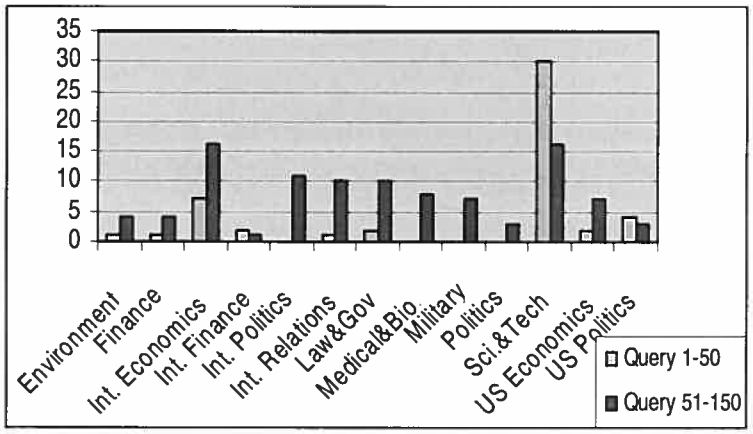

Figure 20. Distribution of domains

As additional test collections, we also use TREC 7 and 8 data, for which no domain is manually indicated for queries. These data are mainly used to test whether we can automatically assign a domain to each query and whether this is helpful. Some statistics of the data are described in Table 17.

Table 17. TREC collection statistics for second series of experiments

Collection	Document sources	Size (G)	Vocab.	# of Doc.	Query
Training (Disk 2)	AP (1988), FR (1988), WSJ (1990-92), Ziff-Davis (1989-90)	0.86	350,085	231,219	1-50
Disks 1-3	AP (1988-90), DOE, FR (1988-89), WSJ (1987-92), Ziff-Davis (1989-90), PAT, SJM (1991)	3.10	785,932	1,078,166	51-150
TREC 7 (Disks 4-5)	FBIS (1994), FR (1994), FT (1991-94), LAT (1989-90)	1.85	630,383	528,155	351-400
TREC 8 (Disks 4-5)		1.85	630,383	528,155	401-450

As previously, all the documents are preprocessed using *Porter* stemmer in *Lemur* and stopwords are removed. Notice that some queries only contain one word: there are 4, 5 and 3 respectively in the three test query sets. For these queries, context-dependent term relations do not apply.

8.3.2 Baseline Methods

Two baseline models are used: the classical unigram model without any expansion, and the model with **FB** (feedback). This latter model is also used as a baseline because this is a common method in current IR, and it produces the state-of-the-art effectiveness. The weight of the feedback model is set at 0.8, which is suggested by [115].

In all the experiments, document models are smoothed by *Jelinek-Mercer* smoothing. This choice is made because our previous experiments did not show much difference between the two smoothing methods. Another reason is due to the

observation made in [116] that this smoothing performs very well for long queries. In our case, as queries are heavily expanded, they perform similarly to long queries.

Table 18 shows the retrieval effectiveness on all the collections with the two baseline methods. As found in many previous studies, the model with feedback performs much better than the basic unigram model. The differences between them are statistically significant.

Table 18. Baseline models

Collection	Measure	Unigram Model	
		Without FB	**With FB**
Disks 1-3	AvgP	0.1570	0.2344 (+49.30%)**
	Recall /48 355	15 711	19 513
	P@10	0.4050	0.5010
TREC7	AvgP	0.1656	0.2176 (+31.40%)**
	Recall /4 674	2 237	2 777
	P@10	0.3420	0.3860
TREC8	AvgP	0.2387	0.2909 (+21.87%)**
	Recall /4 728	2 764	3 237
	P@10	0.4340	0.4860

8.3.3 Knowledge Models

Let us first test the integration of the **knowledge** model, together with the **feedback** model. We also compare the context-dependent knowledge model with the traditional co-occurrence model. The column **WithoutFB** is compared to the baseline model without feedback, while **WithFB** is compared to the baseline with feedback. *T*-test is performed for statistical significance: ** and * mean significant changes in *t*-test with respect to the baseline without feedback at the level of $p < 0.01$ and $p < 0.05$

respectively, while ++ and + are similar but compared to the baseline model with feedback.

Table 19. Integration of knowledge models

Collection	Measure	CIQE		CDQE	
		Without FB	With FB	Without FB	With FB
Disks1-3	AvgP	0.1884 (+20.00%)**	0.2432 (+3.75%)++	0.2164 (+37.83%)**	0.2463 (+5.08%)++
	Recall/ 48 355	17 430	20 020	18 944	20 260
	P@10	0.4640	0.5160	0.5050	0.5120
TREC7	AvgP	0.1823 (+10.08%)**	0.2350 (+8.00%)+	0.2157 (+30.25%)**	0.2401 (+10.34%)++
	Recall / 4 674	2 329	2 933	2 709	2 985
	P@10	0.3780	0.3760	0.3900	0.3900
TREC8	AvgP	0.2519 (+5.53%)	0.2926 (+0.58%)	0.2724 (+14.12%)**	0.3007 (+3.37%)
	Recall / 4 782	2 829	3 279	3 090	3 338
	P@10	0.4360	0.4940	0.4720	0.5000

Once again, we can observe that simple co-occurrence relations produce some improvements, but context-dependent term relations can produce much stronger improvements in all cases.

When the feedback model is added, the additional improvements with both types of relations are reduced. This is not surprising. This result indicates that there is a common effect produced by **feedback** model and **knowledge** model.

However, we still observe higher effectiveness than the baseline with feedback. In two out of three cases, the additional improvements are statistically significant. This indicates that another part of the effects produced by **feedback** and **knowledge** is different and complementary. So both models can be combined to produce even better effectiveness.

8.3.4 Domain Models

In this section, we test several strategies to create and to use **domain** models. There are two different ways to create domain models and two different ways to determine the domain for queries.

Strategies for creating domain models:

- **Manual collection (C1)**: With this strategy, we collect documents that are classified manually into domains. To simulate this process, we use the ***relevant documents*** manually judged to select example documents for each domain. In order to avoid bias, when we test on a query, only the relevant documents for the other queries are used to build domain models;
- **Automatic collection (C2)**: This strategy simulates the situation where we do not have manually judged documents in domains. However, the user is willing to indicate the domain of his queries during the period of domain construction. To collect documents for different domains, we simply use the ***top-100 documents*** retrieved with the queries in the corresponding domain. This is a strategy similar to those which observe the user's interactions with the system in order to perform personalized IR. Again, we exclude the current test query in the domain construction process.

Strategies for using domain models:

- **Manual classification (U1)**: The domain model is determined by the ***user manually***. This strategy can be tested on the first dataset (TREC Disks 1-3);
- **Automatic classification (U2)**: The domain model is determined by the ***system automatically*** using query classification. This strategy can also be tested on the second and third dataset (TREC 7 and 8).

8.3.4.1 Creation of Domain Models

We compare strategies C1 and C2 for **domain** model creation. In this series of tests, each of the queries 51-150 is used in turn as the test query while the other queries and their relevant documents (C1) or top-ranked retrieved documents (C2) are used to create domain models. The same method is used on Disk 2 for queries 1-50 to tune the mixture weights.

We mentioned that for large domains, the whole domain model may not be very useful for a particular query. Therefore, sub-domain models are also created and compared with whole domain models.

In the following tests (Table 20 and Table 21), we use manual identification of query domain for Disks 1-3 (U1), but automatic identification for TREC 7 and 8 (U2).

First, it is interesting to observe that the incorporation of **domain** models can generally improve retrieval effectiveness in all the cases. The improvements on Disks 1-3 and TREC 7 are statistically significant. However, the improvement scales are smaller than using **feedback** and **knowledge** models. The smaller improvements can be partly explained by the fact that we do not have sufficient training data to create domain models. Looking at the distribution of the domains (Figure 20), we can see that we only have few training queries in several cases. In addition, topics in the same domain can vary greatly, in particular in large domains such as *Science and technology*, *International politics*, etc. Therefore, the domain models can cover a few topics in the domain. It is expected that this problem can be partly solved when the number of documents in a domain grows: the coverage of the domain model will be better and less skewed.

Table 20. Domain models with relevant documents (C1)

Collection	Measure	Domain		Sub-Domain	
		Without FB	With FB	Without FB	With FB
Disks1-3 (U1)	AvgP	0.1700 (+8.28%)**	0.2454 (+4.69%)++	0.1918 (+22.17%)**	0.2461 (+4.99%)++
	Recall / 48 355	16 517	20 141	17 872	20 212
	P@10	0.4370	0.5130	0.4490	0.5150
TREC7 (U2)	AvgP	0.1715 (+3.56%)**	0.2389 (+9.79%)+	0.1842 (+11.23%)**	0.2408 (+10.66%)++
	Recall / 4 674	2 270	2 965	2 428	2 987
	P@10	0.3720	0.3740	0.3880	0.3760
TREC8 (U2)	AvgP	0.2442 (+2.30%)	0.2957 (+1.65%)	0.2563 (+7.37%)	0.2967 (+1.99%)
	Recall / 4 728	2 796	3 308	2 873	3 302
	P@10	0.4420	0.5000	0.4280	0.5020

Table 21. Domain models with top-100 documents (C2)

Collection	Measure	Domain		Sub-Domain	
		Without FB	With FB	Without FB	With FB
Disks1-3 (U1)	AvgP	0.1718 (+9.43%)**	0.2456 (+4.78%)++	0.1799 (+14.59%)**	0.2452 (+4.61%)++
	Recall / 48 355	16 558	20 131	17 341	20 155
	P@10	0.4300	0.5140	0.4220	0.5110
TREC7 (U2)	AvgP	0.1765 (+6.58%)**	0.2395 (+10.06%)++	0.1785 (+7.79%)**	0.2393 (+9.97%)++
	Recall / 4 674	2 319	2 969	2 254	2 968
	P@10	0.3780	0.3820	0.3820	0.3820
TREC8 (U2)	AvgP	0.2434 (+1.97%)	0.2949 (+1.38%)	0.2441 (+2.26%)	0.2961 (+1.79%)
	Recall / 4 728	2 772	3 318	2 734	3 311
	P@10	0.4380	0.4960	0.4280	0.5020

Second, we observe that the two methods to create domain models perform equally well (Table 20 vs. Table 21). In other words, providing relevance judgments for queries or performing a manual classification does not add much advantage for the purpose of creating domain models. This may seem surprising. However, an analysis immediately shows the reason: a domain model (in the way we created) only captures term distribution in the domain. Even if the top documents retrieved for the in-domain queries are not relevant to the query, they are nevertheless selected because of their strong correspondence with the query terms. So these documents still contain characteristic terms related to the query. Therefore, they can be assumed to be related to the domain, and be able to produce a reasonable term distribution for the domain. This result opens the door for a simpler method that does not require relevance judgments, for example using user's search history.

Third, the **sub-domain** models can bring very strong improvements in retrieval effectiveness when the feedback model is not used. When the feedback model is used, the improvements lessen. Compared to the baseline with feedback model, the utilization of sub-domain models can still bring quite strong improvements which are significant in 2 cases out of 3. However, compared to the case where the whole domain model is used together with the feedback model, we do not see additional advantage of using sub-domain models.

This observation can be explained by the fact that the **sub-domain** model exploits a similar idea to pseudo relevance **feedback**. The only difference between the feedback model and the sub-domain model is that the former is extracted from the whole collection, while the latter is extracted from a domain. In both cases, some documents related to the query will be obtained. The terms from these documents, extracted either from the whole collection or from a domain, can be very similar. Therefore, when the feedback model is already added, no additional effect is produced by the sub-domain model.

Finally, we can observe that in both cases with and without feedback models, the impact of the domain model is steady. This indicates that the effects of both models are different and complementary. It is beneficial to combine them.

8.3.4.2 Determining Query Domain Automatically

It is not realistic to always ask users to specify a domain for their queries. Here we examine the possibility to identify the query domain automatically, once domain models have been constructed.

Table 22 shows the results with this strategy using both ways to construct domain models. We can observe that with automatic domain identification, the effectiveness is only slightly lower than those produced with manual identification of query domains (see Table 20 and Table 21). This shows that automatic domain identification is a feasible way to select domain models. Therefore, once domain models are constructed, the user can issue queries as usual without having to indicate their domains. The domain model constructed previously can be automatically used to enhance these queries.

Table 22. Automatic query domain identification (U2)

Coll.	Measure	Dom. with rel. doc. (C1)		Dom. with top-100 doc. (C2)	
		Without FB	With FB	Without FB	With FB
Disks1-3 (U2)	AvgP	0.1650 (+5.10%)**	0.2444 (+4.27%)++	0.1670 (+6.37%)**	0.2449 (+4.48%)++
	Recall / 48 355	16 343	20 061	16 414	20 090
	P@10	0.4270	0.5100	0.4090	0.5140

Looking at the accuracy of the automatic domain identification, however, we see that it is quite low: for queries 51-150, only 38% of the domains determined

automatically correspond to the manual identifications. The following table shows more details about the accuracy of automatic domain identification:

Table 23. Accuracy of automatic domain identification

Domain	Number	Domain	Number
Environment	2 / 4	Medical and Biological	6 / 8
Finance	0 / 4	Military	5 / 7
International Economics	4 / 16	Politics	0 / 3
International Finance	0 / 1	Science and Technology	8 / 16
International Politics	8 / 11	U.S. Economics	2 / 7
International Relations	3 / 10	U.S. Politics	0 / 3
Law and Government	0 / 10		

This is much lower than the 80% rates reported in [58]. A detailed analysis reveals that the main reason is the closeness of several domains in TREC queries. For example, *International relations*, *International politics*, *Politics* can all contain similar terms. In this case, a query in one of these domains can be easily classified into a wrong domain.

However, in this situation where domains are close, a wrong domain assigned to a query is not always irrelevant and useless. For example, even when a query in *International relations* is classified into *International politics*, the latter domain can still suggest useful terms to the query. Therefore, the relatively low classification accuracy does not mean low usefulness of the domain models. The terms from the identified domain can still be useful.

It would also be possible to use several domain models, each with a weight $P(domain|Q)$, to expand the query. This will alleviate the problem due to wrong classification. We leave it to a future work.

8.3.5 Extracting Term Relations within Domains

We mentioned that one possible approach to select more relevant expansion terms is to use domain-specific term relations. In the following experiments, we will test whether it is useful to extract term relations from documents in each domain and use them to expand the queries in that domain.

Again, we use the relevant documents judged for the other queries to constitute a domain. Both co-occurrence term relations and context-dependent term relations are extracted and used. Table 24 shows the results when such relations are used.

Table 24. Domain-specific term relations (C1)

Coll.	Measure	Domain	Domain-specific Co-occurrence	Domain-specific Biterm
Disks1-3 (U1)	AvgP	0.1700	0.1908 (+12.24%)**	0.1979 (+16.41%)**
	Recall / 48 355	16 517	17 886	18 176
	P@10	0.4370	0.4760	0.4740
TREC7 (U2)	AvgP	0.1715	0.1847 (+7.70%)*	0.1847 (+7.70%)*
	Recall / 4 674	2 270	2 427	2 403
	P@10	0.3720	0.3860	0.3800
TREC8 (U2)	AvgP	0.2442	0.2552 (+4.50%)	0.2531 (+3.64%)
	Recall / 4 728	2 796	2 839	2 833
	P@10	0.4420	0.4300	0.4300

From this table, we can see that if we extract term relations from the documents in each domain, the term relations obtained can make a larger impact on retrieval effectiveness than when the domain model is used (as a term distribution). This may show that the domain model that we use can be too large for a particular query: for a

query in a domain, it may not be the best strategy to add all the strongest terms into that query. Some of these terms may not be related to the query. On the other hand, when we apply the relations extracted from the domain, only the related terms are added. Therefore, compared to the previous domain models, the extraction and utilization of term relations from the domains make a further selection on the expansion terms.

Compared with the term relations extracted from the whole document collection, we can see that the co-occurrence relations extracted from the domains perform slightly better than those extracted from the whole collection. This suggests that this type of relation is highly ambiguous. When the extraction is restricted within a specific domain, the relations are more appropriate for queries in that domain.

On the other hand, for context-dependent relations, the relations extracted from the whole collection are better than those extracted from the domain. This shows that the context-dependent relations are much less ambiguous. Even if they are extracted from the whole collection, they have much less danger than the co-occurrence relations to be applied in wrong contexts. The fact to restrict the extraction within a domain will simply reduce the coverage of the relations. Therefore, such a domain-specific extraction is not necessary for context-dependent relations.

The above results also validate our earlier hypothesis that by adding some context words into terms relations, the relations become less ambiguous and can be applied in the correct contexts.

8.3.6 Complete Models

The results with the complete model are shown in Table 25 and Table 26. This model integrates all the components described in this thesis: **original query model**, **domain model**, **knowledge model** and **feedback model**. The results are compared to both baseline methods. The improvements indicated in (.) and [.] are over unigram

model and feedback model respectively, and */** and +/++ indicate whether the difference is statistically significant at the level of $p < 0.05$ and $p < 0.01$.

In these experiments, the parameters are tuned using the line search method and EM algorithm described in Chapter 7 respectively. When using line search to tune the parameters, the training collection contains documents from Disk 2 and queries 1-50. The values of the parameters are: $\alpha_0 = 0.1$, $\alpha_K = 0.2$, $\alpha_{Dom} = 0.1$ and $\alpha_{FB} = 0.6$. These values are used for the other test collections.

Our first observation is that the **complete** models always produce the best results compared to the other cases where only some of the query models are used. All the improvements over both the baseline models (with or without feedback) are statistically significant. This result confirms that the integration of other contextual factors is beneficial.

Let us look at the mixture weights, which may reflect the importance of each model: $\alpha_0 = 0.1$, $\alpha_K = 0.2$, $\alpha_{Dom} = 0.1$ and $\alpha_{FB} = 0.6$. We see that the most important factor is **feedback** model. This is also the single factor which produced the highest improvements over the original query model. However, even with lower weights, the other models do have strong impacts on the final effectiveness. This demonstrates the benefit of integrating more contextual factors in IR.

Table 25. Complete models (C1)

Collection	Measure	All Document Domain	
		Manu. Dom. Id. (U1)	Auto. Dom. Id. (U2)
Disks 1-3	AvgP	0.2501 (+59.30%)** [+6.70%]++	0.2489 (+58.54%)** [+6.19%]++
	Recall /48 355	20 514	20 367
	P@10	0.5200	0.5230
TREC7	AvgP	N/A	0.2462 (+48.67%)** [+13.14%]++
	Recall /4 674		3 014
	P@10		0.3960
TREC8	AvgP	N/A	0.3029 (+26.90%)** [+4.13%]++
	Recall /4 728		3 321
	P@10		0.5020

Table 26. Complete models (C2)

Collection	Measure	All Document Domain	
		Manu. Dom. Id. (U1)	Auto. Dom. Id. (U2)
Disks 1-3	AvgP	0.2502 (+59.36%)** [+6.74%]++	0.2495 (+58.92%)** [+6.44%]++
	Recall /48 355	20 474	20 419
	P@10	0.5220	0.5190
TREC7	AvgP	N/A	0.2469 (+49.09%)** [+13.47%]++
	Recall /4 674		3 014
	P@10		0.4020
TREC8	AvgP	N/A	0.3022 (+26.60%)** [+3.88%]++
	Recall /4 728		3 322
	P@10		0.4960

Let us summarize all the experiments presented so far in the following table:

Table 27. Summary of experiments

Coll.	Mea-sure	UM	UM + Dom		UM + CDQE	UM + FB	Complete	
			C1	C2			C1	C2
Disks 1-3 (U1)	AvgP	0.1570	0.1700** (+8.28%)	0.1718** (+9.43%)	0.2164** (+37.83%)	0.2344** (+49.30%)	**0.2501** (+59.30%)**	**0.2502** (+59.36%)**
	Recall /48 355	15 711	16 517	16 558	18 944	19 513	**20 514**	**20 474**
	P@10	0.4050	0.4370	0.4300	0.5050	0.5010	**0.5200**	**0.5220**
TREC7 (U2)	AvgP	0.1656	0.1715** (+3.56%)	0.1765** (+6.58%)	0.2157** (+30.25%)	0.2176** (+31.40%)	**0.2462** (+48.67%)**	**0.2469** (+49.09%)**
	Recall /4 674	2 237	2 270	2 319	2 709	2 777	**3 014**	**3 014**
	P@10	0.3420	0.3720	0.3780	0.3900	0.3860	**0.3960**	**0.4020**
TREC8 (U2)	AvgP	0.2387	0.2442 (+2.30%)	0.2434 (+1.97%)	0.2724** (+14.12%)	0.2909** (+21.87%)	**0.3029** (+26.90%)**	**0.3022** (+26.60%)**
	Recall /4 728	2 764	2 796	2 772	3 090	3 237	**3 321**	**3 322**
	P@10	0.4340	0.4420	0.4380	0.4720	0.4860	**0.5020**	**0.4960**

The above table shows the following trend on all the collections:

Complete > UM + FB > UM + CDQE > UM + Dom > UM

From this, we can conclude that the more we incorporate contextual factors, the more the final model is effective.

On individual contextual factors, we can also observe the following trend:

FB > CDQE > Dom

8.3.6.1 Sensibility of Mixture Weights

We have performed an exhaustive test on all the possible combinations of mixture weights on different test collections. The best settings are very similar in all the cases. They are always in the following ranges: $0.1 \leq \alpha_0 \leq 0.2$, $0.1 \leq \alpha_K \leq 0.2$, $0.1 \leq \alpha_{Dom} \leq 0.2$ and $0.5 \leq \alpha_{FB} \leq 0.6$. More specifically, for both manual identification and automatic identification of query domain (U1 and U2), the best settings are respectively:

- TREC Disks 1-3: $\alpha_0 = 0.1$, $\alpha_K = 0.2$, $\alpha_{Dom} = 0.1$ and $\alpha_{FB} = 0.6$;
- TREC 7: $\alpha_0 = 0.1$, $\alpha_K = 0.2$, $\alpha_{Dom} = 0.1$ and $\alpha_{FB} = 0.6$;
- TREC 8: $\alpha_0 = 0.2$, $\alpha_K = 0.2$, $\alpha_{Dom} = 0.1$ and $\alpha_{FB} = 0.5$.

We can see the mixture weights across different collections are quite stable.

8.3.6.2 Using Unsupervised Training

We also tested the utilization of EM algorithm to tune the mixture weights. We tested two settings: when the collection model is combined with different queries models, we can let the mixture weight of the collection vary freely during the EM updates, or we fix it at 0.5. In both settings, the mixture weights vary over queries. So we cannot compare these mixture weights with those trained by supervised training.

We observe that in the first setting, the final retrieval effectiveness is slightly lower than with the second setting. This is because the influence of the collection model varies from a query to another, while in our score function the collection model is not used in the query model. Therefore, the component query models may share a variable part left by the collection model. This leads to very different mixture weights for the component query models. Using the second setting, the part of the collection model is fixed at 0.5, and the other component query models share the other part. Therefore, the relative importance of the mixture weights can be better determined. So we obtained slightly better results. Table 28 and Table 29 describe the results with the second setting.

Table 28. Complete models with unsupervised EM tuning (C1)

Collection	Measure	All Document Domain	
		Manu. dom. Id. (U1)	Auto. dom. Id. (U2)
Disks 1-3	AvgP	0.2474 (+57.58%)** [+5.55%]++	0.2469 (+57.26%)** [+5.33%]++
	Recall / 48 355	20 143	20 076
	P@10	0.5240	0.5260
TREC7	AvgP	N/A	0.2374 (+43.36%)** [+9.10%]
	Recall / 4 674		2 953
	P@10		0.3920
TREC8	AvgP	N/A	0.2738 (+14.70%) [-5.88%]
	Recall / 4 728		3 270
	P@10		0.4700

Table 29. Complete models with unsupervised EM tuning (C2)

Collection	Measure	All Document Domain	
		Manu. dom. Id. (U1)	Auto. dom. Id. (U2)
Disks 1-3	AvgP	0.2473 (+57.52%)** [+5.50%]++	0.2472 (+57.45%)** [+5.46%]++
	Recall / 48 355	20 129	20 082
	P@10	0.5260	0.5220
TREC7	AvgP	N/A	0.2380 (+43.72%)** [+9.38%]
	Recall / 4 674		2 954
	P@10		0.3940
TREC8	AvgP	N/A	0.2733 (+14.50%) [-6.05%]
	Recall / 4 728		3 271
	P@10		0.4700

From the above tables, we can observe that in general, the retrieval effectiveness is much improved over UM model (the percentage in (.) and **). However, compared to the model with feedback, for the two first collections, we observe further improvements. In particular, the improvement on the first collection is statistically significant. However, on the third collection, the effectiveness is lower than the baseline with feedback. However, notice that the mixture weights for the models with feedback have been tuned to their best, while those for the complete models are tuned in an unsupervised manner. So this lower effectiveness is not surprising.

Overall, the comparison between the two training methods indicates that if we have a training data with relevance judgments, then a supervised training is better than an unsupervised training. Especially, we have observed that the optimal parameters are quite stable across collections. However, in absence of such a training data, we can still use unsupervised training and this can also lead to better retrieval effectiveness than the baseline model.

8.4 Discussions

In these experiments, we have tested different query models: **knowledge**, **domain** and **feedback**. We have found that each of these models can enhance the original query model with MLE. The best results are obtained with the **complete** query model which integrates all the components models.

On term relations, our experiments demonstrated that context-dependent relations are more suited for query expansion than context-independent co-occurrence relations. In addition, this type of relation does not need to be extracted from the specific domain. It can be extracted on the whole collection, and they do not run a high risk to be applied in wrong contexts due to the context terms added into the condition.

On the creation of domain models, we have tested two approaches: with or without manual judgments. We have shown that it is possible to collect documents to train domain models without manual judgments. In addition, documents collected in this way perform equally well to the documents classified manually.

To determine the query domain, we have compared the manual identification and automatic identification. Our experiments show that both strategies can perform equally well. The differences between them are marginal.

Our experiments also show that when a new contextual factor is added, we always obtain improvements in retrieval effectiveness, although the scales of the improvement may vary. This result indicates that the three contextual factors we integrated can all enhance the query from different perspectives. Their effects are quite complementary. Overall, it is shown that the method that we proposed to integrate different contextual factors is both feasible and effective in practice.

Chapter 9

General Discussions and Conclusions

Traditional IR approaches usually consider the query as the only element available about user's information need. In reality, there are many other factors that specify the information need from different perspectives.

In this study, we exploited different contextual factors in order to complete the original query. The key problem is to add correct terms. For this purpose, we proposed to create context-dependent term relations to expand the query, to build domain models to capture background terms for the query, and to use pseudo feedback documents to reflect the collection characteristics related to the query. This completion of query aims to increase not only recall, but also precision. As our experiments have shown, when relevant terms are added into a query, the retrieval effectiveness is increased on all the recall levels, including for the top ranked documents.

Previous studies have also tried to add terms into queries, basically through the following approaches: query expansion using term relations, query expansion using pseudo relevance feedback, personalization. In comparison to these existing methods, we have developed new approaches to use these contextual factors and to integrate them. These approaches constitute our main contributions in this study:

- **Intra-query context – knowledge model:**

We observed that in traditional approaches to query expansion using term relations, the relations are usually created between two single terms. This led to the application of inappropriate relations. The key problem that we observed is the lack of context in the relations. Therefore, we proposed the creation of context-dependent term relations, which contain more terms in the condition of a relation that help us specify the correct situation to apply the relation. Our experiments showed that this new type of relation is more effective than traditional term relations. In addition, there is no need to add many context terms. With only one additional term in the condition part of relations, we can obtain the best results.

This type of relation exploits context words that exist within the query. These words define a new type of context – Intra-query context.

- **Extra-query context – domain model:**

Many previous studies have investigated personalization of IR. The usual approach consists of constructing a user profile corresponding to the interests of the user. However, we observed that such a user profile that mixes up all topic domains may not be effective for new queries. Therefore, we proposed to model topic domains instead of one user profile.

We have tested two different ways to create domain models, either with manual relevant judgments or without them. We have shown that both strategies can be equally effective.

To determine the appropriate domain for a query, we have used automatic query classification. Our experiments showed that the automatic domain identification is as effective as manual identification, although it identifies different domains.

- **Combining multiple contexts:**

  In previous studies, usually only one type of context is considered. We argue that multiple contexts can be used and they are often complementary. Therefore, a general model is proposed to integrate all the contextual factors that we considered. We have shown that the complete model outperforms any partial model. This clearly shows that it is beneficial to take into account the contextual factors as much as possible.

- **Language modeling framework:**

  Our general model is created using language modeling framework. We have further extended the language modeling framework to integrate different contextual factors. This framework has proven to be flexible for the integration of different factors. It seems to be a suitable tool for integrating more contextual factors.

Overall, our study aimed at the consideration of contextual factors in IR operations. We have successfully shown that this is feasible and it can produce large impact on the retrieval effectiveness.

Limitations and future work:

This work has explored several aspects of context-sensitive IR. On each of the aspects, our approaches have some limitations:

- Different contextual factors are considered to be independent. They are combined using a simple interpolation. In reality, contextual factors may interfere with each other. It would be interesting to investigate other ways to combine different contextual factors;
- Term relations are also considered separately. When we apply one term relation, the application is independent from other relations. In addition, we

assume that all the relations are consistent and no contradiction can be encountered. In practice, situations may present where different relations become contradictory. Non-monotonic reasoning may be required to deal with this problem. This situation has been investigated by Lau et al. [53]. However, it is still unclear whether and how their approach can be applied on larger dataset. More methods are still needed to solve this problem in the future;

- We have used topic domains to replace user profile. In reality, domain and user profile can be used together. For example, a user profile can indicate the importance of each domain for the user, according to his interests. Then given a query, it can be classified into one of the domains in the user profile. In this way, we can take advantage of domain models to apply the appropriate terms to a particular query. On the other hand, we also have a better idea about the preferred topic domains of the user. This will allow query classification to better choose the domains for the user. In addition, the user's preferences cannot be always classified in terms of topic domains only. Many preferences are not related to topic domains, such as the preferred medium, document source, and so on. Therefore, other types of user models are required for them. The approaches using a general user profile could be used to deal with some of these problems by creating a unique user profile to reflect non-classified user preferences and behavior;
- Our investigation is based on a language modeling framework. It is also limited by this. In particular, we only used unigram language models. This is a very limited model. Even though the limitation leads to a higher efficiency, in the future, it would be necessary to investigate the possibility to account for term dependencies. Traditional *n*-gram models do not seem to solve this problem, as [97] showed. A possible solution may lie in the integration of NLP techniques within the language modeling framework. For example,

strong noun phrases can be modeled with a higher-order language model, while weak statistical dependencies can be filtered out. This may reduce the problem of noise introduced by traditional *n*-gram models;

- We have used relatively simple parameter tuning methods. In particular, we have assumed that the mixture weights are the same for all the queries and query terms. In fact, the mixture weight may depend on the query terms. For example, if term relations do not cover the query terms, then lower mixture weight should be attributed to the knowledge model. It may be interesting to investigate more sophisticated training methods for term- and query-dependent setting of parameters;
- Finally, more contextual factors can be investigated and integrated into the same framework.

References

[1] Allen, B. L. (1991). Cognitive research in information science: implications for design. *Annual Review of Information Science and Technology*, 26: 3-37.

[2] Bai, J., Nie, J.-Y. and Paradis, F. (2004). Using language models for text classification. In *Proceeding of Asia Information Retrieval Symposium (AIRS'04)*, Beijing, China.

[3] Bai, J., Nie, J.-Y. and Cao, G. (2005). Integrating compound terms in Bayesian text classification. In *Proceeding of the 2005 IEEE/WIC/ACM International Joint Conference on Web Intelligence (WI'05)*, Compiègne, France.

[4] Bai, J., Song, D., Bruza, P., Nie, J.-Y. and Cao, G. (2005). Query expansion using term relationships in language models for information retrieval. In *Proceeding of the 14th ACM Conference on Information and Knowledge Management (CIKM'05)*, pp. 688-695, Bremen, Germany.

[5] Bai, J., Nie, J.-Y. and Cao, G. (2006). Context-dependent term relations for information retrieval. In *Proceeding of the Empirical Methods in Natural Language Processing (EMNLP'06)*, pp. 551-559, Sydney, Australia.

[6] Bai, J., Nie, J.-Y., Bouchard, H. and Cao, G. (2007). Using query contexts in information retrieval. In *Proceeding of the 30th ACM Conference on Research and Development in Information Retrieval (SIGIR'07)*, pp.15-22, Amsterdam, Netherlands.

[7] Barry, C. (1993). The identification of user criteria of relevance and document characteristics: beyond the topical approach to information retrieval. *Unpublished doctoral dissertation*, Syracuse University, NY.

[8] Barry, C. and Schamber, L. (1995). User-defined relevance criteria: a comparison of two studies. In *Proceedings of the 58th Annual Meeting of the American Society for Information Science*, 32: 103-111.

[9] Belkin, N. J. (1980). Anomalous states of knowledge as a basis for information retrieval. *Canadian Journal of Information Science*, 5: 133-143.

[10] Berger, A. and Lafferty, J. (1999). Information retrieval as statistical translation. In *Proceeding of ACM-SIGIR'99*, pp. 222-229.

[11] Blair, D. C. (1990). Language and representation in information retrieval, Elsevier.

[12] Bouchard, H. and Nie, J.-Y. (2006). Modèles de langue appliqués à la recherche d'information contextuelle. In *Proceeding of Conférence en Recherche d'Information et Applications (CORIA'06)*, Lyon.

[13] Brooks, T. A. (2004). The nature of meaning in the age of Google. *Information Research*, 9(3) paper 180.

[14] Bruza, P. and Song, D. (2002). Inferring query models by computing information flow. In *Proceeding of CIKM'02*, pp. 260-269.

[15] Cao, G., Nie, J.-Y. and Bai, J. (2005). Integrating word relationships into language models. In *Proceeding of ACM-SIGIR'05*, pp. 298-305, Salvador, Brazil.

[16] Cao, G., Si, L., Nie, J.-Y. and Bai, J. (2007). Learning to rank documents for ad-hoc retrieval with regularized models. In *Workshop on Learning to Rank for Information Retrieval, ACM-SIGIR'07*, Amsterdam, Netherlands.

[17] Chen S. F. and Goodman J. T. (1996). An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th Annual Meeting of the ACL*, pp. 310-318, Santa Cruz, California.

[18] Chirita, P. A., Paiu, R., Nejdl, W. and Kohlschütter, C. (2005). Using ODP metadata to personalize search. In *Proceeding of ACM-SIGIR'05*, pp. 178-185.

[19] Church, K. W. and Hanks, P. (1989). Word association norms, mutual information, and lexicography. In *Proceeding of ACL'89*, pp. 22-29.

[20] Cooper, William S. (1971). A definition of relevance for information retrieval. *Information Storage and Retrieval*, 7(1): 19-37.

[21] Croft, W. B. (1986). User-specified domain knowledge for document retrieval. In *Proceeding of ACM-SIGIR '86*. pp. 201-206.

[22] Croft, W. B., Cronen-Townsend, S. and Lavrenko, V. (2006). Relevance feedback and personalization: a language modeling perspective. In *the DELOS-NSF Workshop on Personalization and Recommender Systems Digital Libraries*, pp. 49-54.

[23] Croft, W. B. and Wei, X. (2005). Context-based topic models for query modification, *CIIR Technical Report*, University of Massachusetts.

[24] Cui, H. Wen, J.-R. Nie, J.-Y. Ma and W.-Y. (2003). Query expansion by mining user logs, *IEEE Transactions on Knowledge and Data Engineering*, 15(4): 829-839.

[25] Dempster, A. P., Laird, N. M. and Rubin, D. B. (1997). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39: 1-38.

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

[27] Dumais, S., Cutrell, E., Cadiz, J., Jancke, G., Sarin, R. and Robbins, D. C. (2003). Stuff I've seen: a system for personal information retrieval and re-use, In *Proceeding of ACM-SIGIR'03*, pp. 72-79.

[28] Eisenberg, M., and Schamber, L. (1988). Relevance: the search for a definition. In *Proceeding of ASIS'88*, pp. 164-168.

[29] Fagan, J. L. (1987). Experiments in automatic phrase indexing for document retrieval: a comparison of syntactic and non-syntactic methods. *PhD dissertation*, Department of Computer Science, Cornell University.

[30] Froehlich, T. J. (1994). Relevance reconsidered - Towards an agenda for the 21st century: introduction to special topic issue on relevance research. *Journal of the American Society for Information Science*, 45(3): 124 – 134.

[31] Fuhr, N. (1992). Probabilistic models in information retrieval, *The Computer Journal*, 35(3): 243-255.

[32] Gao, J., Nie, J.-Y., Wu, G. and Cao, G. (2004). Dependence language model for information retrieval. In *Proceeding of ACM-SIGIR'04*, pp. 170-177.

[33] Gao, J., Qi, H., Xia, X. and Nie, J.-Y. (2005). Linear discriminative model for information retrieval. In *Proceeding of ACM-SIGIR'05*, pp. 290-297.

[34] Gauch, S., Wang, J., and Rachakonda, S. M. (1999). A corpus analysis approach for automatic query expansion and its extension to multiple databases. *ACM Transaction of Information System* 17(3): 250-269.

[35] Goodman, J. (2001). A bit of progress in language modeling, *Computer Speech and Language*, October 2001, pp. 403-434.

[36] Goole Personalized Search, http://www.google.com/psearch.

[37] Grefenstette, G. (1992). Use of syntactic context to produce term association lists for text retrieval. In *Proceedings of the 15th ACM Conference on Research and Development in Information Retrieval (SIGIR'92)*, pp. 89–97.

[38] Grefenstette, G. (1994). Corpus-derived first-, second- and third order word affinities, *EURALEX*.

[39] Gulli A. and Signorini A. (2005). The indexable web is more than 11.5 billion pages. In *Proceedings of the 14th International World Wide Web Conference (WWW'05)*, pp. 902-903, Chiba, Japan.

[40] Harman, D. (1993). Overview of the second text retrieval conference (TREC-2). *NIST Special Publication 500-215: The Second Text Retrieval Conference (TREC 2).*

[41] Harman, D. (1994). Overview of the third text retrieval conference (TREC-3). *NIST Special Publication 500-226: The Third Text Retrieval Conference (TREC 3).*

[42] Harper, D. J. and van Rijsbergen, C. J. (1978). An evaluation of feedback in document retrieval using co-occurrence data. *Journal of Documentation*, 34(3): 189-216.

[43] Hiemstra D. (1998). A linguistically motivated probabilistic model of information retrieval. In *Proceeding of ECDL'98*, pp. 569-584

[44] Hipp, J., Guntzer, U. and Nakhaeizadeh, G. (2000). Algorithms for association rule mining: a general survey and comparison. In *Proceeding of SIGKDD'00 Explorations*, 2(1): 58-64.

[45] Hofmann, Thomas (1999). Probabilistic latent semantic indexing. In *Proceeding of ACM-SIGIR*'99, pp. 50-57.

[46] Jansen, B., Spink, A., and Saracevic, T. (2000). Real life, real users, and real needs: a study and analysis of user queries on the web. *Information Processing and Management.* 36(2): 207-227.

[47] Jing, Y. and Croft, W. B. (1994). An association thesaurus for information retrieval. In *Proceeding of RIAO'94*, pp. 146-160.

[48] Kim, H.-R. and Chan, P. K. (2005). Personalized ranking of search results with learned user interest hierarchies from bookmarks. In *Proceeding of WEBKDD'05 Workshop* at *ACM-KDD*, pp. 32-43.

[49] Kraft, D. H. and Buell, D. A. (1983). Fuzzy sets and generalized Boolean retrieval systems. *International Journal on Man-Machine Studies*, 19: 49-56.

[50] Lafferty, J. and Zhai, C. X. (2001). Document language models, query models, and risk minimization for information retrieval. In *Proceeding of ACM-SIGIR'01*, pp. 111-119.

[51] Lalmas, M. Crestani, F. van Rijsbergen, C. J. (eds.) (1998). *Information retrieval, uncertainty and logics*. Kluwer Academic Publishers.

[52] Lavrenko, V. and Croft, W. B. (2001). Relevance-based language models. In *Proceeding of ACM-SIGIR'01*, pp. 120-127.

[53] Lau, R., Bruza, P. and Song, D. (2004). Belief revision for adaptive information retrieval. In *Proceeding of ACM-SIGIR'04*, pp. 130-137.

[54] Lawley, K., Soergel, D. and Huang, X. (2005). Relevance criteria used by teachers in selecting oral history materials. In *Proceedings of the Annual Meeting of the American Society for Information Science and Technology*, 42, pp. 421-448.

[55] Lesk, M. (1986). Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In *Proceedings of the 5th Annual international Conference on Systems Documentation,* pp. 24-26, Toronto, Canada.

[56] Lewis, D. (1973). *Counterfactuals*. Oxford: Blackwell.

[57] Lin, J. (2006). The role of knowledge in conceptual retrieval: a study in the domain of clinical medicine. In *Proceeding of ACM-SIGIR'06*, pp. 99-106.

[58] Liu, F., Yu, C. and Meng, W. (2002). Personalized web search by mapping user queries to categories. In *Proceeding of CIKM'02,* pp. 558-565.

[59] Liu, X. and Croft, W. B. (2004). Cluster-based retrieval using language models. In *Proceeding of ACM-SIGIR'04*, pp. 186-193.

[60] Luhn, H. P. (1958). The automatic creation of literature abstracts. *IBM Journal of Research and Development*, 2, pp. 159-165.

[61] Miller George A. (1995). WordNet: A lexical database for English. *Communication of the ACM* 38(11): 39-41.

[62] Ng, K. (1999). A maximum likelihood ratio information retrieval model. In *Proceeding of TREC8*, pp. 483-482.

[63] Och, Franz. (2003). Minimum error rate training in statistical machine translation. In *Proceeding of ACL'03*, pp. 160-167.

[64] Pantel P. and Lin. D. (2001). A statistical corpus-based term extractor. In Stroulia, E. and Matwin, S. (Eds.) *AI 2001, Lecture Notes in Artificial Intelligence*, pp. 36-46, Springer-Verlag.

[65] Park, T. K. (1994). Toward a theory of user-based relevance: a call for a new paradigm of inquiry. *Journal of the American Society for Information Science*, 45: 135-141.

[66] Peat, H. J. and Willett. P. (1991). The limitations of term co-occurrence data for query, *Journal of the American Society for Information Science*, 42(5): 378-383.

[67] Peng, F., Schuurmans, D. and Wang, S. (2004). Augmenting Naive Bayes classifiers with statistical language models. *Information Retrieval,* 7(3-4): 317-345.

[68] Pitkow, J., Schütze, H., Cass, T., Cooley, R., Turnbull, D., Edmonds, A., Adar, E. and Breuel, T. (2002). Personalized search, *Communications of ACM*, 45: 50-55.

[69] Ponte, J. M. and Croft, W. B. (1998). A language modeling approach to information retrieval. In *Proceeding of ACM-SIGIR'98,* pp. 275-281.

[70] Porter, M. F. (1980). An algorithm for suffix stripping. *Program*, 14(3): 130-137.

[71] Pôssas, B., Ziviani, N., Meira, W. and Ribeiro-Neto, B. (2002). Set-based model: a new approach for information retrieval. In *Proceeding of ACM-SIGIR'02,* pp. 230-237.

[72] Press, W. H., Teukolsky, S. A., Vetterling, W. T. and Flannery, B. P. (1992). *Numerical recipes in C: the art of scientific computing*. New York, Cambridge University Press.

[73] Qiu, Y. and Frei, H. P. (1993). Concept based query expansion. In *Proceeding of ACM-SIGIR'93,* pp.160-169.

[74] Radecki, T. (1979). Fuzzy set theoretical approach to document retrieval. *Information Processing and Management,* 15: 247-259.

[75] Robertson, S. E. (1977). The probability ranking principle in IR. *Journal of Documentation*, 33: 294-304.

[76] Robertson, S. E. and Sparck Jones, K. (1976). Relevance weighting of search terms, *Journal of the American Society for Information Science*, 27: 129-146.

[77] Robertson, S. E., Walker, S., Jones, S., Hancock-Beaulieu, M. and Gatford, M. Payne, A. (1994). Okapi at TREC-3. In *Proceedings of the Third Text REtrieval Conference (TREC 1994)*, pp. 73-96.

[78] Salton, G. (1971). The SMART retrieval system: experiments in automatic document processing. Prentice-Hall.

[79] Salton, G., McGill, M. (1983). Introduction to modern information retrieval. McGraw-Hill.

[80] Salton, G., Fox, E. A. and Wu, H. (1983). Extended Boolean information retrieval. *Communications of the ACM*, 26(12): 1022-1036.

[81] Sanderson, M. (1994). Word sense disambiguation and information retrieval. In *Proceeding of ACM-SIGIR'94*, pp. 142-151.

[82] Sanderson, M. (2000). Retrieving with good sense. *Information Retrieval,* 2(1): 49-69.

[83] Saracevic, T. (1975). Relevance: a review of and a framework for the thinking on the notion in information science. *Journal of the American Society for Information Science*, 26(6): 326-343.

[84] Saracevic, T. (1996). Relevance reconsidered. In *Proceedings of the Conference on Conceptions of Library and Information Science (COLIS'96),* pp. 201-218, Copenhagen.

[85] Savoy, J. (1999). A stemming procedure and stopword list for general French corpora. *Journal of the American Society for Information Science*, 50(10): 944-952.

[86] Schamber, L., Eisenberg, M. B. and Nilan, M. S. (1990). A re-examination of relevance: towards a dynamic, situational definition. *Information Processing and Management*, 26(6): 755-774.

[87] Schamber, L. (1994). Relevance and information behavior. *Annual Review of Information Science and Technology*, 29: 3-48.

[88] Schütze, H., Pedersen J. O. (1997). A co-occurrence-based thesaurus and two applications to information retrieval. *Information Processing and Management*, 33(3): 307-318.

[89] Shen, D., Pan, R., Sun, J.-T., Pan, J. J., Wu, K., Yin, J. and Yang, Q. (2006). Query enrichment for web-query classification. *ACM-TOIS*, 24(3): 320-352.

[90] Shen, X., Tan, B. and Zhai, C. X. (2005). Context-sensitive information retrieval using implicit feedback. In *Proceeding of ACM-SIGIR'05,* pp. 43-50.

[91] Shin, K. and Han S.-Y. (2004). Improving information retrieval in MEDLINE by modulating MeSH term weights. In *Proceeding of the 9th International Conference on Applications of Natural Language to Information Systems, NLDB*, pp. 388-394.

[92] Silverstein, C., Henzinger, M., Marais, H. and Moricz, M. (1999). Analysis of a very large Web search engine query log. In *ACM-SIGIR Forum*, 33(1): 6-12.

[93] Sinclair, J. (1991). Corpus, concordance, collocation. Oxford University Press: Oxford.

[94] Smadja, F. (1993). Retrieving collocations from text: Xtract. *Computational Linguistics*, 19(1): 143-178.

[95] Smeaton, A. F., van Rijsbergen, C. J. (1983). The retrieval effectiveness of query expansion on a feedback document retrieval system, *Computer Journal*, 26: 239-246.

[96] Song, D. and Bruza, P. (2001). Discovering information flow using a high dimensional conceptual space. In *Proceeding of ACM-SIGIR'01*, pp. 327-333.

[97] Song, F. and Croft, W. B. (1999). A general language model for information retrieval. In *Proceedings of 8th ACM Conference on Information and Knowledge Management (CIKM'99)*, pp. 279-280.

[98] Sparck Jones, K., (1972). A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28: 111-121.

[99] Srikanth, M. and Srihari, R. (2002). Biterm language models for document retrieval. In *Proceeding of ACM-SIGIR '02*, pp. 425-426.

[100] Sundin, O. and Johannisson, J. (2005). The instrumentality of information needs and relevance. In *Proceeding of the 5th International Conference on Conceptions of Library and Information Sciences (CoLIS'05)*, Glasgow.

[101] Teevan, J., Dumais, S. T. and Horvitz, E. (2005). Personalizing search via automated analysis of interests and activities. In *Proceeding of ACM-SIGIR'05*, pp. 449-456.

[102] Van Rijsbergen, C. J. (1977). A theoretical basis for use of co-occurrence data in information retrieval. *Journal of Documentation*, 33(2): 106-119.

[103] Van Rijsbergen, C. J. (1979). *Information Retrieval*. Butterworths.

[104] Van Rijsbergen, C. J. (1986). A new theoretical framework for information retrieval. In *Proceeding of ACM-SIGIR'86*, pp. 194-200.

[105] Voorhees, E. M. (1993). Using WordNet to disambiguate word senses for text retrieval. In *Proceeding of ACM-SIGIR'93*, pp. 171-180.

[106] Voorhees, E. M. (1994). Query expansion using lexical-semantic relations. In *Proceeding of ACM-SIGIR'94*, pp. 61-69.

[107] Wang, P. and Soergel, D. (1998). A cognitive model of document use during a research project: Study I. document selection. *Journal of the American Society for Information Science and Technology*, 49(2): 115-133.

[108] Wei, Xing and Croft, W. B. (2007). Investigating retrieval performance with manually-built topic models. In *Proceeding of RIAO'07*, Pittsburg, US.

[109] Wilson, T. D. (1981). On user studies and information needs. *Journal of Librarianship,* 37(1): 3-15.

[110] Wolfram, D. (1999). Term co-occurrence in Internet search engine queries: an analysis of the Excite data set. *Canadian Journal of Information and Library Science*. 24(2/3): 12-33.

[111] Xu, J. and Croft, W. B. (1996). Query expansion using local and global document analysis. In *Proceeding of ACM-SIGIR'96,* pp. 4-11.

[112] Xu, J. L. (1999). Internet search engines: real world IR issues and challenges. *Conference on Information and Knowledge Management*, Kansas City, Missouri.

[113] Yarowsky, D. (1995). Unsupervised word sense disambiguation rivaling supervised methods. In *Proceeding of ACL'95*, pp. 189-196.

[114] Zhou X., Hu X., Zhang X., Lin X. and Song I.-Y. (2006). Context-sensitive semantic smoothing for the language modeling approach to genomic IR. In *Proceeding of ACM-SIGIR'06*, pp. 170-177.

[115] Zhai, C. X. and Lafferty, J. (2001). Model-based feedback in the language modeling approach to information retrieval. In *Proceeding of CIKM'01,* pp. 403-410.

[116] Zhai, C. X. and Lafferty, J. (2001). A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceeding of ACM-SIGIR'01*, pp. 334-342.

[117] Zhai, C. X. and Lafferty, J. (2001). Dual role of smoothing in the language modeling approach. In *Proceedings of the Workshop on Language Models for Information Retrieval (LMIR'01).*

[118] Zhang, X., Anghelescu, H. G. B. and Yuan, X. (2005). Domain knowledge, search behavior, and search effectiveness of engineering and science students: an exploratory study. *Information Research*, 10(2), paper 217.

Appendix 1

Derivation of Updating Function in EM

The log-likelihood of the feedback documents that we want to maximize is:

$$LL(FB) = \sum_{D \in FB} \sum_{t \in V} tf(t,D)[\sum_{i \in X} \alpha_i P_i(t \mid \theta_Q^i) + \alpha_C P_i(t \mid \theta_C)]$$
$$= \sum_{D \in FB} \sum_{t \in V} tf(t,D) \sum_{i \in X'} \alpha_i P_i(t \mid \theta_Q^i)$$

where $X' = X \cup \{C\} = \{0, K, Dom, FB, C\}$, α_i is the new parameter value and $\sum_{i \in X'} \alpha_i = 1$.

Let us express the above $LL(FB)$ as a function of the mixture weights $\overline{\alpha} = \langle \alpha_0, \alpha_K, \alpha_{Dom}, \alpha_{FB}, \alpha_C \rangle$

$$L(FB, \overline{a}) = \sum_{D \in FB} \sum_{t \in V} tf(t,D) \log \sum_{i \in X'} \alpha_i P_i(t \mid \theta_Q^i)$$

Adding a constraint on $\sum_{i \in X'} \alpha_i = 1$, we can get:

$$\propto \sum_{D \in FB} \sum_{t \in V} tf(t,D) \log \frac{\sum_{i \in X'} \alpha_i P_i(t \mid \theta_Q^i) \frac{\alpha_i' P_i(t \mid \theta_Q^i)}{\alpha_i' P_i(t \mid \theta_Q^i)}}{\sum_{i \in X'} \alpha_i' P_i(t \mid \theta_Q^i)} - \beta(\sum_{i \in X'} \alpha_i - 1)$$

$$= \sum_{D \in FB} \sum_{t \in V} tf(t,D) \log \sum_{i \in X'} \frac{\alpha_i' P_i(t \mid \theta_Q^i)}{\sum_{i \in X'} \alpha_i' P_i(t \mid \theta_Q^i)} \frac{\alpha_i P_i(t \mid \theta_Q^i)}{\alpha_i' P_i(t \mid \theta_Q^i)} - \beta(\sum_{i \in X'} \alpha_i - 1)$$

where α_i' is the previous parameter value. Then we can use *Jesen* inequality here:

$$\geq \sum_{D \in FB} \sum_{t \in V} tf(t,D) \sum_{i \in X'} \frac{\alpha_i' P_i(t \mid \theta_Q^i)}{\sum_{i \in X'} \alpha_i' P_i(t \mid \theta_Q^i)} \log \frac{\alpha_i}{\alpha_i'} - \beta(\sum_{i \in X'} \alpha_i - 1)$$

$$= \Delta(\overline{\alpha}, \overline{\alpha}')$$

Making the partial derivation and make it equal to 0, and let:

$$Z_{i,t} = \frac{\alpha_i' P_i(t \mid \theta_Q^i)}{\sum_{i \in X'} \alpha_i' P_i(t \mid \theta_Q^i)}$$

$$\frac{\partial \Delta(\overline{\alpha}, \overline{\alpha}')}{\partial \alpha_i} = \sum_{D \in FB} \sum_{t \in V} tf(t,D) Z_{i,t} \frac{1}{\alpha_i} - \beta = 0$$

If we let:

$$\beta = \sum_{D \in FB} \sum_{t \in V} tf(t,D)$$

Then we derive our updating function as follows:

E-step: $$Z_{i,t} = \frac{\alpha_i' P_i(t \mid \theta_Q^i)}{\sum_{i \in X'} \alpha_i' P_i(t \mid \theta_Q^i)}$$

M-step: $$\alpha_i = \frac{\sum_{D \in FB} \sum_{t \in V} tf(t,D) Z_{i,t}}{\sum_{D \in FB} \sum_{t \in V} tf(t,D)}$$