Fall 2014

SEARCHING BASED ON QUERY DOCUMENTS

Youngho Kim
University of Massachusetts Amherst

SEARCHING BASED ON QUERY DOCUMENTS

A Dissertation Presented

by

YOUNGHO KIM

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

September 2014

School of Computer Science

SEARCHING BASED ON QUERY DOCUMENTS

A Dissertation Presented

by

YOUNGHO KIM

Approved as to style and content by:

W. Bruce Croft, Chair

James Allan, Member

Benjamin Marlin, Member

Daeyoung Kim, Member

Lori A. Clarke, Chair
School of Computer Science

To those whom I love and those who love me.

ACKNOWLEDGMENTS

Completing this thesis, I appreciate the contributions from various people such as professors, colleagues, researchers, and friends. Among them, I especially thank my advisor, Professor W. Bruce Croft. Without his effort, I would not have been successful in my doctoral study. As many his former students said, Bruce is the greatest advisor and the most innovative researcher, who provided insightful suggestions for my research and inspired me to devise creative solutions. Moreover, I have learned the methods for thinking deeply, approaching to core problems, and elaborating ideas. The lessons from him would be the basis to improve my future researches. Again, I appreciate his guidance and contribution for my Ph.D.

In addition, I really like to thank the committee members, Professor James Allan, Ben Marlin, and Daeyoung Kim. Their comments on my thesis are precious and have really enhanced my work. I also thank Professor R. Manmatha who has helped to initiate this work and served for the proposal.

I also thank the fellows in CIIR; Jangwon Seo, Jin-young Kim, Jae-hyun Park, Myung-ha Jang, Xiabing Xue, Michael Bendersky, Henry Feild, Sam Huston, Mostafa Keikha, Jeff Dalton, Van Dang, Elif Aktolga, Xing Yi, Shiri Dori-Hacohen, Jiepu Jiang, I. Zeki Yalniz, Chia-Jung Lee, Weize Kong, and others.

And thanks to all the friends in UMass, e.g., Yeon-sup Lim and Junghee Jo.

This work was supported in part by the Center for Intelligent Information Retrieval, in part by ARRA NSF IIS-9014442, in part by NSF CLUE IIS-0844226, and in part by NSF grant ♯IIS-0534383. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

ABSTRACT

SEARCHING BASED ON QUERY DOCUMENTS

SEPTEMBER 2014

YOUNGHO KIM

B.Sc., INHA UNIVERSITY, INCHEON, SOUTH KOREA

M.Sc., KOREA ADVANCED INSTITUTE OF SCIENCE AND TECHNOLOGY,

DAEJEON, SOUTH KOREA

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Advised by: Professor W. Bruce Croft

Searches can start with query documents where search queries are formulated based on document-level descriptions. This type of searches is more common in domain-specific search environments. For example, in patent retrieval, one major search task is finding relevant information for new (query) patents, and search queries are generated from the query patents One unique characteristic of this search is that the search process can take longer and be more comprehensive, compared to general web search. As an example, to complete a single patent retrieval task, a typical user may generate 15 queries and examine more than 100 retrieved documents. In these search environments, searchers need to formulate multiple queries based on query documents that are typically complex and difficult to understand.

In this work, we describe methods for automatically generating queries and diversifying search results based on query documents, which can be used for query

suggestion and for improving the quality of retrieval results. In particular, we focus on resolving three main issues related to query document-based searches: (1) query generation, (2) query suggestion and formulation, and (3) search result diversification. Automatic query generation helps users by reducing the burden of formulating queries from query documents. Using generated queries as suggestions is investigated as a method of presenting alternative queries. Search result diversification is important in domain-specific search because of the nature of the query documents. Since query documents generally contain long complex descriptions, diverse query topics can be identified, and a range of relevant documents can be found that are related to these diverse topics.

The proposed methods we study in this thesis explicitly address these three issues. To solve the query generation issue, we use binary decision trees to generate effective Boolean queries and labeling propagation to formulate more effective phrasal-concept queries. In order to diversify search results, we propose two different approaches: query-side and result-level diversification. To generate diverse queries, we identify important topics from query documents and generate queries based on the identified topics. For result-level diversification, we extract query topics from query documents, and apply state-of-the-art diversification algorithms based on the extracted topics. In addition, we devise query suggestion techniques for each query generation method.

To demonstrate the effectiveness of our approach, we conduct experiments for various domain-specific search tasks, and devise appropriate evaluation measures for domain-specific search environments.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure **Page**

CHAPTER 1

INTRODUCTION

1.1 Motivation

Searches in specific domains are very different from general web search. Domain-specific searches (e.g., patent retrieval, legal search, academic literature search and medical information retrieval) have very specific search requirements and environments, and sometimes well-known Information Retrieval (IR) techniques, proven effective for web search, are not successful. For example, in patent retrieval, typical query expansion methods (e.g.,[64, 82]) are less effective [38]. To successfully accomplish search tasks in these domains, search techniques should be designed for their unique search characteristics.

One unique characteristic of domain-specific IR is that searching based on query documents is more common. That is, search queries are typically formulated based on document-level descriptions (i.e., query documents). As an example, prior-art search [36] involves finding past patents that may conflict with new patent applications; in academic literature search [10], academic authors need to find relevant research papers that should be cited in their work. In these search tasks, users need to formulate search queries, after reading query documents, e.g., patent examiners generate queries from a new patent to validate its patentability. However, formulating effective queries is a significant burden for users because query documents are quite long and difficult to understand. In patent retrieval, for example, a patent document contains 3,900 terms on average [48]. In addition, to protect the invention and extend the coverage, the content in a patent document is complex, and the au-

thors tend to use ambiguous words and expressions [107]. Note that in this thesis, we define domain-specific IR or domain-specific search as the search tasks based on query documents. Since searching based on query documents is more common and typical in many domain-specific IR tasks, which is not easily observed in web search environments, we conveniently use the terms domain-specific IR and domain-specific search to represent query documents-based searches.

Another typical characteristic of domain-specific IR is that the search process can take much longer and be more comprehensive. To complete a single task, more queries are used and more search results are examined. According to [53], patent examiners generally spend about 12 hours to complete an invalidity task by examining approximately 100 patent documents retrieved by 15 different queries on average, whereas in web search, people use fewer queries and examine only a few retrieved documents; 81.3% of web search users only issue a single query in a search session, and 44.5% of them examine only one retrieved document [111].

Given that users in domain-specific searches need to formulate more search queries from long documents and examine more retrieval results, we propose to reduce the difficulty of formulating queries and improve the quality of retrieval results by studying automatic query generation and search result diversification based on query documents.

1.2 Issues

In this work, we propose methods to automatically generate queries and diversify search results based on query documents. To improve the effectiveness of the proposed methods, we focus on resolving three main issues related to domain-specific searches: (1) query generation, (2) query suggestion and formulation, and (3) search result diversification. These issues are crucial for improving the search quality of domain-specific IR, and the proposed methods address them explicitly.

First, we propose automatic query generation methods for domain-specific searches, which help users by reducing the burden of formulating queries from query documents. In prior-art search, there has been some previous work on generating effective queries from query patents (i.e., query documents) (e.g., [9, 38, 75, 106]). Since using full texts of query patents as queries is less effective, to generate effective queries, these studies have selected top ranked terms (predicted to be effective) extracted from query patents. However, most of this work can be only applied to patent search environments because patent-specific structures (e.g., International Patent Classification (IPC) codes [75] and patent section information (such as claim, background, and summary) [9, 106]) are a significant part of the generation. In our work, we assume more general settings where only query documents are provided so that proposed methods can be adopted in any domains of interest.

Second, we need to consider the suggestion of generated queries and their representation. In previous work, query generation techniques are mainly designed for generating effective queries which make the search of relevant documents more efficient. To maximize the effectiveness of queries, long and complex weighted queries are typically generated (e.g., *#weight(0.1099 parse 0.1085 dependency 0.0431 label 0.0321 arc 0.0186 head ...)*). This approach can be effective if generated queries are automatically executed for retrieval. However, such complex queries can be less useful when query suggestion is required in domain-specific search environments. For example, Tseng and Wu [98] indicated that the provision of suggested search vocabulary would be helpful in patent search. In domain-specific search tasks, many users are search professionals (e.g., patent examiners in prior-art search) and they need to manipulate search queries to retrieve more relevant documents. In addition, they need to identify how search results are obtained. Thus, we assume that generated queries can be examined before retrieval (i.e., query suggestion), and this setting may be more practical to help domain-specific search users (we discuss more about query

suggestion in Chapter 1.4). In addition to this, professional users often prefer specific forms of queries that are particularly useful for their search tasks. For example, Boolean queries (e.g., $\{signal \wedge analog\}$) are common in patent retrieval, and many patent engineers expect to use Boolean operators in their queries [53] because of their ease of manipulation and self-documentation in that they define precisely the set of documents that are retrieved. Another example is that phrasal-concept queries (e.g., {*"structural paraphrase generation" "large corpora" "multiple sequence alignment"*}) are necessary in academic literature search because phrasal concepts are frequently used in academic papers and natural to users (e.g., academic authors).

Third, diversifying search results is important for domain-specific IR. In general, search result diversification is the process of re-ordering initial retrieval results so that the final ranked list can include more diverse aspects (or topics) associated with the query. In web search, this technique is adopted for clarifying vague information needs, e.g., a web query "slate"' can represent one of a broad range of topics. However, in domain-specific searches, we attempt to improve retrieval effectiveness by covering more of the topics described in query documents. For example, in prior-art search, a query patent is very long and includes complex structure [32]. In that structure, diverse claims are specified, and background patents related to the application are described. In addition, patent applications can describe multiple components. Thus, we can find a range of topics in a query patent, and the relevant documents can pertain to some or all of these topics.

Table 1.1 shows an example query patent and its relevant documents. In this example, the patent application describes several important topics such as *usage profile*, *BIOS*, *operating system*, etc. We can group similar relevant documents pertaining to each topic. For example, R1 and R2 are related to a topic *BIOS*, whereas R3 and R4 refers to *operating system*. In addition, R5 describes a method for controlling network access, which relate to another query topic, i.e., *profile server*. Based on these topics,

Table 1.1: Query Patent Example

Query Patent		
Title: Method and apparatus for providing content on a computer system based on usage profile. **Abstract:** A method and apparatus for determining a *computer system usage profile*. A *basic input output system (BIOS) module* and/or an *operating system module* obtain computer system usage profile information by tracking events such as the frequency of re-boots, the time required to boot-up and shut-down the operating system ... data is collected and communicated to a *profile server* ...		
List of Relevant Documents		
No.	**Title**	**Topic**
R1	Extended *BIOS* adapted to establish remote communication for diagnostics and repair.	*BIOS*
R2	Extended *BIOS* emulation of a hard file image as a diskette.	
R3	*Operating System* architecture with reserved memory space resident program code identified in file system name space.	*Operating System*
R4	Method for loading an *Operating System* through a network.	
R5	Method and apparatus for controlling network and workstation access.	*Profile Server*
...	...	...

the retrieval result can be diversified, meaning that the ranked documents can be optimized to cover the range of topics. Accordingly, from such a diversified search result, the users can easily retrieve relevant documents.

1.3 Query Generation Frameworks

All of the issues described in the previous section need to be resolved for improving users' experience in domain-specific searches. With this in mind, we introduce our approaches to automatic query generation based on query documents. We address the issues by proposing three different query generation models as follows.

We first propose a method of generating effective Boolean queries, described in [58]. For a number of reasons, Boolean queries are preferred in professional search (where search users are search professionals, e.g., patent examiners and information

specialists in companies and law firms). For example, in patent search, previous work [53] revealed that the use of Boolean operators is one of the most important features to formulate effective queries from the perspective of patent professionals. In our model, we generate Boolean queries by exploiting decision trees learned from pseudo-labeled documents, i.e., the top k documents initially retrieved by query documents. We learn a binary decision tree from the pseudo-relevant documents so that the decision tree could determine whether a document is pseudo-relevant or not. Then, each positive decision rule, i.e., a path from the root to a positive leaf node indicating pseudo-relevance, can be the basis of a Boolean query.

Another model that we propose is an automatic phrasal-concept query generation, using the domain of academic literature search [59]. Academic papers frequently use phrases to describe their key ideas, and search users (e.g., authors) are familiar with phrasal concepts. So, we test the assumption that queries generated using phrasal concepts would be more effective than other baseline methods. From a query document, we identify "key concepts" – more effective concepts for finding relevant documents – by using the labeling propagation algorithm [114] which propagates retrieval effectiveness (labels) of the baseline keyword query to associated candidate phrases. Note that the baseline query only contains (unigram) words and is generated by using previous query generation methods (e.g., [47]), whereas phrasal concepts consist of noun phrases longer than unigram words.

Lastly, we study the problem of diverse query generation based on query documents [57]. As described in the previous section, diversified search results that can cover multiple query topics can be useful for the users to ease retrieving relevant documents. To do this, we propose a query-side diversification method which generates multiple queries related to diverse query aspects. Specifically, we assume that a query document can include multiple query topics, and defined a "query topic" as a set of related terms from a query document. Given a query document, we identify n differ-

ent query topics (i.e., term sets) by applying term clustering algorithms (e.g., spectral clustering [101]) to the terms from the document. Afterwards, we learn decision trees by using n distinct sets of pseudo-relevant documents (each of which is obtained by the terms in a query topic), and decision trees to generate diverse queries relevant to the identified topics.

1.4 Query Suggestion Applications

Query suggestion is an effective and practical way to help users formulate queries [6, 54]. In a typical query suggestion process, a list of alternative queries is suggested to a user, after the user inputs an initial query [94]. Query suggestion has been widely discussed in papers and has become part of many commercial systems (e.g., [41, 78, 95]). Domain-specific IR has begun to adopt query suggestion techniques (e.g., [58, 70]). Following this, we generate multiple queries from query documents, and then select a reasonable number of effective queries as suggestions.

To effectively help users in domain-specific searches, query suggestion is an essential and promising application. First, given a large number of generated queries, users can only examine a relatively small number of queries (e.g., 5 to 10 queries), and selecting effective queries is particularly important. Second, we generate diverse queries, and are able to provide diverse suggestions, which can expedite searches. Emphasizing diverse query suggestion is important because otherwise the system may suggest multiple similar queries that would produce near-duplicate search results. In other words, suggesting similar queries can prevent searchers completing their search tasks efficiently, e.g., patent examiners waste time examining similar results instead of using additional queries that can retrieve other relevant patents. Therefore, in this work, we develop query suggestion applications based on the queries generated by our frameworks. To highlight the effectiveness of our query generation methods, we

evaluate the suggestion systems by comparing with other query suggestion methods (e.g., [14]).

1.5 Search Result Diversification Methods

In Chapter 1.2, we discussed the importance of diversifying search results for domain-specific IR. Given this motivation, we propose a result-level diversification method that re-ranks the documents in an initial retrieval to cover more query topics. Note that in Chapter 1.3 we proposed a query-side approach to the diversification problem, which is a somewhat indirect way to generate diverse search results, but in this section, we propose a result-level diversification method which directly manipulates search results for diversification. In this, query topics are first identified, and then re-ranking algorithms (e.g., PM-2 [29] and xQuAD [92]) are applied with the identified topics. Specifically, given a query document, we extract phrase-level topic vocabulary (as the basis for query topics) by ranking candidate phrases (extracted from a query patent) with respect to multiple ranking features (i.e., topicality, predictiveness [66], query clarity [27], relevance to query patents, cohesiveness, etc.). These features indicate how well candidate phrases can represent query topics. For example, topicality and predictiveness are effective features for finding topic terms of initial queries [28]. Then, we consider the top k phrases as topic phrases used for diversification. After generating topic phrases, we apply a state-of-the-art diversification algorithm that can optimize the document-level "diversity" in a final retrieval result. In this work, we choose to use the proportionality-based approach proposed in [29], which re-orders the documents with respect to the "popularity" of their topics in the initial ranking. Finally, diverse ranked results are produced, and the users can easily identify relevant documents based on the diverse results.

1.6 Summary

In this thesis, we explore effective techniques to improve domain-specific searches. As part of this, we propose general query generation frameworks for domain-specific searches. These frameworks include how to generate multiple queries in user-preferred forms, how to formulate more effective queries that can retrieve more relevant documents, how to identify query topics from query documents, and how to generate diverse queries relevant to the topics. In order to use the queries generated by our frameworks, we introduce query suggestion methods adapted to specific query formulations (e.g., Boolean queries and phrasal-concept queries). Furthermore, we devise a method to diversify search results. This method aims to re-rank initial retrieval results so that more diverse query topics are covered in the final rank results. In this method, we describe how to represent query topics from documents, how to identify effective query topics, and how to generate diverse search results based on the identified query topics. By proposing these methods, we attempt to resolve the three main issues raised to improve the search quality of domain-specific IR.

In our evaluations, we conduct experiments on various search domains, namely patent retrieval [36], academic literature search [15], and medical information retrieval [46]. To evaluate query generation frameworks, we employ state-of-the-art query generation methods (e.g., [47, 75, 80, 107]) as baselines to compare with our approach. In addition, we adopt state-of-the-art diversification methods (e.g., [28]) to verify the effectiveness of our diversification approach in domain-specific search environments.

1.7 Contributions

The contributions of our work can be summarized as follows.

- Evidence showing that domain-specific searches can be enhanced by resolving three issues: (1) query generation, (2) query suggestion and formulation, and (3) search result diversification.

- Methods to generate effective queries based on documents.
- Query formulation in user-preferred representations.
- Query-side diversification methods to generate diverse search results.
- Search result diversification frameworks applied to domain-specific searches.
- Algorithms to identify important topics (or aspects) from documents.

1.8 Organization

This thesis is organized as follows: Chapter 2 reviews previous work from a number of related research areas, and Chapter 3 describes evaluation settings including target domains that our methods are applied to, evaluation metrics adapted to domain-specific search environments, test collections (i.e., queries and relevance judgments), etc. From Chapter 4 to Chapter 7, we propose our query generation frameworks, suggestion methods, and diversification approaches; Chapter 4, Chapter 5, Chapter 6, and Chapter 7 describes Boolean query generation, phrasal-concept query generation, diverse query generation, and search result diversification, respectively. In each of these chapters, we provide experimental results and relevant discussion. In Chapter 8, we finally conclude this thesis by summarizing the results and discussing future work.

CHAPTER 2

RELATED WORK

2.1 Overview

Our work is related to a number of research areas: (1) Query Generation, (2) Query Expansion, (3) Query Suggestion, and (4) Search Result Diversification. In this chapter, we will describe prior research related to these areas to provide the background for the approaches proposed in this thesis. We start with a review of query generation approaches, that have mostly been focused on prior-art patent search. We then review significant work in query expansion since several query generation methods exploit pseudo-labeled documents to extract query terms. We also review existing query suggestion techniques and discuss their relevance to query suggestions for domain-specific searches. Research related to search diversification is also described.

2.2 Query Generation

Automatic query generation based on a query document is an important task to find relevant documents for domain-specific searches. Especially in patent prior-art search tasks, this technique is more important. Texts in patent documents are complex and difficult to understand because patent documents contain thousands of words and they intentionally use vague expressions to extend their coverage [48, 107]. So, a number of researches for automatic query generation have been proposed (e.g., [38, 75, 107]). These methods use the full texts of patent applications, and generate queries by ranking the terms in the query patents. For example, Xue and Croft [107] extracted query terms from the "brief summary" section of query patents by

tfidf scoring, and formulated queries by tf-based term weighting. Mahdabi et al. [75] used Kullback-Leibler divergence between query models (estimated from query patents) and collection models for term ranking. In addition to this, they extracted key phrases by tfidf and Mutual Information-based scoring, and expanded the initial term queries. Similar to this approach, Ganguly et al. [38] selected the top sentences ranked by similarity to pseudo-relevant documents for query patents. They first obtain pseudo-relevant documents retrieved by a query patent, and generate queries by selecting the sentences (recognized in the query patent) having more likelihood to the pseudo-relevant documents. In the TREC Chemical track [71], the task of finding relevant patents for new chemical patents was proposed, and among all participants, the approach proposed by Gobeill et al. [40] performed the best. They generated an effective query by identifying chemical concepts from chemical ontology (e.g., PubChem[1]). While these methods are specialized in patent search environments, in this work, we propose more generalizable approaches for generating effective queries. In addition, previous work assumed that generated queries are automatically executed in retrieval, and the generated queries can be complex (e.g., query term weighting is required) and long (e.g., 150 terms in [107]). However, in this work, we consider using generated queries for suggestion, and the users can examine the generated queries before retrieval. In this situation, long and complex queries are less effective.

Outside of patent retrieval, Lee and Croft [67] proposed a learning-based approach to generate queries based on user-selected passages in a (query) document. They assumed that users explicitly specify passages in a document, and extracted important chunks (e.g., noun phrases and named entities) by learning a CRF (Conditional Random Field) model. This model used textual features including web n-grams, query logs, Wikipedia titles, etc. However, these features would be less effective for the doc-

[1]The database of structure and description for chemical molecules (http://pubchem.ncbi.nlm.nih.gov).

uments in domain-specific searches, and our query documents are much longer than the passages extracted from short web documents. Accordingly, the graphical model could not perform effecctively. Smucker and Allan [93] proposed a similarity browsing tool for web retrieval, which could find similar documents to a given document. Given a short web query, they first generate initial retrieval results and for each retrieved document (i.e., query document), similar documents are searched. However, to find similar documents, they simply generated a long query by using the whole text of a query document or partial contexts of each initial query term in the query document. Since they concentrated on elaborating retrieval models, the methods to generate more effective queries were not investigated. Weng et al. [103] presented a document decomposition-based approach. They reduced the dimensionality of both query and target documents, and then used hashing algorithms for indexing and retrieving relevant documents. The application proposed by Yang et al. [108] can automate cross-referencing of online contents from different resources (i.e., news and blogs). From a query document (news), candidate phrases are extracted by calculating tfidf and Mutual Information, and then they score phrases using phrase association in the Wikipedia link graph. They use the top k scored phrases as queries to retrieve relevant blog posts. Although these studies were effective for retrieving similar documents, they are limited in that generating diverse queries was not considered.

2.3 Query Expansion

Another approach to generating effective queries exploits query expansion techniques (e.g., [9, 76, 37]). In general, automatic query expansion [85, 104] has been researched to bridge the gap between users' queries and relevant documents. In particular, pseudo-relevance feedback [90] has been known as one of the most effective techniques. Among many proposed methods for pseudo-relevance feedback (e.g., [63]), some studies are closely related to our work. One related study is the query expansion

method proposed by Mitra et al. [82], which addressed the effectiveness of Boolean filters to improve precision of automatic query expansion. In that, they manually formulated fuzzy Boolean operators (conjunction and disjunction) and selected expanded terms from a set of pseudo-relevant documents refined by the Boolean filters. However, their work is limited in that the Boolean filters are manually constructed while we focus on automatic formulation.

Another related work is concept-based query expansion techniques (e.g., [35, 80]). Xu and Croft [104] proposed local context analyses for query expansion. They used co-occurrence statistics to extract concepts (e.g., single terms and phrases) from passages, and expanded initial queries by them. Metzler and Croft [80] used latent concepts extracted from pseudo-relevant documents to expand short initial queries (e.g., "hubble telescope achievements"). They used the Markov Random Field (MRF) framework [79] to model the dependency between terms in the concepts. In addition, Fonseca et al. [35] also proposed a concept-based query expansion method. In [35], they viewed a past query in a query log as a concept, and past queries related to the current query were suggested to users for selecting more related concepts. One limitation of this study is that a sufficient amount of query log data, which are essential for their approach, cannot be easily acquired in typical small domain-specific search environments.

In addition, interactive expansion methods are related to our work. Kumaran and Allan [61] showed that a selective reduction or expansion of initial long queries can be effective for improving retrieval performance. To minimize interaction with users, they generated selective options (i.e., queries) by merging several effective subqueries (i.e., reduction) and expansion term sets. These options could generate highly overlapping search results of original queries. However, they assumed that the original query is only sentence-length (e.g., 10 or 20 terms) and all possible sub-queries were

examined, and this setting is not applicable to document-length initial queries (i.e., query documents).

In patent retrieval, standard query expansion techniques are less effective with initial queries that use full texts of query patents [38, 76]. Many query expansion methods (e.g., [64, 80, 104]) assume short web queries as initial queries and focus on improving early precision (more emphasized in web search). However, prior-art search tasks are recall-oriented, and their initial queries typically contain hundreds of terms. To alleviate these differences, several query expansion approaches designed for prior-art search environments have been proposed. For example, Ganguly et al. [37] used a decomposition-based approach for extracting expansion terms. In that, a text tiling technique [45] was applied for decomposing a query patent into sub-topic segments, each segment block was used for retrieving pseudo-relevant documents, and the pseudo-relevant documents were interleaved to produce a final ranking result. Mahdabi et al. [76] used term proximity information to identify expansion terms. Given a query patent, they first generate an initial query by taking claim terms, and then build a query-specific lexicon that includes the terms from the same IPC patents. Among many terms in the lexicon, they identify expansion terms by two adjacency operators used in patent examination (i.e., "ADJn" and "NEARn"). Although these expansion techniques are effective for prior-art search, expanded queries typically contain hundreds of terms, and are less useful as suggestions.

2.4 Query Suggestion

Query suggestion is an effective and practical way to help users formulate queries. In a typical suggestion process, a number of alternative queries are displayed to a user after an initial query is input [94]. For web search, there has been significant prior work on query suggestion. Methods for doing this typically rely on using query logs and clickthrough statistics (e.g., [6, 54, 78]) that are available in a web search

environment. For example, Jones et al. [54] proposed a query substitution system that suggests strongly related queries identified from query logs, and the query recommendation techniques proposed in [6] provide alternatives by clustering related queries in query logs. The most widely used technique is exploiting query-click graphs (e.g., [26, 78, 83]. In this approach, a bipartite graph is constructed where the set of vertices is partitioned into two sub-sets: queries and (clicked) documents, and each edge is defined by user's click information [26]. By performing a random walking on this bipartite graph, query similarity can be calculated, and more similar queries can be shown as suggestions. This approach is successful since suggested queries are extracted from query logs. However, such resources are not available in the domain-specific search environments that we focus on.

Without query logs and clickthrough statistics, only a few methods (e.g., [14, 70]) have been proposed for query suggestion, and these studies are strongly related to our work. Bhatia et al. [14] suggest relevant (n-gram) phrases for an initial query without query logs. They extract highly correlated n-grams with the partially input user query, i.e., relevant n-grams are suggested on the fly by completing the query that the user is typing. In our experiments, we use this approach as a baseline to compare with our approach (see Chapter 3). In the medical domain, Luo et al. [70] propose a specialized medical search engine that can suggest related medical phrases. For this, an external ontology (MeSH[2]) is used to extract related phrases as suggestions. However, such an ontology may not be applicable to other domains such as patent retrieval and academic literature search that we also address in this work.

Another line of related work on query suggestion is diversifying query suggestions (e.g., [73, 94]). While search result diversification (e.g., [3, 18]) aims at producing retrieval results that contain a mixture of (topically) different documents, query-side

[2]Medical Subject Headings (http://www.ncbi.nlm.nih.gov/mesh)

diversification focuses on generating a list of diverse queries, which maximizes the diversity between query-suggestion pairs. Ma et al. [73] proposed a framework for diversifying query suggestions. They first generated the Markov random walk model on the query-URL bipartite graph by adding the top ranked query into a candidate set. Then, other queries were ranked by running the expected hitting time analysis, which could demote the ranks of (unranked) queries to the ranked queries. Thus, the result of ranked suggestions (i.e., queries) could be diversified. Song et al. [94] also discussed the same problem, and selected query candidates from query logs by ranking them in the order that maximizes the similarity and diversity between the queries. They measures diversity based on the difference between the original search results and the results of suggested queries. To quantify the difference, several features were devised, e.g., the similarity of the ODP[3] category of two search results, rank correlation coefficient for the URLs in two search results, etc. However, these studies are also limited in their application to domain-specific search environments as they require query logs and clickthrough statistics.

2.5 Search Result Diversification

Search result diversification is the task of generating a ranked list of documents that covers a range of query topics (or aspects). Previous work on this task can be categorized as: (1) implicit or (2) explicit [92]. We provide a brief summary for each category.

Implicit diversification: The implicit approach does not assume any explicit representation of query topics. MMR (Maximal Marginal Relevance) [18] and its probabilistic variants [109] can be included in this approach. For diversification, these methods assume that each document in the initial retrieval results represents

[3]Open Directory Project (http://www.dmoz.org)

its own topic and iteratively selects the documents that are dissimilar to previously chosen documents. To measure the dissimilarity, MMR used content-based similarity functions, but probabilistic distance in the language modeling framework has also been used in [109]. In addition, the correlation between documents is adopted as a similarity measure [87, 102]. Rafiei et al. [87] interpreted the problem of diversifying search results as expectation maximization, and proposed the portfolio model maximizing diversity in search results. Also, Wang and Zhu [102] used an economic theory dealing with financial investments for optimizing relevance (mean) against its risk level (variance) in the search result. Based on this, they devised a document ranking method which generalizes the probability ranking principle for selecting top n documents, and this was adapted to sub-topic retrieval. In these approaches, the diversification problem is viewed as minimizing the correlation, and the proposed algorithms are less effective [3, 29, 92] because sometimes the topics in the final results are not related to the query aspects.

Explicit diversification: In contrast to the implicit method, this approach requires some representation of query topics (e.g., [3, 19, 29, 92]). There are two different approaches to implementing explicit diversification: redundancy and proportionality. The redundancy approach is used in many existing methods (e.g., IA-Select [3], xQuAD [92]). These aim to provide less redundant information in the diversified results, i.e., documents are promoted if they include novel content that has not appeared in early ranks. In particular, xQuAD (eXplicit Query Aspect Diversification) used query reformulations to indicate underlying query aspects, and attempted to maximize the coverage and minimize the redundancy with respect to the underlying aspects. On the other hand, the proportionality-based algorithms (e.g., PM-2 [29]) selected the documents with respect to the "popularity" of their topics in the initial ranking, i.e., ranking the documents is proportional to the popularity of each query topic. This approach exploits the method to allocate seats in party-list proportional

representation, for assigning the portions of query topics such that the number of each topic's documents in the final result is proportional to the weight of the topic. Both of these approaches have been successful with test collections that contain manually created query topics (e.g., from TREC descriptions [29, 92]).

To provide a more realistic context, methods for automatically generating query topics have been studied (e.g., [30, 86]). As an example, query topics have been generated by clustering similar queries from query logs [86] or anchor texts from the web [30]. More recently, term-level diversification [28] has showed the effectiveness of automatic topic generation based on identifying important vocabulary terms. In this approach, query topics are described by some set of terms, and instead of generating the topics directly, only the important words and phrases associated with the topics are automatically identified, e.g., the words "pain", "joint", "woodwork", and "type" are identified for the latent topics of "joint pain" and "woodwork joint type". After identifying the important vocabulary, the diversification framework (e.g., xQuAD or PM-2) can be applied using the identified topic terms (the frameworks consider each term as a topic). The effectiveness of these automatically–found topic terms has been shown to be similar to the manually generated topics, and significantly better than other approaches to automatic topic identification. Our diversification framework for domain-specific searches uses this approach, and we focus on identifying topic phrases (e.g., "file system" and "system service" for patent retrieval) and diversifying with respect to these phrases. In [28], a set of terms to represent initial retrieval results is generated for an initial ranked list of documents. This is similar to the goal of multi–document summarization (e.g., [65, 66]). Thus, DSPApprox, a hierarchical summarization algorithm proposed in [65], has been used for identifying topic terms in [28]. This algorithm iteratively selects the terms which maximize predictiveness and topicality. However, in addition to predictiveness and topicality, we explore additional features to identify topic phrases, e.g., relevance, cohesiveness, and query performance

predictors (described in Chapter 7). Moreover, we examine the effectiveness of these features in the context of diversification.

2.6 Summary

In this work, we propose automatic query generation and search result diversification frameworks to help users in domain-specific searches. We attempt to reduce the burden of formulating queries from query documents, and help users to easily retrieve relevant documents. Figure 2.1 depicts the process of query generation and search result diversification in domain-specific search environments.

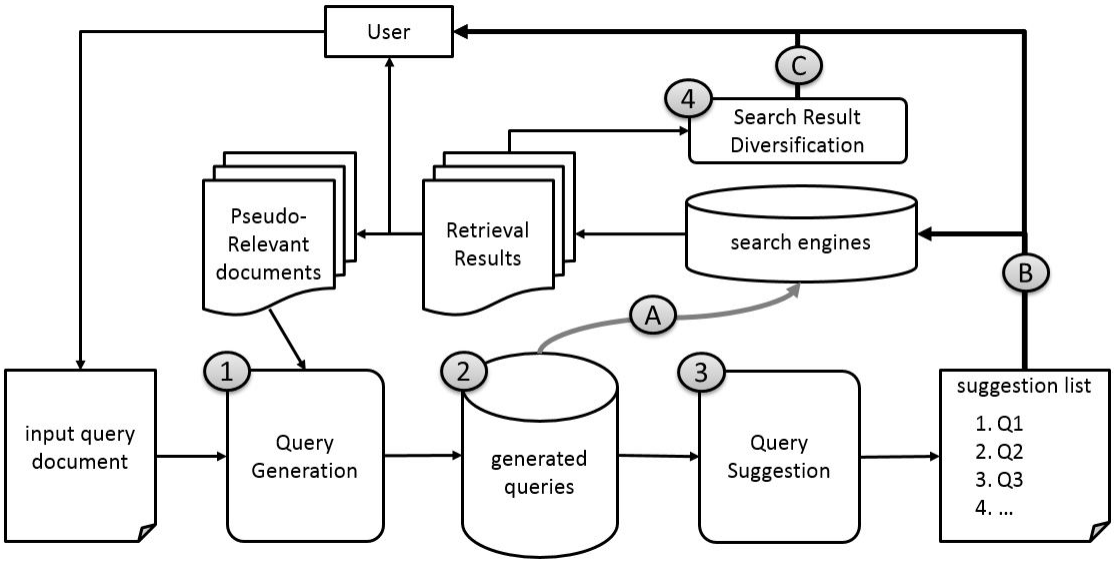

Figure 2.1: Query Generation and Diversification Process

For many domain-specific search tasks, users input a query document, and search queries are generated based on the document. Most of the prior work (e.g., [38, 75, 106]) assumed that generated queries are only used for retrieval by search engines (A in Figure 2.1), and the retrieval effectiveness of a query was mostly focused in query generation. However, we consider using generated queries for suggestion as well as retrieval (B in Figure 2.1). In addition, we consider the diversification of retrieval

results (C in Figure 2.1), which has not been focused in previous work. By using these approaches, we attempt to resolve the three issues in domain-specific searches, i.e., query generation, query suggestion and formulation, and diversification (discussed in Chapter 1.2).

We restate the contributions of our work as follows.

1. For query generation (1 in Figure 2.1), we devise methods to identify effective query terms that can retrieve more relevant documents.

2. In query generation, we also devise algorithms to identify important query topics and generate diverse queries based on the identified topics (i.e., query-side diversification).

3. We propose methods that can formulate queries in particular representations that users prefer, e.g., Boolean queries or phrasal-concept queries (2 in Figure 2.1).

4. We develop methods to generate diverse and effective suggestions (3 in Figure 2.1).

5. For search result diversification (4 in Figure 2.1), we devise methods for phrase-level topic identification, and apply state-of-the-art diversification algorithms.

First, we intend to generate queries in user-preferred representations, e.g., Boolean queries, whereas previous work only assumed to use weighted keyword queries (e.g., [38, 75, 106]). Second, in previous work, diversification in domain-specific searches has not been a major focus (rather than that, only retrieving more relevant documents has been focused). However, we approach the diversification issue by two different methods, i.e., query-side and result-level diversification. Third, as diverse queries are generated, we devise methods to diversify query suggestions. Fourth, we apply

state-of-the-art diversification algorithms to domain-specific search environments by generating topic phrases from query documents.

CHAPTER 3

EVALUATION FRAMEWORKS

3.1 Overview

In this chapter, we describe how to set up experiments for evaluating our proposed methods. We basically design the experiments to simulate domain-specific searches. Specifically, we choose target domains of interest, define search tasks related to the selected domains, and generate test-collections (i.e., queries and relevant documents) for the target domains. Using these, the proposed methods and baselines generate and suggest queries in simulated settings. In addition, we conduct experiments for diversification as we propose search result diversification methods. To quantify the effectiveness of our methods, we measure how many effective queries are generated, how many of them are "actually" suggested to users (if the users only examine a small number of suggestions), and how many relevant documents are retrieved by top suggestions. We also examine how much diverse search results are obtained. For these, we adopt conventional IR evaluation metrics (e.g., precision, recall, and NDCG (Normalized Discounted Cumulative Gain) [49]) as well as "diversity" metrics (e.g., α-NDCG [25] and Intent-Aware precision [3]). In addition, domain-specific metrics (e.g., PRES [74] for patent retrieval) are adopted for evaluations. We provide details of these in the rest of this chapter.

3.2 Domain-specific Search Tasks

We design experiments for three different search tasks considering three domains of interest: the patent, academic, and medical domains. The search task for the patent

domain is patentability search, also known as prior-art search or invalidity search [36, 98]. This task is very common and important in patent retrieval [53], which aims to find prior patents (previously published) that may conflict with a new (query) patent. In this task, given a query patent, we automatically generate and suggest a list of queries that can effectively retrieve relevant patents. For the academic domain, the search task is finding academic papers relevant to a current research project. In this task, we assume that a scientist (user) inputs a summary of his research (e.g., title and abstract texts of his research paper) as an initial query document, and a list of queries is generated and suggested for retrieving existing papers relevant to the research project. The search task for the medical domain is reference retrieval for physicians. We assume that physicians provide a statement of information about their patients as well as their information need, and we generate a list of queries that can retrieve relevant medical references for the information request. For each domain, we can also diversify search results by the proposed diversification methods (described in Chapter 7).

3.3 Test Collection

For the three domain-specific search tasks (Chapter 3.2), we develop test collections as follows.

Patent Domain: To conduct experiments for the patent domain, we use two different corpora: USPTO (United States Patent and Trademark Office) and EPO (European Patent Office) patents. USPTO patents are provided by NTCIR-6 [36, 55]. This collection contains 981,948 patents published from 1993 to 2000. To develop query documents (new patents), we randomly selected 100 patents published in 2000, ensuring that their citations list more than 20 patents and at least 90% of them are included in the test collection. As done in the TREC chemical track [71] and NTCIR-6 [36], we consider patents cited in each query patent as "relevant", and 22.64 relevant

documents are found on average. We call this collection USPAT. The other collection we use for the patent domain is the CLEF-IP 2010 [34] corpus which contains 2.6 million EPO patents. We randomly select 300 query patents from the query patent pool they provide. Although the query documents are described in the three official EPO languages (English, German, French), we only work with English documents. Relevance assessments are provided, which also use the citations listed in each query patent [34]. The average number of relevant documents is 28.87, and we call this collection EPO.

Academic Domain: As an academic corpus, we use the ACL anthology corpus that contains academic literature [15]. This collection includes 10,921 academic papers published from 1975 to 2007, and these papers are about the topics in Natural Language Processing. The full text of each article is available, and metadata (e.g., author names, venues, titles, and citations) is also provided. We removed stop-words including frequently used acronyms (e.g., "fig.") and section names (e.g., "introduction" and "related work") from the documents. To develop query documents (new research projects), we randomly selected 183 query papers published in 2006 from the collection, ensuring that their citations list contain at least 10 articles. As done in previous research [13, 88, 96], we consider the articles cited in each query paper as "relevant" and 12.19 citations are listed on average. Note that we ignore the citations not included in the collection. In addition, we discard the references to articles outside of the collection that is searched, and the query papers are removed from the collection and relevance judgments for other papers.

Medical Domain: For the medical domain, we use the OHSUMED collection [46] which consists of 348,566 medical references (documents) and 106 queries. Each query contains the statement of patient information and information need from physicians, and we consider this as a query document. This collection provides relevance judgments manually annotated using three relevance levels: *definitely relevant*, *possi-*

bly relevant, and *not relevant*. In the experiments, we consider *definitely* and *possibly relevant* as "relevant".

3.4 Assumptions for Experiments

In the experiments, we implement domain-specific search simulations in that multiple queries are generated and suggested. To evaluate query generation and suggestion methods, we made the following assumptions. First, search users directly use suggested queries without reformulation. For each query document, we suggest a list of queries generated by our methods. By assuming that the suggested queries are used without any reformulation, we could show the lower bound of performance that the proposed methods can achieve. In real environments, users may use our suggestions or formulate new queries based on the suggestions. Second, in a multi-query session (i.e., multiple queries are suggested in a search session), users try the queries in the suggestion order. Since modeling user behavior (e.g., [56]) is beyond the scope of our work, we simply assume that users sequentially examine the queries starting from the first one. To evaluate diversification techniques, we assume that an initial ranking result is provided. In other words, for each query document, we generate a baseline query to produce an initial retrieval result. Then we apply the diversification framework we propose.

3.5 Baselines

In this work, we propose the methods to generate multiple queries, suggest effective ones, and diversify search results. Accordingly, we employ appropriate baselines for each proposed method as follows.

3.5.1 Query Generation

PriorArtQuery: As a baseline for patent search, we use a query generation method described in [107]. Given a query patent, this method generates a prior-art query which includes the top n unigrams ranked by their tf.idf weight from the "brief summary" section of the query patent. To produce more effective queries, each query term is weighted by its term frequency in the query patent. We call this weighted query PriorArtQuery.

ReductionQuery: Reduction Query is another baseline for patent search, which is proposed in [38]. Given a query patent, this method first collects pseudo-relevant documents, the top k results initially retrieved by the query patent (Note that PriorArtQuery can be used for generating initial retrieval results). Then, the sentences (in the query patent) more similar to the pseudo-relevant documents are extracted to form a query. We call this query ReductionQuery.

EX–RM: For patent search, we can consider another query generation method proposed in [75]. In that, a unigram query is first generated by ranking the single terms in a query patent; for this, a unigram language model is derived based on the query document, and Kullback–Leibler divergence between the query model and collection model is used for the ranking. Then, the original query is expanded by a relevance model estimated from the same IPC (International Patent Classification code[1]) documents (i.e., the documents containing at least one common IPC code of the query patent). IPC codes are manually annotated to any patent documents, and can classify a patent document into predefined classes. So, the same IPC documents would contain terms more related to the query patent, and the expanded query would be effective for retrieving relevant documents. This expanded query is called EX–RM.

[1] http://www.wipo.int/classifications/ipc/en/

Sequential Dependence Model (SDM): Metzler and Croft [79] proposed a method to capture term dependencies in a query. In this, a joint distribution over a query and target document is modeled using an Markov Random Field (MRF), undirected graphical model, and an MRF is generally defined by a graph and set of non-negative potential functions over the cliques in the graph. Formally, given an undirected graph, G, the joint distribution over a (initial) query, Q, and document, D, is defined as:

$$P(Q, D) = \frac{1}{Z} \cdot \prod_{c \in C(G)} \psi(c; \Lambda) \tag{3.1}$$

where $C(G)$ is the set of cliques in G, Z is a normalizing factor, $\psi(\ \cdot\ )$ is a potential function, and Λ is a corresponding parameter vector.

In constructing G, we assume a sequential dependence between adjacent query terms, and accordingly 3 different types of cliques are formed as follows:

- T_D: set of cliques containing D and exactly one query term.
- O_D: set of cliques containing D and two query terms sequentially appeared in Q.
- U_D: set of cliques containing D and two query terms observed by any order in Q.

O_D is a sub-set of U_D, and we can control the impact of each clique type by tying the corresponding parameters (i.e., λ_{T_D}, λ_{O_D}, and λ_{U_D}). Based on these, the actual ranking function is given as:

$$P(Q, D) = \frac{1}{Z} \cdot \exp\Big\{ \sum_{c \in T_D} \lambda_{T_D} f_{T_D}(c) + \sum_{c \in O_D} \lambda_{O_D} f_{O_D}(c) + \sum_{c \in U_D \cup O_D} \lambda_{U_D} f_{U_D}(c) \Big\} \tag{3.2}$$

where $f_{\cdot}(c)$ is a feature function for a clique, c.

The value of each feature function can be calculated by the log-likelihood of a smoothed language model probability for c, and we empirically set the controlling

parameters as $\lambda_{T_D} = 0.80$, $\lambda_{O_D} = 0.15$, and $\lambda_{U_D} = 0.05$. In various IR tasks SDM has been proven as the most effective technique (e.g., [8, 12, 20]), and we also expect that this model is particularly effective for the academic literature search tasks because sequential dependencies are able to capture effective phrases appeared in relevant papers.

Latent Concept Expansion (LCE): Latent Concept Expansion [80] is a robust pseudo-relevance feedback technique based on an MRF. Comparing to relevance models [64], this method is more generalized and can model term dependencies in a pseudo-relevance feedback process. To obtain feedback terms, we first obtain the top k pseudo-relevant documents (ranked using the sequential dependence model), and then the terms in the set of pseudo-relevant documents, R_D, are ranked by:

$$\text{LCE}(t) = \sum_{D \in R_D} \exp\left\{\gamma_1 SDM(Q, D) + \gamma_2 \log\left((1-\alpha)P(t|D) + \alpha P(t|C)\right) - \gamma_3 \log P(t|C)\right\} \tag{3.3}$$

where t is a feedback term, D is a document in R_d, Q is the initial query, $SDM(Q, D)$ is a ranking score obtained by the sequential dependence model, α is a smoothing parameter, $P(t|D) = tf(t, D)/|D|$, $P(t|C) = tf(t, C)/|C|$, $tf(t, D)$ is the term frequency in D, $tf(t, C)$ is the term frequency in a collection, C, and γ_i is a free parameter.

In this method, a feedback term is obtained by considering three features: (i) document relevance ($SDM(Q, D)$), (ii) term likelihood to the pseudo-relevant document model ($\log((1-\alpha)P(t|D) + \alpha P(t|C))$), and (iii) dampening factor ($\log P(t|C)$) to avoid highly common terms in C. We select the number of top k documents for R_D and m (unigram) terms for feedback, and free parameters are set by n–fold cross validation. In addition, we did experiments using bigrams for the feedback, but could not obtain any significant improvements relative to the results using unigrams.

Relevance Model (RM): Relevance Model is another pseudo-relevance feedback technique with proven effectiveness and robustness [72]. The basic idea is that

to determine feedback terms and their weights, models of feedback documents are combined using query likelihood scores of feedback documents as weights.

Given a (initial) query, Q, and the set of pseudo-relevance documents, R_D, the feedback formula can be given as:

$$P(t|Q) \propto \sum_{D \in R_D} P(t|D)P(D) \prod_{q \in Q} P(q|D) \tag{3.4}$$

where q is a query term in Q.

To improve retrieval performance, we interpolate this relevance model with the original query model, M_Q, [1], and the final formula can be given as:

$$P(t|M'_Q) = (1 - \alpha)P(t|M_Q) + \alpha P(t|Q) \tag{3.5}$$

where M'_Q is an (interpolated) expansion query model and the interpolation parameter was set as 0.5. We can extract the top m terms ranked by Eq. (3.5) for feedback.

Machine Learning-based Expansion (MLE): This method uses a statistical learner for pseudo-relevance feedback, inspired by [47] that exploits supervised learning algorithms. Given an initial query, to obtain a set of feedback terms, a linear regressor is trained with a set of features where each feature corresponds to a (unigram) term appearing in training documents (pseudo-relevant documents obtained by the initial query). Then, the trained regressor estimates the (pseudo) relevance score of a new document, and the terms corresponding to highly weighted features are predicted to be effective for predicting pseudo-relevance. Note that this is a totally unsupervised procedure in that we do not use human-labeled samples.

We generate a set of training examples by using the top 100 pseudo-relevant documents and randomly sampled non-relevant documents which are not in the top 100 as positive and negative samples. We scale (pseudo) relevance to an interval

$[0, 1]$ and use them as target values in training. Specifically, we assume 11 different relevance degrees, i.e., $\{0.0, 0.1, 0.2, \ldots, 1.0\}$, and generate 11 distinct sets, each of which contains an equal number of training examples where each set is mapped to the degree of the relevance; the top 100 pseudo-relevant documents are divided into the degrees from 0.1 to 1.0 (e.g., the top-1 to 10 documents are assigned to 1.0) and the beyond-100 documents are used for 0.0 (non-relevant). A feature set contains all words (except stop-words) from the pseudo-relevant documents, and a feature value is calculated by the tf.idf weight of a term in each document. After training, a weight vector, β is obtained, and among all components of β, we can select the top m features (terms) by ranking them in descending order of their absolute weight values in β. To formulate an expanded query, the initial query is combined with the top m feedback terms, and the weight value from β is used for feedback term weighting. The bias to feedback terms against the initial query is set as 0.5. We also test this method with the features of noun phrases (longer than unigram) syntactically recognized from the training examples using a phrase recognizer, (MLE-P) and n different noun phrases can be selected for feedback.

3.5.2 Query Suggestion

As discussed in 2.4, many methods for query suggestion typically rely on using query logs and clickthrough statistics because they recommend queries for web search users. However, such resources may not be readily available in domain-specific search environments. Bhatia et al. [14] proposed an n-gram query suggestion method that does not use query logs. Given an initial query, they suggest n-grams more correlated with the query. Since the original method aims at providing relevant n-grams when a user partially types an initial query (e.g., types the first l characters of the query), we modify the method to fit in our search environments; we assume that a user finished typing the initial query and query completion is unnecessary. Based on this, the

equation for selecting n-grams is given as:

$$P(p_i|Q_0) \approx P(Q_0|p_i) \tag{3.6}$$

where p_i is an n-gram phrase and Q_0 is an initial query.

We use phrase-query correlations to estimate $P(Q_0|p_i)$ as follows:

$$\log P(Q_0|p_i) \approx \log \prod_{np \in Q_0} P(np|p_i) \approx \sum_{np \in Q_0} \log \frac{df(np, p_i)}{df(p_i)} \tag{3.7}$$

where np is a noun phrase and $df(\cdot)$ denotes the document frequency in a corpus.

For an initial query, Q_0, we use the title of a query document, but in query ranking, as we see in Eq. (3.7), we count only noun phrases (longer than unigram) in Q_0 because counting the correlation of every term in Q_0 is less efficient and noisy (e.g., the title texts contain less important terms such as "in" and "which"). To develop suggestions, we rank all n-grams of order 2, 3, 4, and 5 (i.e., bigrams to five–grams) from pseudo-relevant documents. We call this method NGram throughout this paper.

3.5.3 Search Result Diversification:

To evaluate our search result diversification approach, we adopt the term-level diversification method proposed in [28]. This method exploits an automatic topic term identification for improving diversification. In that, a set of terms to represent query topics is first generated by DSPApprox (a term-level summarization technique) [66] and then diversification algorithms (i.e., xQuAD [92] and PM-2 [29]) are applied with the identified topic terms. In Chapter 7, we provide more details of this method.

3.6 Retrieval Models

In order to run generated queries, we use the following retrieval models.

Indri: Indri [97] is a language modeling-based search engine. The Indri retrieval model combines the language modeling [84] and inference network [100] retrieval frameworks. These approaches have been applied to a broad range of IR tasks, and proven to be effective (e.g., [81]). We basically use the query likelihood model to run baseline queries (e.g., PriorArtQuery and EX–RM). However, more complex approaches (e.g., SDM, RM and LCE) are implemented. In addition, we develop retrieval models for specific query formulations (e.g., Boolean queries).

Statistical Boolean Retrieval Model: To run Boolean queries, we use a statistical Boolean retrieval model. For each query document, we first find all documents satisfying the given Boolean function (i.e., Boolean query) and rank the documents by the generative probability of the query:

$$P(BQ|D) \approx \prod_{q \in BQ} P(q|D) \approx \prod_{q \in BQ} \frac{tf_{q,D} + \mu \cdot P(q|C)}{|D| + \mu} \tag{3.8}$$

where D is a target document satisfying a Boolean query, BQ, q is the query term not associated with negation in BQ, $tf_{q,D}$ is the term frequency of q in D, $P(q|C)$ is the probability of q in the collection, C, and μ is the Dirichlet smoothing parameter [110].

We do not employ any query processing including query term weighting in this Boolean retrieval model. Since many current patent search systems (e.g., Patent Scope[2]) are also based on these simple term statistics, query evaluation using this statistical Boolean retrieval model would be more practical and similar to real search environments than using other enhanced retrieval techniques (e.g., learning-to-rank) that are hard to integrate into current patent search systems.

Learning-to-rank Retrieval Model: For the academic domain, we implement a learning-to-rank retrieval model using SVM$^{\text{rank}}$ (Support Vector Machine for Rank-

[2]http://patentscope.wipo.int/

ing) [51, 52]. This model can efficiently learn the weights of retrieval features from training data. Since academic papers can include multiple meta information (e.g., authors, publishers, venues, and citations), the features extracted from this information could improve retrieval models for the academic literature search task (e.g., research interests of authors [10] and citation behaviors [13]). We select the 12 most effective features from those proposed in [13], which describe age of the query paper, citation pattern, and author citation behavior. In addition, we leverage typical query-based features (e.g., the tf.idf score) described in [17]. Table 3.1 provides the description of each feature. In that, a t, q, d, and dq indicate a term, query, target document, and the query paper where q is generated, respectively; $freq(t, d)$ represents frequency of term t in document d; $idf(t)$ denotes inverse document frequency of term t; C denotes the entire collection; $|C|$ denotes the size of vocabulary in C.

PATATRAS: For the patent domain, some specific retrieval models have been proposed and proven to be effective (e.g., [77, 106]). The PATATRAS model proposed in [68, 69] can improve retrieval effectiveness by combining multiple retrieval models based on multilingual documents. This approach performed the best in CLEF-IP 2010 [33]. In this method, each query patent is processed by lemmatization, key-term extraction, and concept-tagging. Then, the PATATRAS approach is applied, which can combine multiple retrieval models (i.e., Indri and BM25 [89]) by merging the different retrieval results based on regression. Since this method relies on a multilingual concept database and the indexes on multi-language documents, we could only implement this for the EPO. Note that the patents in EPO are written in English, French, and German, while USPTO contains the US patents only written in English.

Table 3.1: Retrieval Features

Category	**Feature**	**Description**
Query	$\text{tf}(q,d)$	$\sum_{t\in q\cap d}\log(freq(t,d)+1)$, frequency of query term
	$\text{idf}(q,d)$	$\sum_{t\in q\cap d}\log(idf(t))$, inverse document frequency
	$\text{tfidf}(q,d)$	$\sum_{t\in q\cap d}\log\left(\frac{freq(t,d)}{\lvert d\rvert}\cdot idf(t)+1\right)$, tfidf score
	$\text{icf}(q,d)$	$\sum_{t\in q\cap d}\log\left(\frac{\lvert C\rvert}{freq(t,C)}+1\right)$, inverse collection term frequency
	$\text{lm}(q,d)$	$\sum_{t\in q\cap d}\log\left(\frac{freq(t,d)}{\lvert d\rvert}+1\right)$, unigram language model score
Citation	$\text{tfidf}_{\text{citation}}(q,d)$	tfidf score between q and all ciations of d
Age	$\text{recency}(d)$	# of years since d was published
Citation Pattern	$\text{cnt}_{\text{citation}}(d)$	# of times d was cited
	$\text{PageRank}(d)$	PageRank score [16] of d in the citation network including all articles
	citation-venue (d)	citation count of articles published by the venue of d
Author Citation Behavior	citation-author (d)	citation count of the most cited author among authors of d
	authors-self (d_q,d)	over-lapping between authors of d_q and authors of d
	authors-citing (d_q,d)	over-lapping between authors of d_q and authors of articles citing d
	authors-anyciting (d_q,d)	over-lapping between authors of d_q and authors of articles citing articles written by any authors of d
	authors-venue (d_q,d)	over-lapping between authors of d_q and authors of articles citing articles published by the venue of d
	authors-coauthor (d_q,d)	over-lapping between any authors of d_q and coauthors of d (i.e., coauthors indicate the authors who have coauthored with any authors of d)

3.7 Evaluation Metrics

3.7.1 Conventional Evaluation Metrics

In order to measure retrieval performance, we use traditional IR evaluation metrics (e.g., Precision and Recall) as well as task-specific metrics (e.g., Patent Retrieval Evaluation Score (PRES) [74]). We also measure Mean Average Precision (MAP) and Normalized Discounted Cumulative Gain (NDCG) [49] at the top k retrieved documents. The calculation of each metric is given as follows.

First, Precision at top k is defined as the fraction of k retrieved results (documents) that are *relevant*, which can be calculated as:

$$\text{Precision}(R, D_k) = \frac{|R \cap D_k|}{|D_k|} \tag{3.9}$$

where D_k is the top k retrieved results and R is the set of relevant documents.

Second, Recall at top k is measured by the fraction of relevant documents that are *retrieved* within the top k results, which can be given as:

$$\text{Recall}(R, D_k) = \frac{|R \cap D_k|}{|R|} \tag{3.10}$$

Since many domain-specific search tasks are recall-oriented (e.g., Prior-Art Search [36, 77]), this metric is important and frequently used in our evaluations.

To evaluate recall-oriented tasks more effectively, we additionally adopt PRES [74]. This metric reflects the normalized recall incorporated with the quality of ranks of relevant documents observed within the maximum number of documents that the user examines. In PRES, we assume that there is a maximum number of retrieved documents to be examined by the user (i.e., N_{max}), and the worst case for retrieval is that all the relevant documents are placed after the such maximum number of documents (obviously the best case is that all the relevant documents are retrieved

at top ranks). Figure 3.1 illustrates how the PRES curve can be drawn with this assumption.

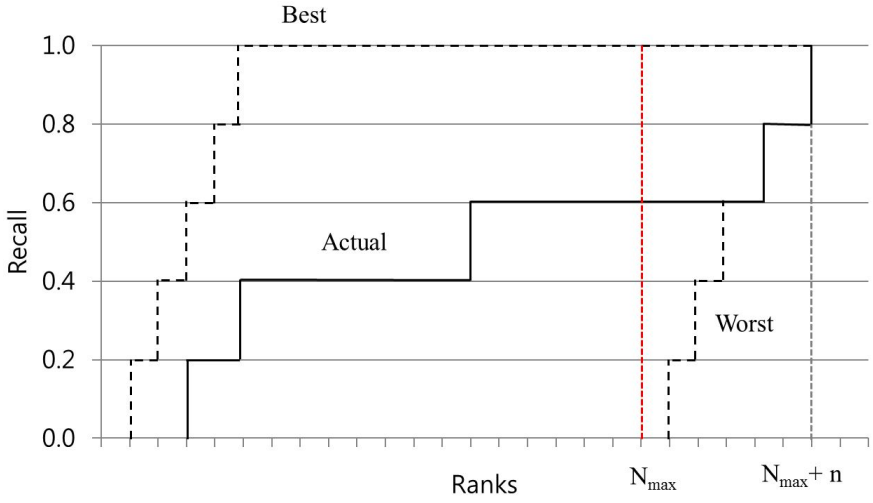

Figure 3.1: PRES Curve

Based on this, the equation for calculating PRES can be given as:

$$PRES = 1 - \frac{\sum_{i=1}^{n\mathrm{R}} \mathrm{rank}(r_i) + n\mathrm{R}(N_{max} + n) - (n\mathrm{R}(n\mathrm{R} - 1)/2) - \sum_{i=1}^{n} i}{n \times N_{max}} \quad (3.11)$$

where n is the number of relevant documents, N_{max} is the maximum number of retrieved documents examined by the user, R is the recall at N_{max}, and rank(r_i) is the rank of i-th relevant document.

Additionally, we consider the F-score for evaluations as it balances precision and recall performance.

$$\mathrm{F}_\beta = (1 + \beta^2) \cdot \frac{Precision \cdot Recall}{(\beta^2 \cdot Precision) + Recall} \quad (3.12)$$

where F_1-score is the harmonic mean of Precision and Recall.

Third, we can measure MAP by calculating Average Precision on retrieval results. Average Precision, AveP, is the average of precision at each point where a relevant document is found is computed as:

$$\text{AveP}(R, D_k) = \frac{\sum_{d_i \in D_k \cap R} P(R, D_i)}{|R|} \tag{3.13}$$

where d_i is a i-th ranked result in D_k and D_i is the results from 1 to i-th ranked document ($D_i \subseteq D_k$).

Then, for a given set of queries, Q, MAP can be calculated by:

$$\text{MAP}(Q) = \frac{\sum_{q \in Q} \text{AveP}(R_q, D_{k,q})}{|Q|} \tag{3.14}$$

where q is a query in Q, R_q is the relevant documents of q, and $D_{k,q}$ is the top k retrieved results of q.

Fourth, NDCG is measured using the Discounted Cumulative Gain (DCG) which discounts the documents placed at the lower ranks in the retrieval list. The DCG of a particular rank, DCG@k, is defined as:

$$\text{DCG@}k = rel_1 + \sum_{i=2}^{k} \frac{rel_i}{\log_2(1+i)} \tag{3.15}$$

where rel_i is the relevance of the result at position i and $rel_i \in \{0, 1\}$.

Based on this, the NDCG at position k, NDCG@k, can be computed as:

$$\text{NDCG@}k = \frac{\text{DCG@}k}{\text{IDCG@}k} \tag{3.16}$$

where IDCG is an ideal DCG score, i.e., when every relevant document is placed at the top of the retrieval list.

In addition to these, we employ session-based metrics that can measure the overall effectiveness of multiple queries because we suggest multiple queries for a search session. Javelin et al. [50] proposed the Nomalized Session Discounted Cumulative Gain (NSDCG) which discounts documents that appear lower in a ranked list of an individual query as well as documents retrieved by the later suggested query. Given a session, NSDCG@k is calculated as follows.

First, a rank list is constructed by concatenating the top k documents from each ranked list of the session. For each rank i in the concatenated list, the discounted gain (DG) is computed as:

$$\text{DG@}i = \frac{2^{rel_i} - 1}{\log_2(1+i)} \tag{3.17}$$

where $rel_i \in \{0, 1\}$

We then apply an additional discount to documents retrieved by later suggestions. For example, the documents ranked between 1 and k are not discounted at all, but the documents ranked between $k+1$ and $2k$ are discounted by $1/\log_{bq}(2+(bq-1))$ where bq is the log base and determined by search behavior. A larger base, e.g., 10, indicates that a searcher is patient and willing to examine more suggestions, while a smaller base, e.g., 2, represents an impatient searcher. In our work, we use $bq = 10$ because academic searchers would use many queries to investigate more relevant articles. Then, Session Discounted Cumulative Gain (SDCG) at top k is calculated by:

$$\text{SDCG@}k = \sum_{i=1}^{nk} \frac{1}{\log_{10}(j+9)} \text{DG@}i \tag{3.18}$$

where $j = \lfloor (i-1)/k \rfloor$ and n is the number of suggestions (queries) in a session.

Accordingly, the final formula for NSDCG@k is given as:

$$\text{NSDCG@}k = \frac{\text{SDCG@}k}{\text{Ideal SDCG@}k} \tag{3.19}$$

where Ideal SDCG@k is an "ideal" score of SDCG obtained by an optimal ranked list in decreasing order of relevance.

3.7.2 Diversity Metrics

In this work, we attempt to diversify domain-specific search results (see Chapter 1). To measure "diversity" on retrieval results, α-NDCG [25], ERR-IA (a variant of ERR (Expected Reciprocal Rank) [21]), NRBP [23], and subtopic recall (S-Recall) are used. These metrics penalize redundancy in retrieval results, i.e., how much of the information in each retrieved relevant document the user has already obtained in earlier ranks. Note that these have been used as standard metrics for diversity tasks in TREC [24].

Moreover, we devise a new metric to measure diversity in "multi-query" sessions because these proposed metrics are not applicable to evaluating multiple queries (suggested for each query document). In addition, there was no emphasis on recall in session search results (but we concentrate on recall-oriented search tasks).

Session Novelty Recall (SNR) is a recall-based metric for multi-query sessions. In this metric, given multiple retrieval results, we ignore relevant documents already found by previous suggestions, i.e., newly retrieved relevant documents are only counted. Besides, following the idea in [50], we discount the documents retrieved by later suggestions. The computation of this metric is given as follows.

First, we construct a rank list, L, by concatenating the top k documents from each ranked list in a session. Next, in the list, we discard any retrieved documents which are retrieved by any previous queries, i.e., the rank list contains only distinct retrieval results. In addition, each retrieved result is labeled by the query which first retrieved it.

$$\mathrm{SNR} = \sum_{i=1}^{|L|} \frac{\mathrm{rel}(d_i^j)}{\log_b(j+b-1)} \Big/ |R| \tag{3.20}$$

where d_i^j is the document placed at the i-th rank in L and retrieved by the j-th suggestion in a session, R is the set of relevant documents, b is the number of queries that the user examines where $b > 1$, rel(d) returns 1 if d is relevant; otherwise, 0.

Ideally, if every relevant document is retrieved by the first query, SNR should be the maximum, i.e., 1. If none of the relevant documents are retrieved by any suggestions, the minimum is obtained, i.e., 0. Note that NSDCG and SNR can be applied to session retrieval results (obtained by multiple queries).

CHAPTER 4

BOOLEAN QUERY GENERATION

4.1 Overview

For a number of reasons, both historic and technical, Boolean queries are particularly common in professional search – domain-specific search tasks whose users are search professionals [58]. For example, in prior-art search, according to the user surveys [5, 53], the use of Boolean operators is one of the most important features to formulate effective queries from the perspective of patent professionals (i.e., search users). In addition, most patent users who participated in the survey from [5] did not regard query term weighting and query expansion as important, whereas more than 95% of the survey participants agreed that implementing Boolean operators is necessary. This is not because Boolean queries are the most effective. In fact, a number of studies over the years (e.g., [77, 82, 99]) have shown that "keyword" queries are often significantly more effective. However, Boolean queries are easy for domain-specific users to manipulate and can provide a record of what documents are retrieved. Thus, professional search users continue to have a strong preference for Boolean queries. Therefore, in this chapter, we propose our method for generating effective Boolean queries based on query documents. We start by defining terms and formulating relevant tasks for Boolean query generation and suggestion. After that, we describe our methods to generate effective Boolean queries and suggest them. In evaluations, we provide experimental results of the proposed methods by comparing with baseline query generation approaches (described in Chapter 3.5).

4.2 Problem Formulation

Definition 1. (Query Document): The query document is an initial document input by a user, which initiates a search task. For example, in patent retrieval, a new patent can be a query document, which initiates a prior-art search task. A query document is a subject for which multiple queries against search engines are formed, and the retrieval results are examined by users.

Definition 2. (Boolean Query): A Boolean query is a sequence of query terms all of which are connected by conjunction and each of which can be prefixed by negation, e.g., $battery \wedge ion$. In our work, as query term candidates, we consider bigrams as well as unigrams. Since too long queries are not much useful as suggestions, we empirically set the maximal number of terms in a Boolean query as 10.

Definition 3. (Pseudo-Relevant Documents): Pseudo-relevant documents are the top k documents initially retrieved by the query document. For example, we can generate a baseline query by the query generation method proposed in [107], and the pseudo-relevant documents are obtained by the baseline query. In our query generation, we exploit the pseudo-relevant documents to generate more effective queries.

Problem 1. (Boolean Query Generation): Boolean query generation is formulating Boolean queries from a set of query term candidates. Using terms appearing in a set of pseudo-relevant documents for a query document, we formulate Boolean queries that consist of effective terms and Boolean operators (AND and NOT), where query term candidates can be unigrams or bigrams extracted from the pseudo-relevant documents.

Problem 2. (Boolean Query Ranking): Boolean query ranking is determining a preference among generated Boolean queries for a query document with respect to an IR evaluation metric, e.g., recall. This is necessary for suggesting a reasonable number of effective Boolean queries (e.g., 5 to 10) to users because many queries can be generated in the Boolean query generation phase. We produce a ranked list of

generated Boolean queries where an effective Boolean query should be placed within the high ranks (e.g., top 10).

4.3 Decision Tree-based Boolean Query Generation

In this section, we propose a decision tree-based method for generating effective Boolean queries. Figure 4.1 describes the process of our Boolean query generation. We train decision trees using the baseline retrieval results (containing the top k pseudo-relevant documents and beyond k non-relevant documents) and formulate corresponding Boolean queries (BQs).

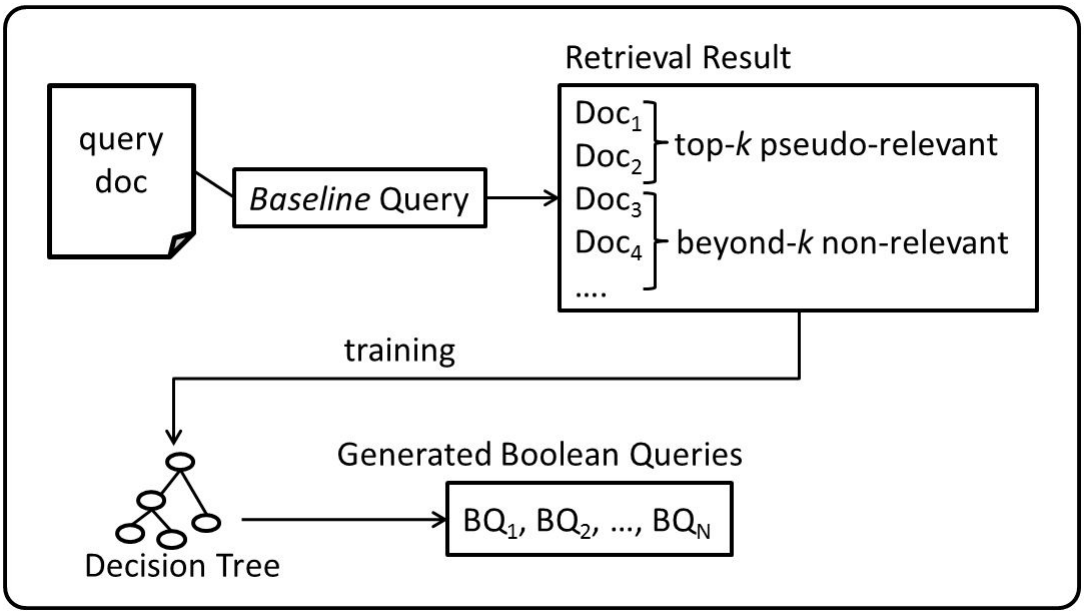

Figure 4.1: Boolean Query Generation Framework

Binary decision trees are equivalent representations of Boolean functions [91]. If we could train a decision tree where a node corresponds to a term appearing in training documents in order to determine whether a document is relevant to a topic, the learned decision tree could imply a Boolean query representing a set of relevant documents. In addition, the length and query terms of a Boolean query are naturally

determined by the depth and the nodes of the tree with reasonable accuracy. A problem, however, is that we do not have training data to learn a tree which can be generalized for every query because each query is associated with a different set of terms. So, instead of relevant documents, we use pseudo-relevant documents (*Def. 3*) as training data. In other words, we learn a decision tree by using the top k documents as positive examples. As negative examples, presumably non-relevant documents (ranked beyond k in the baseline retrieval results) are used. Accordingly, Boolean queries generated from the positive nodes of the learned decision tree are expected to be as effective as the baseline query because the decision tree is learned from the pseudo-relevant documents.

Once we learn a decision tree for a query document, we identify a single path from a root to a positive leaf node in the decision tree and convert the rule (path) into a Boolean query. Accordingly, a decision tree produces as many Boolean queries as the number of positive leaf nodes. Figure 4.2 depicts how to generate Boolean queries

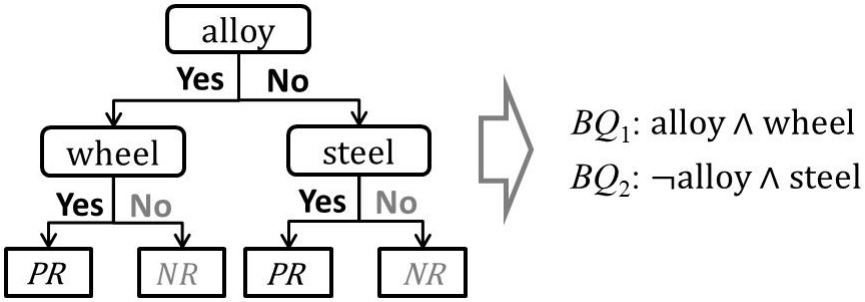

Figure 4.2: Boolean Query Generation Example

from an example decision tree whose attributes (query term candidates) are "alloy", "wheel", and "steel", and *PR* / *NR* values of each leaf node denotes a positive (pseudo-relevant) / negative (non-relevant) decision for input documents. For example, a document including "alloy" and "wheel" is classified as pseudo-relevant because a number of pseudo-relevant examples used for training include the two terms. That

is, the path from "alloy" to the first PR leaf can formulate the query, BQ_1, which is expected to retrieve documents containing "alloy" and "wheel". Since we concentrate on using conjunction and negation operators to formulate Boolean queries, we can generate two queries, BQ_1 and BQ_2, rather than a single unified query such as $(alloy \wedge wheel) \vee (alloy \wedge steel)$. This is because AND and NOT have more impact on the retrieval effectiveness and BQ_1 or BQ_2 performs empirically better than the unified query with respect to recall at the top 100 results that we use to evaluate a Boolean query.

Algorithm 1 Boolean Query Generation

Input:

$T = \{T_1, T_2, \ldots, T_N\}$: N different sets of attributes where T_i is a set of query term candidates

R_b: the baseline retrieval results for a query document

Output:

S: a set of Boolean queries

1: Initialize $S = \{\quad\}$
2: $R_P \leftarrow$ the top k documents from R_b ▷ positive examples
3: $R_N \leftarrow k$ documents randomly selected from the beyond k documents from R_b ▷ negative examples
4: **for** $i = 1$ to N **do**
5: Train a decision tree using $\{R_P, R_N\}$ as training examples and T_i as attributes
6: Find all paths from the root node to every positive leaf node in the trained decision tree and formulate corresponding Boolean queries
7: Append the formulated Boolean queries to S
8: **end for**
9: **return** S

Algorithm 1 shows the process of generating Boolean queries from several sets of query term candidates (i.e., attributes), a set of pseudo-relevant documents (the top k baseline retrieval results) and a set of non-relevant documents (the beyond k baseline retrieval results). To produce a sufficient number of Boolean queries for each query document, we train several decision trees with different attributes, while all the trees are trained by the same training set. In this approach, the training set includes the k positive (pseudo-relevant) documents and an equal number of negative instances

(non-relevant) for a query document. To obtain N sets of attributes, we extract the terms appearing in the query document or pseudo-relevant documents, and change the number of terms that belong to each set. We describe the details of generating attributes as follows.

We consider all terms appearing in the query document or its pseudo-relevant documents as query term candidates, and select m different terms as attributes by ranking them. To do this, we select unigrams which are likely to be generated from the query document or pseudo-relevant documents, assuming that terms are effective for retrieving relevant documents if the terms occur frequently in the query document or pseudo-relevant documents. For the ranking, we use the following language models:

$$\begin{aligned} P(w|qd) &\approx \frac{tf(w, qd)}{|d_q|} \\ P(w|D_{prel}) &\approx \frac{\sum_{d \in D_{prel}} tf(w, d)}{\sum_{d \in D_p rel} |d|} \end{aligned} \tag{4.1}$$

where w is a unigram term, qd is a query document, $tf(w, d_q)$ indicates w's frequency in d_q, and D_{prel} is the set of pseudo-relevant documents for d_q.

In the ranking, stop-words[1] are ignored, and we can rank all terms in the query document or pseudo-relevant documents by using Eq. (4.1). We select the top m terms as attributes for decision trees, and consider N different m's (i.e., $m, 2m, 3m, \ldots, N \times m$) to obtain N different sets of attributes.

In addition to this, we consider bigrams as query term candidates, and add an equal number of bigrams into each set of selected unigrams. To rank bigrams, we estimate smoothed bigram language models for the query document and the pseudo-relevant documents as follow:

[1]Stop-words contain articles, prepositions, acronyms (e.g., "fig."), (relative) pronouns, and general nouns (e.g., "method", "figure", "apparatus", etc.), frequently appeared in domain-specific documents (e.g., patents)

$$
\begin{aligned}
P(w_{i-1}w_i|qd) &\approx (1-\lambda)\frac{tf(w_{i-1}w_i, qd)}{tf(w_{i-1}, qd)} + \lambda P(w_i|qd) \\
P(w_{i-1}w_i|D_{prel}) &\approx (1-\lambda)\frac{\sum_{d\in D_{prel}} tf(w_{i-1}w_i, d)}{\sum_{d\in D_prel} tf(w_{i-1}, d)} + \lambda P(w_i|D_{prel})
\end{aligned}
\tag{4.2}
$$

where λ is a bias to unigrams, and the bigrams containing any stop-word are ignored.

4.4 Boolean Query Ranking

To select a reasonable number of effective queries from a pool of generated Boolean queries, we propose a Boolean query ranking model and introduce features for the model.

4.4.1 Learning-to-Rank Boolean Queries

In order to rank generated Boolean queries, we learn a ranking function which predicts the preference between Boolean queries. That is, given a query document and its generated Boolean queries, our ranking model produces a ranked list of the Boolean queries in descending order of retrieval effectiveness. To measure the retrieval effectiveness of each Boolean query, we need to use an evaluation metric appropriate to the given search task, e.g., recall at 100 (R@100) is used for prior-art search. Thus, we use ranks by the effectiveness of the Boolean queries generated for each query document as target values to be predicted. The formal definition of this ranking model is given as follows.

Suppose that $Y = \{r_1, r_2, \ldots, r_l\}$ is a set of ranks, where l denotes the number of ranks, and we can order the ranks $r_1 \succ r_2 \succ \ldots \succ r_l$ where $\succ$ indicates the preference between two ranks. For training, a set of query documents $QD = \{qd_1, qd_2, \ldots, qd_n\}$ is given and each query document qd_i is associated with $BQ_i = \{bq_{i1}, bq_{i2}, \ldots, bq_{in(qd_i)}\}$, a set of Boolean queries, where $n(qd_i)$ means the number of generated Boolean queries for qd_i and a list of labels $y_i = \{y_{i1}, y_{i2}, \ldots, y_{in(qd_i)}\}$, each of which $y_{ij} \in Y$ indicates the rank of each Boolean query, bq_{ij}. A feature vector $x_{ij} = \Psi(qd_i, bq_{ij}) \in X$ is

generated from each query document and Boolean query pair. We can represent a set of training examples as $S = \{(qd_i, BQ_i, y_i)\}_{i=1}^{m}$.

A ranking function $f : X \mapsto \Re$ maps a feature vector associated with a Boolean query to a score for the query. Specifically, this model generates a permutation of integers spanned in $[1, n(qd_i)]$ for qd_i, the corresponding Boolean query list, and the ranking function f. The permutation $\pi(qd_i, BQ_i)$ is defined as a bijection from $\{1, 2, \ldots, n(qd_i)\}$ to itself where bq_{ij} is identified by an integer of $[1, n(qd_i)]$ and $\pi(j)$ denotes the position of bq_{ij}. The model is learned to minimize a loss function which is defined by the disagreements between permutation $\pi(qd_i, BQ_i)$ and rank list y_i for every training query document.

For learning, we use SVM$^{\text{rank}}$. In contrast to Boolean Query Generation where only pseudo-relevance is considered, we use real relevance judgments to compute the retrieval effectiveness of training examples for Boolean Query Ranking. This is because Boolean Query Ranking uses generalizable features while Boolean Query Generation uses terms which strongly depend on the given query documents.

4.4.2 Features

In order to compose a feature vector for our query ranking model, we leverage features from previous studies for predicting query performance (e.g., [27, 42, 113]). The study described in [62] proved that query quality predictors are effective for ranking sub-queries. Since generated Boolean queries also consist of subsets of terms related to query documents, we can expect those quality predictors also help to recognize effective Boolean queries. However, we additionally use more features specialized for our task because we observed that Boolean queries often show different characteristics from adhoc queries. Accordingly, we categorize our features into two groups, General Query Quality Predictors and Boolean Query Quality Predictors. Table 4.1 summarizes the features in each group. General Query Quality Predictors contain

Table 4.1: Two categories of Boolean Query Ranking Features

General Query Quality Predictors	
QCS	Query Clarity Score [27]
QS	Query Scope [42] in pseudo-relevant documents
SOQ	Similarity to Original Query [42]
SCQ	Similarity Collection Query [113]
IDF	Inverse Document Frequency
ICTF	Inverse Collection Term Frequency
Boolean Query Quality Predictors	
BQCB	Boolean Query retrieval list Coverage of Baseline retrieval results
BQS	Boolean Query Scope in pseudo-relevant documents
LBQR	Length of Boolean Query retrieval Results
BQTF	Boolean Query Term Frequency in pseudo-relevant documents

features proposed by previous studies for quality prediction of adhoc queries. Table 4.1 describes these features, named QCS, QS, SOQ, SCQ, IDF, and ICTF. Since Boolean queries show different aspects from adhoc queries for which those features have been proposed, we need to adjust the way these features are computed. For example, since adhoc queries do not contain negation (e.g., $\neg tartar$) in contrast to a Boolean query, we consider terms associated only with conjunctions. SOQ measures cosine similarity between a Boolean query and the baseline query while QS is computed only within pseudo-relevant documents, not within the whole collection because we aim to generate Boolean queries to retrieve pseudo-relevant documents. For IDF, ICTF, and SCQ, as [62] did, we calculate the sum, the standard deviation, the ratio of the maximum to the minimum, the maximum, the arithmetic mean, the geometric mean, the harmonic mean, and the coefficient of variation of each value of a query term. These modified rules are applied to both unigrams and bigrams.

Boolean Query Quality Predictors are features with the purpose of estimating Boolean query quality. All these features except BQTF are related to the retrieval results of a Boolean query because comparing a Boolean query retrieval list with the baseline results is a simple and effective way to predict Boolean query quality.

BQCB is the ratio of the number of documents retrieved by both a Boolean query and the baseline query to the number of documents retrieved by the baseline query. This feature denotes how many of the documents retrieved by the baseline query can be found by a Boolean query. BQS is a measure of the number of pseudo-relevant documents retrieved by a Boolean query relative to the whole size of pseudo-relevant documents, i.e., k. This feature helps to assure the effectiveness of a Boolean query. LBQR measures the number of retrieved documents for a Boolean query. Since we found that an effective Boolean query sometimes returns a shorter result list containing highly relevant documents than the baseline results, we consider this feature as a signal to find such Boolean queries. BQTF counts the frequency of a conjunctive query term in pseudo-relevant documents, assuming that a frequent term in pseudo-relevant documents might be effective for retrieving the documents. Note that we do not consider negation terms because they rarely appear in pseudo-relevant documents. Besides, for BQTF, the same statistics as used for IDF are calculated. Overall, a feature vector contains 37 different feature values (from 10 different types).

4.5 Evaluation

We evaluate our Boolean query generation and suggestion methods by simulating professional search. We first provide the details of experimental setup and then report experimental results and discussion with Boolean query examples.

4.5.1 Experimental Setup

To perform decision tree learning, the C4.5[2] algorithm was used, with pruning turned on to obtain more accurate trees. For Boolean Query Ranking, SVM^{rank} is used as a learning-to-rank algorithm, and 10-fold cross-validation is performed with random partitioning. Queries and documents are stemmed by the Krovetz stemmer

[2]http://en.wikipedia.org/wiki/C4.5_algorithm

[60]. We also conduct the experiments for two domain-specific search tasks: (1) patent search and (2) medical reference retrieval (see Chapter 3.2). Accordingly, we use USPTO and OHSUMED test collections, described in Chapter 3.3. In addition, we adopt PriorArtQuery (described in Chapter 3.5 to generate baseline queries for USPTO and the query likelihood model for the queries in OHSUMED. Indri is used to implement retrieval models, and we assume that the top 100 documents of the baseline retrieval results are pseudo-relevant. To run our Boolean queries, we use the statistical Boolean retrieval model described in Chapter 3.6.

To measure the retrieval effectiveness of each query, we use recall at top 100 (R@100) because the search tasks in the patent and medical domains are known as recall-oriented and the top 100 patents are typically examined in real examination processes as reported in [53]. In addition, the F1-score is used since it can capture both recall and precision simultaneously and help to measure search efficiency. To compare the effectiveness between a Boolean query and baseline query, we use the best recall score of the top n Boolean query suggestions for each query document. As described in Chapter 3.4, the users would sequentially examine the suggested queries from the top 1 to n, and can eventually identify the best one. By doing this, we can figure out the maximum performance that our Boolean queries can achieve, and identify how many suggestions need to be examined to find an effective Boolean query.

Table 4.2: Boolean Query Length Statistics

Collection	Mean	Std. Dev.	Min
USPTO	3.26	1.01	2
OHSUMED	3.14	1.46	2

We also measure each generated query's length (i.e., the number of terms in a generated query), as shown in Table 4.2. Note that we use unigram queries for this statistics, and the terms associated with negation can be counted. In both collections,

generated queries can contain about three terms on average, and the minimum length is 2 (i.e., at least two terms are included in every query). However, the average length of USPTO queries is significantly longer than that of OHSUMED queries (by the Wilcoxon rank-sum test with $p < 0.05$).

To evaluate the Boolean query generation method, we additionally define the following two metrics: (1) Failure Rate and (2) Success Rate.

Failure Rate measures the percentile ratio of "failure" Boolean queries to all generated ones for each query document. Boolean queries which failed to retrieve any target documents are considered as a "failure".

Success Rate measures the percentile ratio of "effective" Boolean queries to the all generated ones, where "effective" means a Boolean query performing identical to or better than the baseline query with regard to R@100. This metric denotes how many Boolean queries achieve the baseline performance.

4.5.2 Results

4.5.2.1 Generation Performance

The first experiment is conducted to verify the effectiveness of Boolean Query Generation. In the USPAT, we generate 4 types of attribute sets; unigrams and unigrams+bigrams from a query patent (i.e., qd), and unigrams and unigrams+bigrams from the pseudo-relevant documents (i.e., D_{prel}). For OHSUMED, only 2 types, unigrams and unigrams+bigrams from the pseudo-relevant set, are used because each provided query contains only a few sentences. Table 4.3 and 4.4 show the performance of Boolean query generation for both domains. We report the average of each evaluation metric over all query documents. Note that we additionally measure the portion of "moderate" queries to all generated queries, which perform moderately (i.e., between success and failure) for retrieving relevant documents.

Table 4.3: Boolean Query Generation Performance for Patent Domain

USPTO				
Evaluation Metric	unigram (qd)	unigram+bigram (qd)	unigram (D_{prel})	unigram+bigram (D_{prel})
Success Rate	7.26%	4.56%	7.84%	4.63%
Avg. ♯ of Success	14.13	9.62	14.32	10.86
Failure Rate	9.47%	7.64%	6.98%	6.19%
Avg. ♯ of Failure	17.25	15.06	11.97	13.28
Moderate Portion	83.27%	83.27%	83.27%	83.27%
Avg. ♯ of Moderate	157.59	180.34	149.57	194.02
Avg. ♯ of Gen.	188.97	205.02	175.86	218.16

Table 4.4: Boolean Query Generation Performance for Medical Domain

OHSUMED		
Evaluation Metric	unigram (D_{prel})	unigram+bigram (D_{prel})
Success Rate	6.18%	5.85%
Avg. ♯ of Success	14.58	16.81
Failure Rate	7.93%	5.85%
Avg. ♯ of Failure	16.37	18.22
Avg. ♯ of Gen.	206.53	238.43

Our decision tree-based generation algorithm can generate a substantial number of distinct Boolean queries. About 200 queries are generated for each query document, of which 6 to 9% fail to retrieve any target documents. In USPAT, pseudo-relevant documents are more reliable resources to generate Boolean queries than query patents because of the smaller failure rate on average. Also, adding bigrams can lead decision trees to generate more queries, and the relative failure rate could drop. However, bigrams seems to be harmful in terms of the success rate. In addition, considering the number of "effective" Boolean queries (the number of successes), about 7% of queries show better or equal performance to the baseline query. Although this percentage may look low, we obtain many effective queries via this generation process. Indeed, as you see from the number of successes, more than 10 effective queries are generated

for each query document. If we can place these effective queries at top ranks using our Boolean query ranking method, search users who examine these suggestions will find the effective queries. We address the performance of the query ranking technique in the following section.

4.5.2.2 Retrieval Performance

In the next series of experiments, we evaluate the effectiveness of Boolean Query Ranking by investigating if it succeeds in placing effective Boolean queries at high ranks, i.e., top 1 to 10. In training, generated queries for each query document are ordered by their R@100 scores. For the evaluation, we compare the best-performing query among the top 1 to 10 suggestions with the baseline query for each query document. Thus, we calculate R@100 and F1@100 of the results obtained by the best-performing query and baseline query.

Table 4.5 shows the retrieval results within the top 1 to 10 ranked Boolean queries by 10-fold cross validation. In the table, a † indicates a significant difference from the baseline and a $*$ denotes a significant difference of unigram results from unigram+bigram results in each row (the paired t-test is performed with $p < 0.05$). In addition, significantly improved results over the baseline in each column are marked in bold, and "cut-off" indicates that all Boolean queries ranked within the cut-off ranks are examined. From this table, we can identify how many top n Boolean queries need to be examined to find an "effective" one (i.e., performing as well as the baseline). In other words, the results of the top n queries which are not significantly different from the baseline result show that at least one effective Boolean query can be within the top n suggestions.

In Table 4.5, we see that effective Boolean queries can be found within the top 2 or 4 suggestions in each corpus. In USPAT, an effective Boolean query is observed within the top 2 ranks in both unigram and unigram+bigram cases. Furthermore, in

Table 4.5: Boolean Query Ranking Performance

Domain	**Metric**	**Recall@100**		**F1@100**	
USPTO	Baseline	0.2557		0.1184	
	cut-off	unigram	unigram+ bigram	unigram	unigram+ bigram
	1	0.2227†	0.2174†	0.1062†	0.0969†
	2	0.2538	0.2445	0.1204	0.1096
	3	0.2670	0.2529	0.1264	0.1166
	4	0.2761	0.2535	0.1303	0.1169
	5	0.2820	0.2592	**0.1330**$^{\dagger *}$	0.1191
	6	0.2852	0.2597	**0.1345**$^{\dagger *}$	0.1194
	7	**0.2883**$^{\dagger *}$	0.2622	**0.1359**$^{\dagger *}$	0.1209
	8	**0.2911**$^{\dagger *}$	0.2695	**0.1370**$^{\dagger *}$	0.1257
	9	**0.2952**$^{\dagger *}$	0.2710	**0.1388**†	0.1265
	10	**0.2991**$^{\dagger *}$	0.2722	**0.1402**†	0.1277
OHSUMED	Baseline	0.4377		0.2636	
	cut-off	unigram	unigram+ bigram	unigram	unigram+ bigram
	1	0.3068†	0.3052†	0.2155†	0.2222†
	2	0.3618†	0.3611†	0.2490	0.2580
	3	0.3865†	0.3754†	0.2669	0.2774
	4	0.3970	0.3923†	0.2763	0.2874
	5	0.4009	0.4032	0.2836	0.2944
	6	0.4137	0.4082	0.2959	**0.3042**†
	7	0.4141	0.4106	**0.2961**†	**0.3045**†
	8	0.4143	0.4170	**0.2963**†	**0.3046**†
	9	0.4393	0.4232	**0.3076**†	**0.3169**†
	10	0.4411	0.4232	**0.3089**†	**0.3185**†

the unigram case, significantly improved results in terms of R@100 can be obtained by examining 7 or more Boolean queries. This is surprising to us because we expected Boolean queries to perform similar to the baseline. However, the result is a good indication that our method provides effective suggestions. In terms of F1, the top 5 unigram queries contain the queries that can outperform the baseline. These suggested queries retrieve about the same number of relevant documents as the baseline results, but with higher precision. That is, these Boolean queries may be more efficient in that they can allow users to examine fewer documents. On the other hand,

effective queries are not successfully generated in the case of unigram+bigram. For example, the number of generated effective queries in the unigram+bigram case is smaller than in the unigram case as seen in Table 4.3. Furthermore, many unigram results show statistically significant improvements over the unigram+bigram results, when comparing query performance at the same top n suggestions.

In OHSUMED, more queries need to be examined to find effective Boolean queries compared to USPAT. For example, four query suggestions should be examined in the unigram case in the OHSUMED, while only two queries are needed in the USPAT. Furthermore, even more queries should be examined in the unigram+bigram case. In addition, we could not obtain significantly better Boolean queries with respect to R@100 in this domain. For the F1-score, however, we also identify more efficient Boolean queries by examining the top 6 or 7 queries. A critical difference between OHSUMED and USPTO is that there is little distinction between unigram and unigram+bigram results in the OHSUMED while unigram queries are consistently better than unigram+bigram queries in the USPTO. Overall, our ranking model is effective in placing "effective" Boolean query suggestions within the top 2 to 5 ranks.

4.5.2.3 Qualitative Analysis

We now provide a qualitative analysis of our system via real examples. Table 4.6 shows the top 5 Boolean queries suggested by our method, for a sample query document in USPAT. The title of the sampled query document is "compressor driving apparatus", and, in the table, ♯Ret. indicates the number of documents retrieved by each query. For this sample query document, the baseline query shows moderate performance (0.30 for R@100, 0.10 for F1@100), and some suggestions generated by our method can outperform the baseline. Many Boolean queries retrieve less than 100 documents, and some long suggestions (e.g., $powerunit \wedge air\ \ conditioner \wedge output \wedge inverter \wedge circuit$) can precisely retrieve relevant documents in the short result

Table 4.6: Examples of Generated Boolean Queries

Rank	Unigram Queries	R@100	F1@100	♯Ret.
1	$inverter \wedge compressor$	0.35	0.12	100+
2	$inverter \wedge compressor \wedge circuit$	0.55	0.18	100+
3	$inverter \wedge motor$	0.05	0.02	100+
4	$\neg inrush \wedge \neg metallic \wedge inverter \wedge$ $compressor \wedge relay$	0.10	0.04	72
5	$\neg inrush \wedge \neg metallic \wedge \neg board \wedge circuit$ $\wedge compressor \wedge supply \wedge inverter$	0.25	0.13	58
Rank	**Unigram+Bigram Queries**	**R@100**	**F1@100**	**♯Ret.**
1	$\neg inverter\ \ driving \wedge inverter \wedge$ $compressor \wedge circuit$	0.20	0.07	87
2	$inverter \wedge air\ \ conditioner \wedge circuit$	0.50	0.17	100+
3	$\neg power\ \ unit \wedge air\ \ conditioner \wedge output$ $\wedge inverter \wedge circuit$	0.55	0.25	68
4	$\neg relay\ \ driver \wedge \neg compressor\ \ driving \wedge$ $inverter \wedge circuit$	0.05	0.03	48
5	$switching\ \ elements \wedge air\ \ conditioner$	0.30	0.10	100+

lists. Several suggestions return significantly more relevant documents. The suggested Boolean queries can provide reasonable query contexts. For example, "compressor" is often combined with "inverter", "supply", "circuit" in Table 4.6 because compressor driving apparatus can include power supply, inverter drivers and storage circuits. Moreover, looking at the negated terms, professional searchers can recognize where negation is applied in the provided context. For example, "power unit" is negated when it comes with "air conditioner", "output", "inverter", and "circuit". Since we found that past cited patents are dealing with inverters or circuits for air conditioners, power supplies can be considered less important.

4.6 Summary

In this chapter, we proposed a framework to automatically generate and suggest Boolean queries to assist professional users. We assume that many domain-specific search tasks are interactively performed by information professionals. In our method,

we first generate Boolean queries by exploiting decision tree learning and pseudo-relevant documents. To provide a reasonable number of suggestions, we rank the generated queries by a query ranking model using query quality predictors. In the evaluation, we found that our method can not only generate many effective Boolean queries but also select highly effective queries for suggestion.

CHAPTER 5

PHRASAL-CONCEPT QUERY GENERATION

5.1 Overview

Academic literature search (e.g., finding relevant research papers) is one of the most promising domains that can be helped by query generation. In this domain, the typical users are scientists, and they need to find existing articles relevant to their current work. Since a scientific study is related to a number of research topics, people typically use many queries for retrieving a comprehensive list of related papers. In this situation, query generation can reduce the burden of formulating effective queries and the complexity of the search by providing effective query examples. In addition, sometimes scientists need to find relevant papers outside their specific area of expertise, and generated queries can be a good guideline for exploring new areas.

To develop effective queries for literature search, we need to consider its unique characteristics. In contrast to general web search, the literature search task is carried out in a very specific environment, and a query generation method should be designed for the unique characteristics of that environment. One unique characteristic is that phrasal concepts and terminology (e.g., "lexicon acquisition using bootstrapping") are frequently used as keywords in target documents (i.e., research papers). Since scientists use longer technical terms to describe their research ideas, phrasal concepts are frequently observed in academic writing. It follows that queries that emphasize phrasal concepts should be more effective for discriminating relevant documents from non-relevant documents in retrieval. In addition, typical users of literature search may prefer using phrasal-concept queries because phrases and terminology tend to have

clear meanings, and users can more easily understand the areas that the generated queries are targeting.

Given that phrasal concepts are important for literature search, we propose a query generation method that can formulate phrasal-concept queries by exploiting pseudo-labeled documents. We first define relevant terms and problems for phrasal-concept query generation, and then describe our method to generate and suggest phrasal-concept queries. For evaluation, the ideal situation is that scientists provide their research descriptions as initial queries, and relevant articles are identified by asking the same scientists. However, no such data is available, and there have been alternatives proposed to automatically generate evaluation data from existing citation databases (e.g., [88]). For example, He et al. [44] developed an initial query using the sentences containing citations from a published paper, and regard the citations as the relevant articles. This approach favors a local recommendation because it only considers local contexts of the query paper (i.e., published paper) [43]. On the other hand, the settings used in [13, 96] assume that the abstract and title of the query paper are a research summary written by the user, and the list of references cited in the paper is the set of relevant documents. This method uses the global context of the query paper for retrieval, and we adopt this approach in our work.

Furthermore, we evaluate our phrasal-concept query generation method based on user preference as well as retrieval effectiveness. We conduct user experiments to verify that users prefer the queries generated by our technique, compared to other effective query generation and query expansion methods. To assess the retrieval effectiveness of our method, we compare the retrieval performance to other query expansion methods in simulated literature search environments.

5.2 Problem Formulation

Definition 1. (Baseline Query): Given an initial query (e.g., a summary of a research work), a baseline query is its improvement by state-of-the-art query expansion methods (e.g., LCE and RM). We exploit the baseline query to generate more effective phrasal-concept queries.

Definition 2. (Phrasal Concept): A phrasal concept is a syntactic expression recognized as a noun phrase in a document. Syntactic phrases will be more recognizable to users in general than arbitrary n-grams (e.g., bigrams and trigrams). In addition, noun phrases are suitable for representing important "concepts" in academic papers (e.g., technique names such as "Markov Random Field"), and noun phrase concepts have been shown to be effective for improving retrieval effectiveness [11]. In our work, we use the terms phrasal-concept and concept, interchangeably.

Definition 2. (Key Concept and Related Concept): A key concept is an effective phrasal-concept for finding relevant documents, and a related concept is a phrasal-concept related to a key concept, which helps users to understand the key concept better. For example, "text classification via WordNet" can be a key concept, and "Support Vector Machine" and "WordNet similarity feature" could be related concepts. A key concept can have multiple related concepts, and to measure the relation between a concept and the key concept, various statistical similarity measures can be used (see Chapter 5.3).

Problem 1. (Key Concept Identification): Given a set of phrasal concepts, key concept identification is ranking the concepts by their estimated retrieval effectiveness, i.e., highly ranked concepts are predicted to be more effective for retrieving relevant documents. We assume that the top n ranked concepts are the key concepts.

Definition 3. (Phrasal-Concept Query): A phrasal-concept query is a combination of a key concept and a set of related concepts. To improve the understandability

of each suggestion and maximize retrieval performance, we include only a single key concept and its related concepts in a phrasal-concept query.

Problem 2. (Phrasal-Concept Query Suggestion): Phrasal-concept query suggestion is suggesting a list of phrasal-concept queries to users. We suggest up to n queries which are sorted in descending order of predicted retrieval effectiveness of their key concepts. Since the key concepts in Problem 1 are ranked by their predicted retrieval effectiveness, we can address this problem by solving Problem 1.

5.3 Phrasal-Concept Query Generation

Phrasal-concept queries, which explicitly specify important phrases, are effective and useful for academic literature search. Given a query paper (e.g., a summary of a new research project), we generate a list of n phrasal-concept queries in the following steps:

Step-1 : Generate a baseline query and gather the pseudo-relevant documents of the baseline query.

Step-2 : Extract candidate concepts from the pseudo-relevant documents.

Step-3 : Identify n key concepts by ranking the candidate concepts using the baseline query. Related concepts may be also extracted.

Step-4 : Construct a list of n concept queries as query suggestions.

Given a query document, the first step is generating an effective baseline query. For this, we can use existing query expansion methods (e.g., LCE and RM) for generating more improved queries. Since we assume that the users simply input a bag of words (describing a new research idea) as an initial query, such an initial query may perform poorly and may not be helpful for obtaining effective pseudo-relevant documents where phrasal concepts are extracted in the next step. To alleviate this,

we use query expansion methods to generate a more effective set of pseudo-relevant documents. The query weighting schemes corresponding to the expansion method can also be applied. To formulate better baseline queries, we conducted preliminary experiments with several query expansion and generation methods and found that the LCE and MLE (described in Chapter 3.5) performed significantly better in our search environments; we provide more details about this in Chapter 5.4. Once a baseline query is formulated, we can obtain the top k pseudo-relevant documents from the retrieval results.

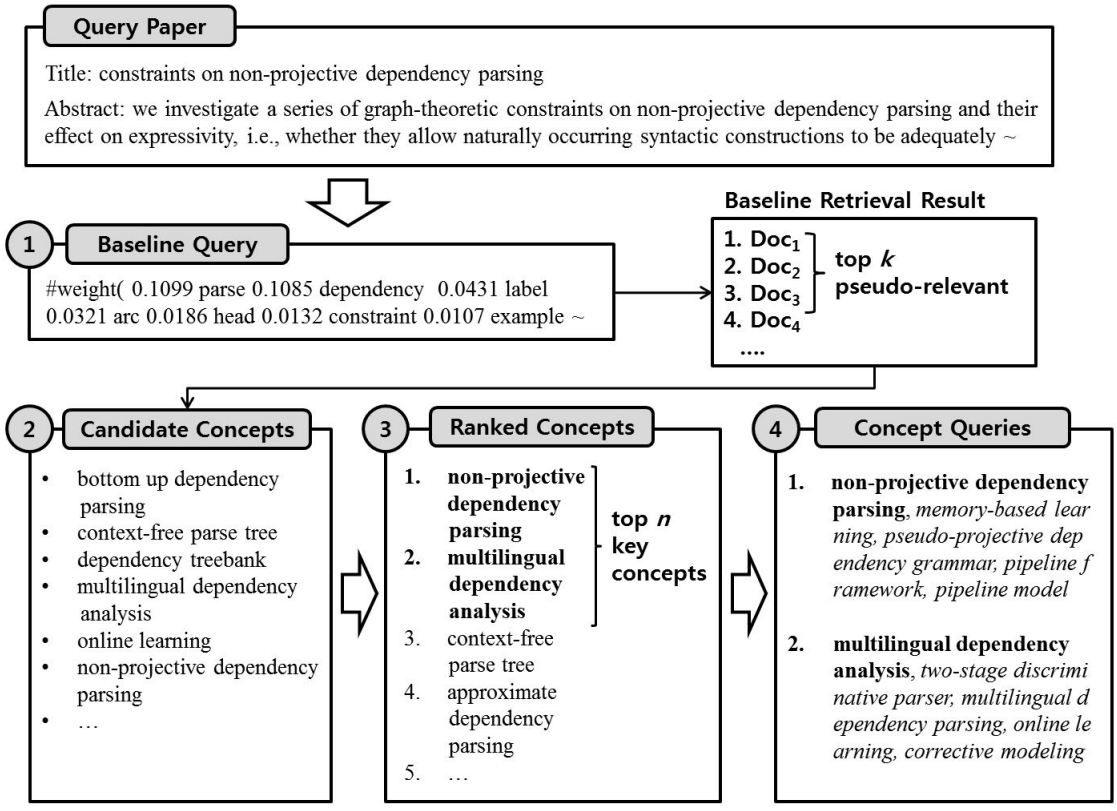

Figure 5.1: Phrasal-Concept Query Generation Example

Next, we extract candidate (phrasal) concepts by ranking the phrases recognized from the pseudo-relevant documents. Then, in the third step, we rank the candidates with respect to their retrieval effectiveness predicted from the baseline query terms.

After ranking, we assume that the top n (phrasal) concepts are key concepts, and combine each key concept with the related concepts that have high co-occurrence with the key concept. Finally, we can construct a list of phrasal-concept queries, each of which includes a single key concept and multiple related concepts. Figure 5.1 shows an example of phrasal-concept query generation following this process, and the details of each step are described in the following sections.

5.3.1 Extracting Candidate Phrasal-Concepts

In the second step, we collect candidate (phrasal) concepts used for identifying key concepts and their related concepts. By retrieving documents with the baseline query, we obtain pseudo-relevant documents, and then use them to extract candidate phrasal-concepts. Instead of using the pseudo-relevant documents, we can directly extract candidate concepts from only query documents. However, in academic literature search, query documents can be relatively short (e.g., a few paragraphs for describing new research projects) and more effective concepts may not be observed by a small pool of candidate concepts, derived from only query documents; typically pseudo-relevant documents could provide more effective terms for retrieval (e.g., [64, 38, 80]). In experiments, we generate a query document by concatenating title and abstract text, and most concepts in such a query document are appeared in pseudo-relevant documents.

As we consider a noun phrase (NP) as a phrasal concept (*Def. 2*), we apply an NP recognizer[1] to the pseudo-relevant documents. However, due to the long length of academic articles (such as journal papers), too many phrasal-concepts are recognized from the whole text of an article. Therefore, to reduce the size of the candidate set, we assume that a title and abstract contain important phrasal-concepts which can represent the whole article. Accordingly, we can generate two different candidate sets:

[1]Montylingua (http://web.media.mit.edu/ hugo/montylingua/)

(i) all phrasal-concepts from only the titles of pseudo-relevant documents, and (ii) N important phrasal-concepts from titles and abstracts of pseudo-relevant documents; among all the recognized phrasal-concepts, we can use n-gram language models to estimate the importance of each phrasal-concept recognized from the titles and abstracts of pseudo-relevant documents. In the evaluation, we use 300 phrasal-concepts extracted by using trigram language models. The ranking function based on this model is given as:

$$\begin{gathered} P(w_1 w_2 \dots w_l) \approx \prod_{i=3}^{l} P(w_i | w_{i-2} w_{i-1}) \\ P(w_i | w_{i-2} w_{i-1}) \approx \lambda_1 P(w_i | w_{i-2} w_{i-1}) + \lambda_2 P(w_i | w_{i-1}) + \lambda_3 P(w_i) \end{gathered} \tag{5.1}$$

where $w_1 w_2 \dots w_l$ is a concept whose word-length is l and λ_j is a bias to each language model.

To avoid the sparseness problem, the trigram language models are smoothed by bigram and unigram language models, and for each model we use maximum likelihood estimations based on term frequencies in the pseudo-relevant documents. We empirically set the biases as $\lambda_1 = 0.7, \lambda_2 = 0.2$, and $\lambda_3 = 0.1$. If a phrasal-concept is longer than a trigram, we identify multiple trigrams from the phrasal-concept (see the first part of Eq. (5.1), and take a product of the probability of each trigram to estimate the probability of the whole concept.

5.3.2 Identifying Key Phrasal-Concepts

After collecting candidate phrasal-concepts, we identify key concepts by ranking the candidate (phrasal) concepts with respect to their predicted retrieval effectiveness. Given a set of candidate concepts and the baseline query, we assume that the concepts more similar to the baseline query will be more effective because the baseline query is effective for retrieving relevant documents. As an example, in Figure 5.1, the query document describes some graph-theoretic constraints for non-projective dependency

parsing. In the baseline query, "dependency" and "parse" are effective keywords and highly weighted, and we can infer that among many related phrasal-concepts for this paper, "non-projective dependency parsing" is one of the most important phrasal-concepts. Since this phrasal-concept intuitively looks very similar to the keywords in the baseline query (i.e., "dependency" and "parse"), it may have higher retrieval effectiveness. To identify this phrasal-concept as a key concept, we use the similarity between the phrasal concept and keywords. Thus, in ranking, we place the phrasal concepts more similar to many baseline query terms at higher ranks, and the highly ranked phrasal concepts are regarded as "key concepts". To do this, we use the label propagation algorithm [114] where the labels (effectiveness) of the baseline query terms are propagated to the candidate concepts through a similarity matrix which defines the similarities between the candidate concepts and baseline query terms.

Algorithm 2 Phrasal-Concept Ranking

Input:

V is an input set divided into two sub-sets: the set of baseline query terms, $V_b \subset V$, and the set of candidate phrasal-concepts ,$V_c \subset V$

Y is a label vector divided into two sub-sets: the set of baseline query terms, $Y_b \subset Y$, and the set of candidate phrasal-concepts, $Y_c \subset Y$

W is a similarity matrix which defines the similarities between $\forall v_i, \forall v_j \in V$

t is the number of iterations

Output:

V_c is the ranked list of candidate concepts

1: Let D be a diagonal and row sum matrix of W
2: Initialize $Y^{(0)} = [Y_b, Y_c]$ where $\forall y_b \in Y_b, y_b = 1$ and $\forall y_c \in Y_c, y_c = 0$
3: **for** $i = 0$ to $t - 1$ **do**
4: Calculate $Y^{(i+1)} = D^{-1} \cdot W \cdot Y^{(i)}$
5: **end for**
6: Sort $Y_c^{(t)} \subset Y^{(t)}$ in decreasing order
7: **return** the list of V_c where the ranking of $v_c \in V_c$ corresponds to the order of $y_c \in Y_c^{(t)}$

Suppose that we construct two vectors: (i) the vector of baseline query terms, V_b, and (ii) the vector of candidate phrasal-concepts, V_c. Define a term vector, V, as $V =$

$[V_b, V_c]$ and construct a label vector $Y = [Y_b, Y_c]$ where each $y_b \in Y_b$ is mapped to each $v_b \in V_b$ and each $y_c \in Y_c$ is mapped to each $v_c \in V_c$, i.e., $(v_1, y_1), (v_2, y_2), \ldots, (v_m, y_m)$ where $m = |V| = |Y|$. In addition, we define a $|V| \times |V|$ similarity matrix, W which represents the similarities between $\forall v_i$ and $\forall v_j$, i.e., $W[i, j] = sim(v_i, v_j)$. To calculate $sim(v_i, v_j)$, we can use one of the following similarity measures.

Point-wise Mutual Information (PMI) is a statistical measure which quantifies the discrepancy between the co-occurrence probability in the joint distribution of v_i and v_j where the co-occurrence probability is estimated using their individual distributions. Using a corpus, the PMI of two terms (i.e., v_i and v_j) is calculated as:

$$\text{PMI}(v_i, v_j) = \log \frac{P(v_i, v_j)}{P(v_i)P(v_j)} \approx \log \frac{\text{df}(v_i, v_j) \times N}{\text{df}(v_i)\text{df}(v_j)} \tag{5.2}$$

where $v_i, v_j \in V$, df( ) denotes the document frequency in a corpus, and N is the number of all documents in the corpus.

Chi-square statistics (χ^2) is a statistical method that determines whether v_i and v_j are independent by comparing the observed co-occurrence frequencies with the expected frequencies assuming independence.

$$\chi^2(v_i, v_j) = \frac{(a \times d - b \times c)^2 \times N}{(a + b) \times (a + c) \times (b + d) \times (c + d)} \tag{5.3}$$

where $a = \text{df}(v_i, v_j)$, $b = \text{df}(v_i) - a$, $c = \text{df}(v_j) - a$, and $d = N - a - b - c$.

Likelihood (LK) measures the likelihood of v_j to v_i, i.e., how much v_j can be generated from v_i. The calculation is given as:

$$\text{LK}(v_i, v_j) = P(v_j|v_i) \approx \frac{\text{df}(v_i, v_j)}{\text{df}(v_i)} \tag{5.4}$$

Unlike the other measures, LK is directional, i.e., $\text{LK}(v_i, v_j) \neq \text{LK}(v_j, v_i)$.

With V, Y, and W, we perform the phrasal concept ranking algorithm (Alg. 2) which produces a ranked list of the candidate (phrasal) concepts. In ranking, an initial output vector $Y^{(0)}$ contains Y_b corresponding to V_b and Y_c corresponding to V_c where the values of Y_b are 1.0 which indicates "labeled" (the highest retrieval effectiveness) and the values of Y_c are 0 which indicates "unlabeled". Given a number of iterations (i.e., t), the propagation runs iteratively, and the values of Y_c of phrasal concepts more similar to the baseline query terms may have higher values than the others less similar to the baseline query terms. Since t is a controlling parameter, if an excessively high value of t is input, too many propagations are executed, and the values of $\forall v \in V$ would be converged, i.e., the values of all candidate concepts are equal. Therefore, an appropriate value of t can be found by retrieval experiments (described in Chapter 5.4). After t iterations, the algorithm ranks V_c by the corresponding values of Y_c, and the phrasal concepts with greater values are placed at higher positions in the output list. In the output list, we assume that the top n phrasal concepts are "key concepts".

After identifying key concepts, we extract related concepts for each key concept. Since a similarity measure (e.g., PMI) can be defined between two phrasal concepts, we use it to extract "related concepts" among all candidate phrasal-concepts. In extraction, for each key concept, v_{KC}, we determine the set of "related concepts", V_{RC}, as:

$$V_{RC} = \{v | \text{sim}(v_{KC}, v) > \theta\} \tag{5.5}$$

where θ is the cut-off value, v_{KC} is a key concept, v is a candidate phrasal-concept, $v_{KC} \neq v$. In the experiments, we empirically set θ as 0.01, 0.02, and 0.01 for PMI, χ^2, and LK, respectively.

Note that key concepts are identified as highly effective for retrieval, whereas related concepts are just strongly related to a key concept and provide additional context to the key concept for the users.

5.3.3 Constructing Phrasal-Concept Queries

Given the top n key (phrasal) concepts, we construct n phrasal-concept queries by associating each key concept with its related concepts. As defined in Chapter 5.2, we ensure that a phrasal-concept query contains only a single key concept because a long query which contains several key concepts may be too complex to understand as a query suggestion. In addition, to further simplify the suggestions, we select the l most related concepts in the set of related concepts, V_{RC} (see Eq. (5.5)). In the experiments, we empirically set l as 4, i.e., we make a query contain at most 5 phrasal-concepts including a key concept. Finally, the n phrasal-concept queries are suggested to users, where each query is formed as $< Key\ Concept, Related\ Concept_1, Related\ Concept_2, \dots >$. The queries are listed in descending order of predicted retrieval effectiveness of their key concepts.

5.4 Retrieval Experiments

5.4.1 Experimental Setup

We implement retrieval experiments that simulate the processes of literature search based on the assumptions described in Chapter 3.4. For the experiments, we conduct two different search tasks considering the academic and medical domains, and accordingly two test-collections, i.e., ACL and OHSUMED (described in Chapter 3.3), are adopted. To develop initial queries, we use the title and abstract of each query paper. To measure retrieval performance, we use MAP and NDCG. In addition, the multi-query session-based metric, NSDCG, is used for optimizing the proposed method that generates multiple queries for a query document. To run phrasal-concept queries, we implement the learning-to-rank retrieval model described in Chapter 3.6, and 16 features (listed in Table 3.1) are used. In addition to this, we create four concept-specific features because we generate phrasal-concept queries and can improve the impact of the concepts in these queries. Table 5.1 describes

these concept-specific retrieval features, and overall 20 features are used for running phrasal-concept queries.

Table 5.1: Concept-specific Retrieval Features

Feature	**Description**
exit-key-concept(q, d)	binary feature which returns 1 if the target document, d, contains the key concept of the query, q; otherwise, returns 0
exit-all-concepts(q, d)	binary feature which returns 1 if d contains all concepts of q; otherwise, returns 0
loglike-key-concept(q, d)	log-likelihood of q for d, estimated only by the key concept of q. $\log P(q\|d) \approx \log freq(kc, d)/(l_d - l_{kc} + 1)$ where kc is a key concept of q, l_d is the length of d ($\sharp$ of words in d) and l_{kc} is the length of kc ($\sharp$ of words in kc).
loglike-all-concepts(q, d)	log-likelihood of q for d, estimated by every concept of q. $\log P(q\|d) \approx \sum_{c \in q} \log freq(c, d)/(l_d - l_c + 1)$ where c is a concept of q, l_d is the length of d ($\sharp$ of words in d) and l_c is the length of c ($\sharp$ of words in c).

5.4.2 Baseline Query Investigation

In order to adopt more robust baselines, we conduct a preliminary experiment. Among many successful methods to generate effective queries for initial queries (e.g., [11, 35, 47, 63, 79, 80], we select several methods using pseudo-relevance feedback (i.e., RM and MLE) and dependence models (i.e., MRF and LCE), which can emphasize concepts that are important in a search query and more applicable to academic search environments. We use the Indri search engine to implement each method (for MLE and MLE-P, least-angle regression [31] is used), and the initial query uses the query-likelihood (QL) model [84]. For LCE, we use unigrams for the feedback, which performs better than using bigrams. In addition, 3-fold cross-validation is performed to find optimal parameters (e.g., the number of feedback terms) for each model. We use the top 30 and 100 retrieval results for measuring retrieval performance, and Table 5.2 shows the retrieval results for each method. Note that statistically significant

improvements are marked using the last letter of each method, e.g., D indicates a significant improvement over SD, and the paired t-test is performed with $p < 0.05$.

Table 5.2: Baseline Retrieval Results

Method \Metric	**MAP**	**NDCG@30**	**NDCG@100**
SD	0.1201	0.2488	0.2905
QL	0.1228	0.2507	0.3019
RM	0.1317^{DL}	0.2587^{D}	0.3106^{D}
MLE-P	0.1331^{DL}	0.2530	0.3220^{DL}
LCE	0.1354^{DL}	0.2624^{DL}	0.3243^{DLE}
MLE	0.1470^{DLEPM}	0.2773^{DLEPM}	0.3411^{DLEPM}

In Table 5.2, the dependence model (SD) performs badly because two term dependencies are less effective for capturing longer academic concepts. Moreover, since we use an entire title and abstract for an initial query, in such a long query, we observed that many unreliable dependencies are constructed, which is harmful for retrieval. As an example, "$\sharp 1(task\ provide)$" and "$\sharp 1(provide\ empirical)$" are formed from the query of "*Experiments on the classification task provide empirical support for the qualitative and relational ...*". Note that "$\sharp N(\ldots)$" indicates an ordered window which means that terms must appear ordered, with at most $N-1$ terms between any terms , e.g., "$\sharp 1(task\ provide)$" matches "*task provide*" as an exact phrase. In addition, the methods using PRF (LCE, RM, MLE-P, and MLE) can outperform SD and QL. However, MLE-P is less effective than MLE because it is hard to find discriminative phrases which are commonly shared only within positive documents, compared with unigrams. In other words, the number of discriminative phrases, which only appear in many pseudo-relevant documents (not frequent in many non-relevant documents), is much smaller than the number of discriminative unigrams. Since MLE and LCE can perform better than the others, we choose them to formulate the baseline queries for each query document.

5.4.3 Optimizing Parameters

Before the evaluation, we optimize the parameters of our method. In the phrasal-concept ranking algorithm (Alg. 2), the number of iterations and a similarity measure which defines a similarity matrix can influence the determination of key phrasal concepts. In addition, for academic literature search, we can use two different sets of candidates for ranking: (i) phrasal concepts only from titles of pseudo-relevant documents, and (ii) phrasal concepts from titles or abstracts of pseudo-relevant documents (see Chapter 5.3). Thus, we test with different numbers of iterations, combinations of 2 candidate sets, and three different similarity measures. However, for medical reference retrieval, we use all phrasal concepts identified from pseudo-relevant documents because the OHSUMED collection does not provide section information, but the three different similarity measures can be tested.

Figure 5.2 depicts the average NSDCG@100 over 1 to 20 iterations using the ACL collection. 'TTL' indicates concepts from the titles of pseudo-relevant documents, and 'TTL+ABST' means concepts from the titles and abstracts of pseudo-relevant documents. Besdies, LK, PMI, and χ^2 denotes the likelihood, PMI, and Chi-Square similarity measures, respectively. Indri is used to run the queries generated from each setting, and 3-fold cross-validation is applied. For each session, we generate 10 phrasal-concept queries using the 6 different combinations. First, as the number of iterations increases, the performance reached a peak and afterward slightly decreases. Second, among the three proposed similarity measures, LK (likelihood) shows significantly better performance than PMI and χ^2. Third, the queries using the concepts from titles only (TTL) can reach the maximum more quickly and are slightly better than the queries using the concepts from titles or abstracts (TTL+ABST). This is because, in many papers, titles are sufficiently expressive while the abstract is often more verbose and noisy. To find an optimal combination, we compared the average NSDCG@100 of every combination, and the queries generated using TTL, LK and 5

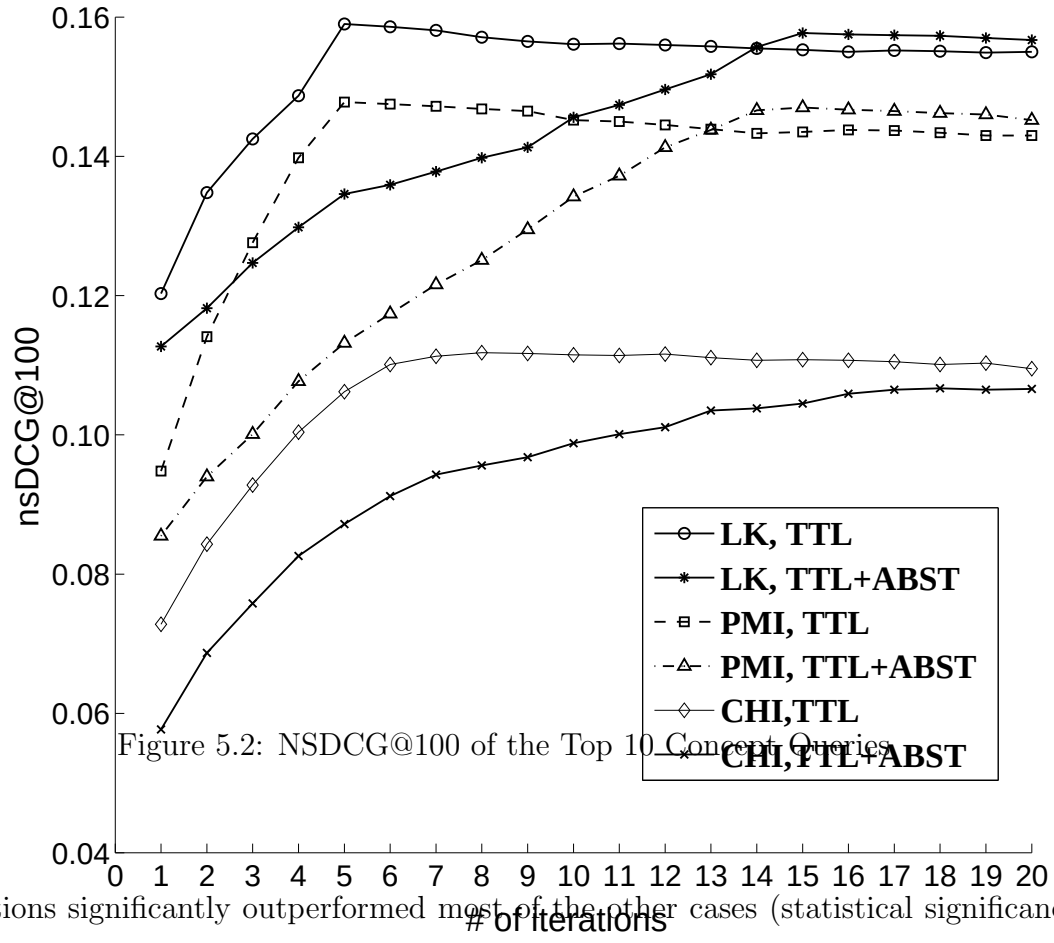

Figure 5.2: NSDCG@100 of the Top 10 Concept Queries

iterations significantly outperformed most of the other cases (statistical significance in $p < 0.05$). Experiments using the OHSUMED collection showed similar tendencies.

5.4.4 Retrieval Results

With the optimized parameters, we verify the retrieval effectiveness of our method on the two different search tasks. We use 3-fold cross-validation for evaluations, and LCE and MLE queries are used as baselines. As another baseline, we can consider the n-gram suggestion method (NGram). However, we do not use it for this experiment

because NGram focuses on finding relevant phrases for an initial query rather than improving their performance. Instead, we use that for user experiments (see Chapter 5.5). Besides, since the query expansion methods can significantly outperform n-gram suggestion in retrieval effectiveness, they can provide stronger baselines for retrieval experiments.

For academic literature search, we use the 20 features described in Table 3.1 & 5.1 for our phrasal-concept queries, and the 16 features (Table 3.1) for baseline queries since the baseline queries do not contain phrasal concepts so we cannot use the four concept-specific features (Table 5.1). In the experiments with medical reference retrieval, we only use query-based features among the features in Table 3.1 because OHSUMED does not provide the meta information that is essential to implement non-query features (i.e., Age, Citation, Citation Pattern, and, Author Citation Behavior in Table 3.1). So, only 5 features (i.e., Query in Table 3.1) are used with LCE and MLE queries, and four concept-specific features are additionally included for phrasal-concept queries in OHSUMED experiments.

To compare the performance between our method (PHRASAL-CONCEPT) and the baseline, we use the best average precision scores of the top 1 to 10 ranked phrasal-concept queries for each session, e.g., if the users browse the top 10 suggestions, we select the best query whose average precision score is the highest. Since our method generates multiple queries for a session, we select a single best query by the assumption that users examine the search results by all the top n queries and identify the best query among them. In other words, we report an upper bound of the performance achieved by our method. Since authors sometimes need to use many queries to explore more relevant articles to their papers, browsing all of the top n suggestions is not unusual, and they can subsequently recognize the most effective query among them. Besides, the baseline method can only generate a single best query, and the metric for multiple-query session (i.e., NSDCG) is not applicable.

Table 5.3: Best Query Retrieval Results for ACL and OHSUMED

Collection		**ACL**		**OHSUMED**	
Method		NDCG@100	MAP	NDCG@100	MAP
LCE		0.4874	0.2638	0.4321	0.2748
MLE		0.5086	0.2744	0.4249	0.2660
PHRASAL CONCEPT	Top 1	0.5301^{LM}	0.2899^{LM}	0.4328	0.2812^{M}
	Top 2	0.5471^{LM}	0.3073^{LM}	0.4865^{LM}	0.3398^{LM}
	Top 3	0.5626^{LM}	0.3211^{LM}	0.5236^{LM}	0.3737^{LM}
	Top 4	0.5715^{LM}	0.3294^{LM}	0.5387^{LM}	0.3865^{LM}
	Top 5	0.5780^{LM}	0.3364^{LM}	0.5505^{LM}	0.3973^{LM}
	Top 6	0.5833^{LM}	0.3426^{LM}	0.5601^{LM}	0.4058^{LM}
	Top 7	0.5873^{LM}	0.3473^{LM}	0.5643^{LM}	0.4097^{LM}
	Top 8	0.5909^{LM}	0.3497^{LM}	0.5695^{LM}	0.4145^{LM}
	Top 9	0.5933^{LM}	0.3518^{LM}	0.5748^{LM}	0.4183^{LM}
	Top 10	0.5941^{LM}	0.3546^{LM}	0.5791^{LM}	0.4228^{LM}

Table 5.3 shows the average NDCG@100 and MAP of the results obtained by the best-performing query within the top 1 to 10 suggestions. In each column, a statistically significant improvement is marked using the first letter of each baseline method, e.g., M denotes a significant improvement over MLE. Note that the paired t-test is performed with $p < 0.05$. First, in ACL, from the first suggestion, users can find an effective phrasal-concept query which can significantly outperform any baselines. Second, in OHSUMED, users need to examine the top two or more queries to find an effective phrasal-concept query that can perform significantly better than the best baseline (i.e., LCE). Third, phrasal-concept queries are significantly better than the baselines in most cases. Unlike the baseline queries, phrasal-concept queries can exploit the concept-specific features, and this leads to significant improvements over the baselines. For example, in Table 5.4, phrasal concepts in the concept query can effectively work with the concept-specific features for retrieval, whereas those features are not applied to the baseline query. This result is quite significant because we can identify that phrasal concepts can be new effective features for the literature search task, and are complementary to the previously developed features.

Table 5.4: Initial Query Example

Initial Query	**Title**: acquisition of verb entailment from text **Abstract**: the study addresses the problem of automatic acquisition of entailment relations between verbs. while this task has much in common with paraphrases acquisition which aims to discover ...
Baseline Query	verb, emnlp, acquisition, entailment, semantic, pantel, related, text, deepak, value, special, grenoble, taxonomy, ...
Phrasal-Concept Query	paraphrases and textual entailment, generic paraphrase-based ap-proach, semantic approach, relation extraction, entailment relation

5.4.5 Further Analysis

In Table 5.5, we show the number of improved or degraded queries with respect to the best baseline (i.e., MLE), within the top 10 suggestions for the 183 queries in the ACL collection. From this table, we can study the robustness of the proposed approach. About 70.6% of the queries generated by our method are more effective than the baseline. Moreover, about 44.4% of the generated queries dramatically outperform the baseline (i.e., improvements are greater than or equal to 25%).

Table 5.5: Improved or Degraded Queries against the Best Baseline

Improved \Degraded	Query Count	Percentile Ratio
$(\infty, -25\%]$	139	7.6%
$(-25\%, 0.0\%)$	398	21.8%
0.0%	0	0.0%
$(0\%, +25\%]$	480	26.2%
$[+25\%, \infty)$	813	44.4%
Sum	1830	100%

5.5 User Experiments

In the user experiments, we conduct a questionnaire survey to identify preferences among a number of query suggestions. In other words, we ask users to select the most effective suggestion among many query examples generated by several methods. By doing this, we intend to identify which methods can generate more useful queries for users. We first describe the details of the survey, and then provide the results.

5.5.1 Survey Settings

In our survey, we assume a situation where users (assessors) need to construct a list of articles relevant to a given paper (the "query" papers in our previous experiments). Each assessor is asked to select the most effective queries from the list of queries for finding the relevant articles. For each query paper, we first provide its title and abstract as a summary of the paper. Then, we list 8 different query suggestions generated by 4 different methods (NGram, RM, MLE; see Chapter 3.5), and our method (PHRASAL-CONCEPT)) to an assessor. For each baseline, we generate two different queries by selecting the top 1 to 5 and top 6 to 10 terms (or n-grams) ranked by the method. We also use the top 1 and 2 phrasal-concept queries generated by our method. As a result, 8 queries are suggested, and two suggestions per method were provided. To prevent assessors from inferring methods by the order of suggestions, we randomly shuffle the suggestion order. We ask assessors to select one or two queries that they believe would be more useful to retrieve relevant articles among the 8 suggestions. By doing this, the methods that can generate more effective queries for users would be chosen.

Figure 5.3 shows an example of a question in the survey. To collect query papers, we selected 15 papers among the 183 query papers in our ACL collection (described in Chapter 3.3). For a fair comparison, the 15 papers were selected considering the results of retrieval experiments (reported in Table 5.5); first, we selected 5 papers for

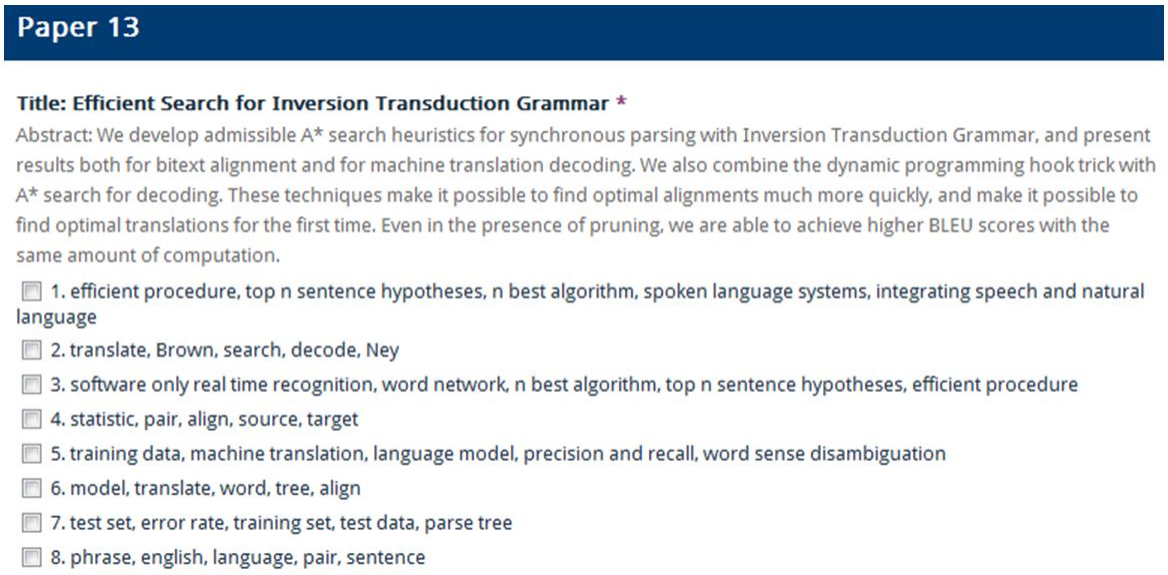

Figure 5.3: User Survey Example

which our proposed method worked significantly better than the baseline method in retrieval experiments (i.e., MLE); second, 5 papers were chosen for which the baseline method outperformed our method; finally, 5 papers were randomly selected among the papers for which our method performed as well as the baseline. This survey was done by the help of 20 volunteers who were graduate students majoring in computer science and familiar with the topics in computational linguistics (on which the ACL query papers focus).

5.5.2 Survey Results

In the survey, a total of 484 responses was collected, and for each question (query paper), a respondent selected 1.61 queries on average, out of 8 queries (we asked to select only one or two of the best queries). We first analyze the quality of queries generated by each method.

Table 5.6 shows the top one and two suggested queries by each method for two research papers. In the table, the number in parenthesis indicates the number of

responses which selected each method. First, it is clear that our phrasal-concept queries can present more plausible phrases than the baselines. For instance, "extracting structural paraphrases" refers to a task while "multiple sequence alignment" refers to a technique used in the field of paraphrase recognition (paper 1). Also, "extracting product features and opinions" and "learning subjective nouns" are important tasks in the study of opinion analysis (paper 2). Thus, these key concepts are related to many citations of each query paper. Second, the quality of NGram suggestions looks poor. Most of the suggested phrases are too general, and their meanings are vague since this method simply counts only correlations between the initial query and phrases without considering properties needed for queries in a specific domain. Another interesting point is that MLE tends to suggest the names of important authors who published frequently cited papers, e.g., "Regina Barzilay" (for paper 1) and "Theresa Wilson" (for paper 2). This is because MLE uses statistical learning to extract highly discriminative terms, e.g., author name.

Next, we provide the average number of responses that selected queries generated by each method per question, as shown in Table 5.7. In the survey, 20 assessors answered each question, and each assessor can choose one or two queries among 8 different suggested queries generated by 4 different methods. For example, 20.87% of RM means that for a question, 20.87% of all 20 assessors prefer the query suggestions generated by RM. The statistically significance is marked using the first letter of each method (the paired t-test is performed with $p < 0.001$).

First, users strongly prefer to use our phrasal-concept queries, i.e., PHRASAL-CONCEPT accounted for 62% of the all responses. Second, although NGram can suggest phrases to the user, NGram suggestions are significantly less preferred because of their poor quality. As discussed above, the concepts suggested by our method look more readable and effective to retrieve relevant documents, and thus the assessors in the survey show preference for phrasal-concepts. However, user preferences in the

Table 5.6: Examples of 8 Query Suggestions

Query Paper 1. Title: paraphrase recognition via dissimilarity significance classification.		
	Top 1 Suggestion	*Top 2 Suggestion*
RM (3)	paraphrase, sentence, word, pair, translate	phrase, match, align, extract, parallel
MLE (5)	barzilay, paraphrase, align, synonymy, pair	similar, regina, call, high, contiguous
NGram (0)	noun phrase, artificial intelligence, training data, test set, machine translation	machine learning, total number, statistical machine translation, human language technology
PHRASAL-CONCEPT (31)	extracting structural paraphrases, aligned monolingual corpora, paraphrase generation, large paraphrase corpora, multiple sequence alignment	unsupervised construction, sentential paraphrases, exploiting massively parallel news sources, monolingual machine translation, paraphrase identification and corpus construction

Query Paper 2. Title: feature subsumption for opinion analysis.		
	Top 1 Suggestion	*Top 2 Suggestion*
RM (7)	feature, word, sentence, set, opinion	polarity, classify, term, train, data
MLE (0)	feature, fix, Theresa, classify, classification	set, class, recall, Joachim, manual
NGram (0)	noun phrase, part of speech, training data, test set, machine learning	supervised learning, error rate, statistical learning, number of words, set of features
PHRASAL-CONCEPT (31)	extracting product features and opinions, review classification via human provided information, extraction pattern boot-strapping, learning extraction patterns, learning subjective nouns	phrase level sentiment analysis, con-textual polarity, opinionated sentences, review classification via human provided information, subjectivity analysis

Table 5.7: Average Number of Responses

Method	Response	Percentile Ratio
NGram	0.73	2.27%
MLE	4.93	15.29%
RM	6.74^{N}	20.87%
PHRASAL-CONCEPT	19.87^{RMN}	61.57%
Sum	32.27	100.0%

survey may not reflect the exact effectiveness of suggestions in retrieval. Nevertheless, these preference results reveal that phrasal-concepts are more preferred by academic search users. Accordingly, our method is more useful than the baseline methods from the user perspective.

5.6 Summary

In this chapter, we proposed a phrasal-concept based query generation technique, which is specifically designed for academic literature search. To generate more effective queries, we identify key concepts from pseudo-relevant documents by exploiting a label propagation technique and baseline query. By combining the key concept and its related concepts, a phrasal-concept query is generated. Through user studies and retrieval experiments, we show that users strongly prefer to use phrasal-concept queries generated by our method, and the phrasal-concept queries can improve retrieval performance in literature search environments.

CHAPTER 6

DIVERSE QUERY GENERATION

6.1 Overview

In this chapter, to improve domain-specific searches, we introduce the concept of diverse query generation based on query documents. While most previous work on query generation (e.g., [38, 107]) has focused on generating a single best query that can retrieve more relevant documents from a single retrieval result, little work has been done for generating diverse queries that can improve overall retrieval effectiveness in sessions (i.e., more relevant documents in aggregated retrieval results obtained by multiple queries in a session). In other words, emphasizing diverse query generation is important because query documents typically contain several different aspects (or topics) and many different types of relevant documents could be related to these aspects. We have already discussed this in Chapter 1.2, and now propose a diverse query generation framework. We first formulate the diverse query generation problem, and define associated terms. Then, we describe our framework to generate diverse queries. In addition, we propose a diverse suggestion method that suggest diverse and effective queries. To evaluate our framework, we conduct retrieval experiments on the patent and academic domains.

6.2 Problem Formulation

Definition 1. (Query Aspect): *Query aspect* denotes a topic in a query document. We assume that a query document includes multiple query aspects. Since a query document is generally very long (e.g., a patent contains about 3,000 terms on average

[48]) and can include complex structures, multiple aspects can be identified in a query document. To represent a query aspect, we use the set of related terms from the query document.

Definition 2. (Keyword Query): A *keyword query* is a set of terms, e.g., $Q = \{\text{stereo}, \text{digital}, \text{sound}, \text{amplifier}\}$. We simply create keyword queries for diverse query generation. In addition, term weighting is not considered here because it is less useful for suggestion. Since very long queries are also not useful as suggestions, we empirically restrict query length to 5 terms.

Problem 1. (Query Aspect Identification): We assume n different query aspects in a query document, and *query aspect identification* is generating n distinct sets of terms, each of which contains relevant terms to represent a query aspect.

Problem 2. (Diverse Query Generation): *Diverse query generation* is generating diverse queries based on query aspects (identified in Problem 1). Given n query aspects, we generate n sets of queries, each of which related to a query aspect.

Problem 3. (Diverse Query Suggestion): *Diverse query suggestion* is suggesting k diverse and effective queries from the generated queries (Problem 2).In this, diversification is based on the query aspects (recognized in Problem 1), and suggested queries should be effective for retrieving relevant documents as well as being related to diverse aspects. By doing this, the overall search results obtained by the suggestions would contain more relevant documents related to the diverse aspects.

6.3 Diverse Query Generation Framework

We now describe our framework for generating diverse queries, as shown in Figure 6.1. In this framework, we adopt a two-step process: (Step-1) identifying n different query aspects and (Step-2) generating multiple queries related to n query aspects. Query aspect identification (Step-1) is required for generating diverse queries in Step-2. We generate queries based on the query representation that users explicitly specify,

and exploit pseudo-relevant documents to generate effective queries that can retrieve more relevant documents. We provide the details of each step as follows.

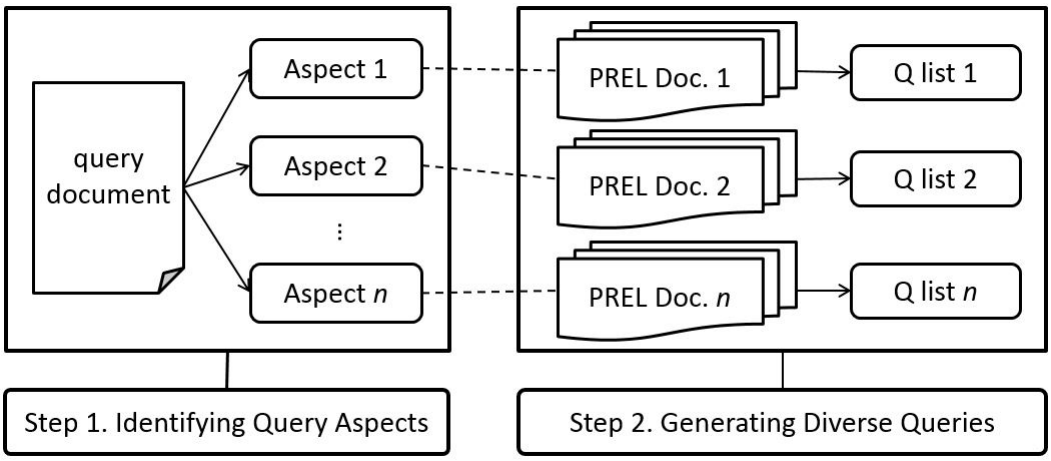

Figure 6.1: Diverse Query Generation Framework

6.3.1 Query Aspect Identification

The first step is identifying n query aspects. By defining a query aspect as a set of related terms from the query document (*Def. 1*), we transform this task into a term clustering problem, i.e., given terms extracted from a query document we form n different clusters, each of which contains a subset of the terms. Specifically, for a query document, we extract m distinct terms by their tf.idf weights (stop-words are not extracted), and generate $m \times (m-1)/2$ term pairs (the similarity is undirected). By estimating the similarity for each term pair $< t_i, t_j >$, we can generate a m-by-m symmetric similarity matrix whose diagonal value (similarity of $< t_i, t_i >$) is 1. Then, we apply a term clustering algorithm using this matrix to generate n different term sets. In our experiments, we extract 500 terms from each query document, and use the spectral clustering algorithm implemented by [22], but any other clustering methods can be applied. Next, we describe our method to estimate the similarity of $< t_i, t_j >$.

We define similarity between terms by a mixture of topical relatedness (or association) and retrieval effectiveness when terms are clustered together. In other words, we make clustering algorithms group the terms if they are topically associated and are also effective for retrieving relevant documents. To achieve this, we introduce the following similarity function.

$$\text{Sim}(t_i, t_j) = (1 - \lambda) \cdot T(t_i, t_j) + \lambda \cdot R(t_i, t_j) \tag{6.1}$$

where t_i and t_j is a term pair from a query document.

In Eq. (6.1), $T(t_i, t_j)$ measures topical relatedness between t_i and t_j, while $R(t_i, t_j)$ estimates retrieval effectiveness. λ is a controlling parameter. For measuring topical association (T), we utilize term statistics obtained from the document corpus (e.g., PMI). To estimate R, we leverage the features from query performance predictors (e.g., query clarity [27], query scope [42], etc.). Table 6.1 lists the features for implementing topical association and retrieval effectiveness.

Table 6.1: Features for Similarity Learning

Category	Features
Topical Relatedness	PMI of $< t_i, t_j >$ calculated by 8-word windows recognized in all documents in a corpus PMI of $< t_i, t_j >$ measured by titles PMI of $< t_i, t_j >$ calculated by 8-word windows identified in query document
Retrieval Effectiveness	Query Clarity (QC) [27] Query Scope (QS) [42] Similarity Collection / Query (SCQ) [113] Inverse Document Frequency (IDF) Inverse Collection Term Frequency (ICTF)

Using these similarity learning features, we can rewrite the Eq. (6.1) as follows.

$$\text{Sim}(t_i, t_j) = \sum_k w_k \cdot f_k(t_i, t_j) \tag{6.2}$$

where f_k indicates a feature defined in Table 6.1 and $_k$ is a weight of the k-th feature.

To predict more accurate similarity, we employ a supervised learning approach. Given a term pair $< t_i, t_j >$, a supervised learner estimates its similarity score by learning an optimal value of the feature weights ($w = w_1, \ldots, w_k$). To do this, we generate training examples (i.e., labeled term pairs) as follows.

For each query document, N different term pairs are extracted, and we label each pair (i.e., example) as positive or negative, i.e., $L(< t_i, t_j >) \in 0, 1$. Since we represent similarity by topical association and retrieval effectiveness, a term pair is positive if its terms are highly associated and effective for retrieving relevant documents; otherwise, the term pair is negative. To determine this, we use the following conditions, and an example is positive if it satisfies every condition; otherwise the example is negative.

1. Two terms involve high "retrieval effectiveness" if they have a high generation probability based on the language model estimated for any relevant document.

2. Two terms are highly "associated" if their PMI estimated from any relevant document is greater than a threshold.

For the first condition, we calculate the probability of a term t for a relevant document as follows.

$$P(t|RD) = \frac{tf(t, RD) + \mu \cdot P_c(t)}{|RD| + \mu} \tag{6.3}$$

where $tf(t, RD)$ is the frequency of t in a relevant document RD, $P_c(t)$ is a corpus probability, and μ is a Dirichlet smoothing parameter.

For each RD, we assume that that the top 100 terms ranked by Eq. (6.3) satisfy the criteria for effectiveness. For the second constraint, we assume that PMI estimated from RD indicates topical associations that are effective for retrieving relevant documents. To calculate PMI of $< t_i, t_j >$, we use 8-word windows, and t_i and t_j are highly associated if PMI$(t_i, t_j : RD) > \theta$ where θ is a cut-off value. Since we exploit real "relevant" documents for labeling training examples, we use l-fold

cross-validation; $(l-1)/l$ query documents and their relevant documents are used for training, and the other query documents whose relevant documents are hidden are used for testing.

6.3.2 Diverse Query Generation

As a result of the first step, we obtain n distinct sets of terms, each of which represent one query aspect in a query document. Based on this, we generate queries by exploiting the query generation method described in Chapter 4.3. For each query aspect (i.e., a set of terms), we first retrieve pseudo-relevant documents (PRD) obtained by the terms in the aspect; we use those terms as a query and assume that the top k retrieved documents are pseudo-relevant. In addition, we generate an equal number of non-relevant documents (NRD) by randomly selecting another k documents from those ranked below the top k. Then, we train binary decision trees using PRD and NRD where the terms in PRD are used as attributes (see Alg. 1). Once a decision tree is learned, we generate a query by extracting attributes (terms) on a single path from the root to a positive leaf node (i.e., pseudo-relevance). We define a query as a list of keywords (e.g., $Q = \{$"battery", "charger", "cellular", "phone"$\}$), and ignore the attributes associated with negation.

6.4 Diverse Query Suggestion

We define *diversifying query suggestions* as suggesting k queries that will be effective for finding relevant and novel documents for a query document. To do this, we exploit the xQuAD diversification model proposed in [92] and introduce the following probabilistic query suggestion framework. Note that the proportionality-based diversification (which can perform slightly better than the xQuAD approach) is proposed after we develop the diverse query generation method, and the xQuad framework was the state-of-the-art method when we research on diversifying query suggestions. In

the xQuAD approach, among all generated queries, we select the queries that are more relevant to the query document and novel relative to the current suggestion list. Alg. 3 describes this framework.

Algorithm 3 Diverse Query Suggestion (DivQS)

Input:
L is a list of generated queries
D_Q is a query document
k is the number of queries to be suggested
Output:
S is the ranked list of query suggestions

1: $S \leftarrow \emptyset$
2: **while** $|S| < k$ **do**
3: $\quad q^* \leftarrow \arg\max_{q \in L \setminus S}(1-\lambda) \cdot P(q|D_Q) + \lambda P(q, \overline{S}|D_Q)$
4: $\quad L \leftarrow L \setminus q^*$
5: $\quad S \leftarrow S \cup q^*$
6: **end while**
7: **return** S

Given a query document D_Q and a list of generated queries L, we iteratively choose the most probable query obtained by:

$$(1-\lambda) \cdot P(q|D_Q) + \lambda \cdot P(q, \overline{S}|D_Q) \tag{6.4}$$

where S is the list of suggested queries and q is a candidate query from L.
In Eq. (6.4), $P(q|D_Q)$ denotes the relevance of q to D_Q, while $P(q, \overline{S}|D_Q)$ indicates the novelty of q to S. That is, these two probabilities are optimizing relevance and diversity, controlled by λ. $P(q|D_Q)$ can be computed by $\prod_{t \in q} P_{LM}(t|D_Q)$, i.e., the unigram language model estimated from D_Q, and $P(q, \overline{S}|D_Q)$ can be estimated using the identified query aspects.

Using the set of query aspects A_Q we can marginalize $P(q, \overline{S}|D_Q)$ as:

$$P(q, \overline{S}|D_Q) = \sum_{ap \in A_Q} P(ap|D_Q) \cdot P(q, \overline{S}|ap) \tag{6.5}$$

where ap is a query aspect in A_Q.

In Eq. (6.5), we consider $P(ap|D_Q)$ as the importance of an aspect ap for D_Q, which is estimated by $\prod_{t \in ap} P_{LM}(t|D_Q)$.

By assuming that the current candidate query q is independent of the queries already selected in S, $P(q, \overline{S}|ap)$ can be derived as:

$$P(q, \overline{S}|ap) = P(q|ap) \cdot P(\overline{S}|ap) \tag{6.6}$$

$P(q|ap)$ measures the *coverage* of q with respect to ap, and $P(\overline{S}|ap)$ provides a measure of *novelty* to the current suggestion list S for a given ap.

To estimate these probabilities, we use retrieval results obtained by q, S, and ap. Specifically, we assume that a query's top 100 retrieved documents can represent underlying topics of the query, and $P(q|ap)$ can be estimated by how much the topics in ap are covered by q. The equation is given as:

$$P(q|ap) \cong \frac{|Ret_q \cap Ret_{ap}|}{|Ret_{ap}|} \tag{6.7}$$

where Ret_{ap} is the set of the top 100 documents retrieved by ap. Note that we use the terms in a query aspect as a query.

For the estimation of $P(\overline{S}|ap)$, we further assume that the queries chosen as suggestions in S are independent for ap, and the following estimation can be given.

$$P(\overline{S}|ap) \cong P(\overline{qs_1, qs_2, \ldots, qs_{n-1}}|ap) \cong \prod_{qs \in S} (1 - P(qs|ap)) \tag{6.8}$$

where qs is a query in S and $P(qs|ap) \cong |R_{qs} \cup R_{ap}|/|R_{ap}|$.

As a result, Eq. (6.4) can be rewritten as:

$$(1 - \lambda) \cdot P(q|D_Q) + \lambda \cdot \sum_{ap \in A_Q} \left[P(ap|D_Q) \cdot P(q|ap) \cdot \prod_{qs \in S} (1 - P(qs|ap)) \right] \tag{6.9}$$

We use Eq. (6.9) in the diversification algorithm (Alg. 3), and for each query document, k queries are selected as suggestions.

6.5 Evaluation

6.5.1 Experimental Setup

For evaluation, we conduct experiments on the patent and academic domains (see Chapter 3.2), and the corresponding test collections, i.e., USPAT and ACL (described in Chapter 3.3), are used. Queries and documents are stemmed by the Krovetz stemmer [60]. To identify query aspects and generate diverse suggestions, we perform 5-fold cross-validation with random partitioning. For learning similarity, we use Logistic Regression [4]. To run each suggested query, we use the query likelihood model [84] implemented by Indri [97]. In addition, as discussed in Chapter 3.4, we evaluate with the top 100 documents ranked by each query suggestion. To measure retrieval effectiveness and diversity, we use session-based metrics, i.e., Normalized Session Discounted Cumulative Gain (NSDCG) and Session Novelty Recall (SNR) (see Chapter 3.7 for more detail) because multiple queries are suggested for each query document.

In the experiments, we first evaluate query aspect identification (described in Chapter 6.3.1) which is important for generating diverse queries, and then verify the effectiveness of diverse queries generated for each query document. We empirically set the number of generated aspects as 10 or 20, and this setting could provide some significantly improvements in the experiments. However, further explorations to find somewhat optimal values can be necessary. For evaluating query aspect identification, we generate an initial baseline query (BL0) by using ReductionQuery (described in Chapter 3.5). To evaluate diverse suggestion results, we employ two different baselines. The first baseline (BL1) is NGram that can suggest multiple n-grams more correlated with the query document. The other baseline (BL2) is the decision tree-based query generation method proposed in Chapter 4. We generate keyword queries

by ignoring the terms associated with negation. The difference between BL2 and our diverse query generation (DivQ) is that DivQ identifies multiple query aspects (from a query document) to generate diverse queries.

6.5.2 Retrieval Results

(**Query Aspect Identification Performance**) In this experiment, we hypothesize that more relevant documents are retrieved if an identified query aspect is effective. We measure the retrieval effectiveness of each query aspect by formulating a query using the terms in each query aspect. Table 6.2 shows the retrieval results of query aspects and the baseline (BL0). For each query document, 10 query aspects are identified and a single baseline query is used. We measure recall (R@100) in two different ways: (1) selecting the best one among n different query aspects (Max R@100) and (2) aggregating the retrieved relevant documents (within rank 100) by all query aspects (Agg. R@100). We report an average value of each metric over the query documents in each corpus. BL0 and QA indicates ReductionQuery and the identified query aspects (see Chapter 6.3.1), respectively. In addition, a * indicates a significant improvement over the baseline (BL0), and the paired t-test is performed with $p < 0.05$. First, regarding Max. R@100, our method can generate at least one query

Table 6.2: Query Aspect Evaluation

Metric	**PAT(patent)**		**ACL(academic)**	
	BL0	QA	BL0	QA
R100	0.1091	–	0.4452	–
Max. R100	–	0.1491*	–	0.4695*
Agg. R100	–	0.1918*	–	0.6369*

aspect which can significantly outperform the baseline. Second, from Agg. R@100 we see that significantly more relevant documents are retrieved when using all identified aspects. This is a useful result because query aspects can find relevant documents

that are missed by BL0 and the query suggestions generated by these aspects should also perform well.

Table 6.3: Session Evaluation

PAT(patent)					
Metric	♯Q	BL1	BL2	DivQ ($n = 10$)	DivQ ($n = 20$)
SNR	5	0.1560	0.1715^{1}	0.1855^{1}	$\mathbf{0.1961}^{12}$
@100	6	0.1625	0.1813^{1}	0.1982^{1}	$\mathbf{0.2099}^{12}$
	7	0.1688	0.1870^{1}	0.2086^{12}	$\mathbf{0.2223}^{12}$
	8	0.1759	0.1913^{1}	0.2158^{12}	$\mathbf{0.2340}^{12}$
	9	0.1809	0.1953^{1}	0.2248^{12}	$\mathbf{0.2442}^{12}$
	10	0.1893	0.1989	0.2322^{12}	$\mathbf{0.2509}^{12}$
NSDCG	5	0.0812	0.0827	0.1209^{12}	$\mathbf{0.1319}^{12}$
@100	6	0.0808	0.0851	0.1188^{12}	$\mathbf{0.1296}^{12}$
	7	0.0799	0.0876	0.1172^{12}	$\mathbf{0.1274}^{12}$
	8	0.0791	0.0906	0.1151^{12}	$\mathbf{0.1251}^{12}$
	9	0.0785	0.0932	0.1137^{12}	$\mathbf{0.1235}^{12}$
	10	0.0783	0.0959	0.1127^{12}	$\mathbf{0.1212}^{12}$
ACL(academic)					
SNR	5	0.5459	0.5731^{1}	0.6329^{12}	$\mathbf{0.6519}^{12}$
@100	6	0.5561	0.5928^{1}	0.6581^{12}	$\mathbf{0.6811}^{12}$
	7	0.5770	0.6095^{1}	0.6775^{12}	$\mathbf{0.6981}^{12}$
	8	0.5893	0.6174^{1}	0.6961^{12}	$\mathbf{0.7122}^{12}$
	9	0.6011	0.6260^{1}	0.7106^{12}	$\mathbf{0.7260}^{12}$
	10	0.6078	0.6351^{1}	0.7192^{12}	$\mathbf{0.7392}^{12}$
NSDCG	5	0.3273	0.3116	0.4200^{12}	$\mathbf{0.4347}^{12}$
@100	6	0.3304	0.3121	0.4264^{12}	$\mathbf{0.4402}^{12}$
	7	0.3338^{2}	0.3120	0.4312^{12}	$\mathbf{0.4438}^{12}$
	8	0.3362^{2}	0.3119	0.4345^{12}	$\mathbf{0.4461}^{12}$
	9	0.3379^{2}	0.3110	0.4363^{12}	$\mathbf{0.4468}^{12}$
	10	0.3385^{2}	0.3099	0.4357^{12}	$\mathbf{0.4457}^{12}$

(**Diverse Query Suggestion Performance**) Next, we evaluate the diverse query generation method in terms of retrieval effectiveness and diversity. For each query document, we use the diverse query suggestion method (i.e., Alg. 3 described in Chapter 6.4) to suggest 5 to 10 queries, and 10 or 20 different query aspects are used for generating queries (i.e., $n = 10$ or 20). The baselines (BL1&2) generate the same number of query suggestions for the same query document. Table 6.3 reports

retrieval performance of each method using the patent and academic collections. In each row, the best result is marked by bold, and a significant improvement over each baseline is denoted by the number of the baseline, e.g., 1 denotes an improvement over "BL1" (the paired t-test is performed with $p < 0.05$). First, in both domains, BL2 can outperform BL1 in terms of SNR. Second, diverse queries (DivQ) can generate significantly more diversified results and retrieve more relevant documents. SNR verifies that DivQ is more effective at finding new relevant documents missed by previous queries (since SNR ignores the relevant documents retrieved by any previous queries). Third, considering NSDCG, DivQ is significantly better at placing relevant documents at higher ranks. This is because the queries generated by DivQ contain more discriminative terms from relevant documents.

6.5.3 Further Analysis

We now provide some additional analysis of our diverse query generation. The main reason for generating diverse queries is so that more relevant documents can be retrieved, which cover "diverse" aspects of a query document (as shown in Table 6.3). In accordance with this, by examining more query suggestions, we can find new relevant documents which are not covered by previously suggested queries. To highlight this, we measure the average number of new relevant documents retrieved by the k-th query in the top 10 suggestions. Figure 6.2 shows this for the patent domain.

In Figure 6.2, our diverse queries (DivQ) can retrieve more new relevant documents than the baseline queries (i.e., BL1&2), and as a result of examining the top 10 suggestions, more relevant documents are retrieved by the diverse queries.

In addition, we evaluate the quality of our suggested queries. For this, we employ five users (assessors) and ask them to determine whether each suggestion looks useful for retrieving relevant documents. We randomly selected 10 query papers from the

ACL test collection (described in Chapter 3.3), and for each query paper, we provide its title, abstract, and 5 queries generated by our method. Table 5.6 shows an example of query suggestions and query paper. For each query, we assume that the query is "useful" if three or more assessors tag it as useful. Out of 50 queries, we found that 37 queries (i.e., 74%) are useful. For example, in Table 5.6, the query {"WCDG", "information", "dependency", "grammar"} is particularly useful because this query paper proposes an hybrid parser based on Weighted Constraint Dependency Grammar (i.e., WCDG) and many cited papers are related to this query. In addition, research about probabilistic parsing models are also cited in this query paper, and the query {"Collins", "statistical", "parse"} is quite useful for retrieving this work.

6.6 Summary

In this chapter, we proposed a framework for generating diverse queries, which can help to retrieve more relevant documents. We identify diverse query aspects, generate queries related to these aspects, and then suggest a diverse ranked list of these queries. Through experiments, we showed that the suggestions generated by our approach produce more diverse and effective search results in comparison to baseline methods. Our method is easily reproducible and general; we do not require any

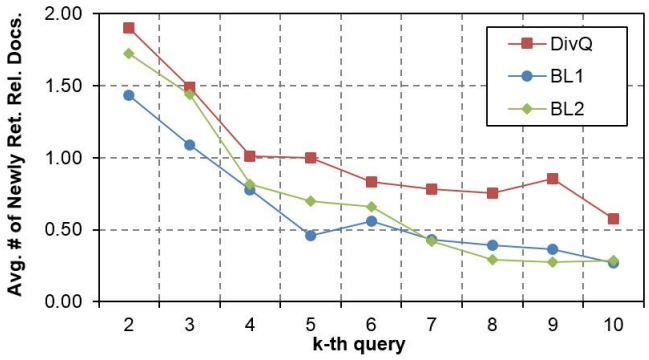

Figure 6.2: Average Number of New Relevant Documents for Patent Domain

Table 6.4: Diverse Query Suggestion Example

Query Paper
Title: Hybrid Parsing – Using Probabilistic Models as Predictors for a Symbolic Parser Abstract: In this paper, we investigate the benefit of stochastic predictor components for the parsing quality which can be obtained with a rule-based dependency grammar. By including a chunker, a supertagger, a PP attacher, and a fast probabilistic parser we were able to improve upon the baseline by 3.2%, bringing the overall labelled accuracy to 91.1% on the German NEGRA corpus. We attribute the successful integration to the ability of the underlying grammar model to combine uncertain evidence in a soft manner, thus avoiding the problem of error propagation.

No.	**Suggested Query**
1	{probabilistic, model, treebank, dependency, predictor}
2	{Collins, statistical, parse}
3	{WCDG, information, dependency, grammr}
4	{best, probabilistic, parse, tree}
5	{Charniak, grammar, treebank, parse}

manually constructed data or external resources, and effectiveness was verified in two different domains.

CHAPTER 7

SEARCH RESULT DIVERSIFICATION

7.1 Overview

Domain-specific search has some unique features relative to web search. As discussed in Chapter 1, one major characteristic in domain-specific search is that search queries are formulated from a query document (e.g., a new patent in prior-art search). Typically these queries are complex and can cover diverse aspects of the query document in order to retrieve relevant documents that cover the full scope of the query document. Given this context, search diversification techniques can potentially improve the retrieval performance of domain-specific search by introducing diversity into the document ranking. In Chapter 6, we proposed a query-side diversification approach (i.e., diverse query generation) to generate diversified search results. However, in this chapter, we examine the effectiveness of a result-level diversification framework that can re-order an initial retrieval result so that the final ranked list can include more diverse aspects (or topics) associated with the query document. Note that "query aspect" is referred to as "query topic" in this chapter. In this diversification process, query topics are first identified, and then re-ranking algorithms (e.g., [92, 29]) are applied with the identified topics. Using this framework involves developing methods to identify effective phrases related to the topics mentioned in the query document. In experiments, we evaluate the result-level diversification approach using standard measures of retrieval effectiveness and diversity.

7.2 Problem Formulation

Diversifying domain-specific search results is designed to improve the retrieval effectiveness of initial ranked results. As discussed in Chapter 1.5, we assume that diverse topics are involved in a query document, and that diversification of initial search results based on those topics will improve retrieval performance.

Given a query document QD, let $T = t_1, t_2, \ldots, t_n$ be a topic set for QD and $W = w_1, w_2, \ldots, w_n$ be a weight set where some weight w_i is assigned to each topic t_i. Note that this weight is used as the importance [92] or popularity [29] by the diversification algorithm applied. In addition, an initial document list for QD is given, i.e., $D = d_1, d_2, \ldots, d_m$, and each d_i's relevance to t_i can be estimated, i.e., $P(d_i|t_i)$. Using $< T, W, P(d_i|t_i) >$, typical diversification algorithms (e.g., [29, 92]) generate a subset of D which forms a diverse rank result S where $|S| = K$. However, recent work [28] found that explicitly specified topic structures (e.g., grouping topic terms to form a topic such as "user interface", "sharing") are less beneficial for improving search performance. Instead, identifying topic terms (e.g., "user", "interface", "sharing") and directly using such terms without the more complex step of topic identification can be effective. Following this, we formulate a phrase-level diversification method for domain-specific search. We assume that phrases are more useful than words (i.e., unigrams) to express domain-specific topics. For example, patent documents frequently contain longer technical terms (e.g., "portable duplex radio telephone") and academic papers also include many phrasal concepts (e.g., "lexical acquisition"). Thus, we identify a set of topic phrases for T, and apply diversification frameworks (e.g., [28]) using these phrases.

The formal definition of phrase-level diversification is given as follows. Let us assume that a topic $t \in T$ can be represented by an arbitrary set of phrases, i.e., $t = \{p_1, p_2, \ldots, p_{n(t)}\}$ where p_i is a topic phrase for QD and $n(t)$ is the number of phrases to form t. Then, T can be rephrased as:

$$T^{'} = \Big\{ \{p_1^{t_1}, p_2^{t_1}, \ldots, p_{n(t_1)}^{t_1}\}, \ldots, \{p_1^{t_n}, p_2^{t_n}, \ldots, p_{n(t_n)}^{t_n}\} \Big\} \tag{7.1}$$

We hypothesize that there is a set of phrases that can contain all phrases in $T^{'}$, i.e., $P^{'} = \{p | \forall p \in T^{'}\}$, and the phrase-level diversification is defined as generating a diverse ranked list $S \subset D$ using $P^{'}$. In effect, each phrase is treated as a topic in the diversification model (see Chapter 7.3). As a result of diversification, S covers more topic phrases and contains more diverse relevant documents.

7.3 Diversification Framework

Explicit diversification methods (e.g., PM-2 [29] and xQuAD [92]) assume that some set of query topics (or aspects) is specified, and generate diverse ranked results based on these topics. Among many algorithms, we select to use the proportionality-based approach (PM-2) for our diversification task, which is the most recently proposed state-of-the-art technique. This approach exploits the Sainte-Laguë method, allocating seats in proportional representation, for assigning the portions of topics in S such that the number of each topic's documents in S is proportional to the weight of the topic, i.e., w_i. Specifically, PM-2 requires a set of topics T, an initial document retrieval list D, and an empty list S. In each iteration, the quotient qt_i of each topic t_i is computed as:

$$qt_i = \frac{w_i}{2s_i + 1} \tag{7.2}$$

where s_i is the current portion of t_i in S.

Using this, PM-2 selects the most proportional topic t_i^* with the largest qt_i, and places the document $d^* \in D$ into S such that d^* is mostly relevant to t_i^* as well as other topics:

$$d^* \leftarrow \arg\max_{d \in D} \lambda \cdot qt_i^* \cdot P(d|t_i^*) + (1 - \lambda) \sum_{i \neq i^*} qt_i \cdot P(d|t_i) \tag{7.3}$$

where $P(d|t_i)$ is an estimated relevance of d to t_i.

Although Eq. (7.3) is effective for diversifying web search results, there are limitations when using it for domain-specific search where a search process starts from a query document. In PM-2, Eq. (7.3) only considers the relevance of a document to each topic, not directly to the whole query document. This setting could work for web search results because the diversification aims to clarify ambiguous web queries. On the other hand, many domain-specific search tasks are recall-oriented, i.e., not missing relevant documents in a relatively long retrieval result is more important than placing them at top ranks. So, keeping the documents "relevant" to QD (by some estimation) in S is important. To do this, we combine Eq. (7.3) with the relevance score of d for QD.

$$d^* \leftarrow \arg\max_{d \in D} \mu \cdot \text{relevance}(d) + (1 - \mu) \cdot \text{diversity}(d) \tag{7.4}$$

where relevance(d) is an estimated relevance score of d for QD and diversity(d) is the diversity score calculated by Eq. (7.3).

Using Eq. (7.4), we can choose the document not only related to the appropriate topic but also highly relevant to the query patent. In experiments, we use the retrieval score obtained by the baseline retrieval model as the estimation of relevance(d). After selecting d^*, the algorithm updates the portion of each topic in S (i.e., s_i) by its normalized relevance to d^*:

$$s_i \leftarrow s_i + \frac{P(d^*|t_i)}{\sum_j P(d^*|t_j)} \tag{7.5}$$

Then, this process is repeated with the updated s_i, and stops after S contains K documents. The final ranking of a document is determined by the order in which the document is included in S. As described in Chapter 7.2, we use phrase-level diversification for domain-specific search, and thus the set of topic phrases (interpreted

as topics) is the input to this diversification model. In the next section, we present our method to generate topic phrases, which is important for diversification performance.

7.4 Automatic Topic Phrase Identification

The goal of identifying topic phrases is generating a list of effective phrases for diversification. As discussed in Chapter 7.2, we need to generate $P^{'}$ which contains all possible phrases to represent query topics. This is an important task because the diversification model (described in Chapter 7.3) assigns the documents in S primarily based on the input phrases. To identify effective topic phrases, we assume that query patents include sufficient phrases for query topics. In Chapter 5, we extract phrasal concepts from pseudo-relevant documents (initially ranked by the query document). However, in the diversification framework, we use the phrases appeared in the query document, and they are directly used to represent query topics (i.e., the topics residing in the query document).

Given a query document QD, we extract a set of noun phrases, $P = p_1, p_2, \ldots$ syntactically recognized in QD, and assume that some subset of P can be the effective set of topic phrases, i.e., $P^{'}$. Note that we use OpenNLP[1] to recognize syntactic phrases.

7.4.1 Greedy Approximation for Dominating Set Problem

To obtain $P^{'}$, previous work [28] has used DSPApprox, the multi-document summarization technique proposed in [65, 66]. In this approach, it is assumed that an effective topic term (i.e., phrase) is useful to predict other terms, and its conditional probability is used for measuring how well the term predicts others [65]. In general, this approach can find topic terms by identifying a set of terms that are highly probable to predict many other terms in a vocabulary. In fact, finding such a set can be

[1]NLP library (http://opennlp.apache.org)

viewed as a Dominating Set Problem (DSP). Since the generalized DSP is NP-hard [39, 65], DSPApprox is a greedy approximation to solve DSP.

In [65, 66] DSPApprox is originally used for generating a set of terms to summarize target documents. However, it is also useful to find a diverse set of topic terms (i.e., phrases) for the query document. As done in [28], we can also use DSPApprox to select a set of topic phrases as follows.

Given an initial ranking result (i.e., D) and a query document (i.e., QD), the algorithm first extracts candidate phrases from QD (i.e., P) and generates a set of vocabulary from D, which includes many topic terms. Then, it measures the "topicality" of each phrase (e.g., relevance to QD) and its "predictiveness", i.e., how well the phrase can predict the appearances of other vocabulary terms, which can be represented by $P(p|t)$ where p is a candidate phrase and t is a vocabulary term. After this, the algorithm greedily selects a subset of P (i.e., P') by maximizing both topicality and predictiveness of P'. Next, we provide how to estimate "topicality" and "predictiveness" of each phrase.

Topicality measures how informative a phrase is to describe QD, and to compute this, we generate a relevance model [64] for QD, $P_R(t|QD)$.

$$P_R(t|QD) = \sum_{d \in R} P(t|d)P(d|QD) \tag{7.6}$$

where R is a set of (pseudo) relevant documents for QD.

Then, a topicality of a phrase p is calculated as:

$$\text{Topic}(p) = P_R(p|QD) \log_2 \frac{P_R(p|QD)}{P_c(p)} \tag{7.7}$$

where $P_c(p)$ is a collection probability of p.

This actually is the same as the contribution of p's clarity score for QD [28].

Predictiveness measures each phrase's ability to predict the occurrences of other terms in the initial retrieval result. The calculation of this is given as:

$$\text{Predict}(p) = \frac{1}{Z} \sum_{v \in C_p} P_{\text{w}}(p|v) \tag{7.8}$$

where $P_w(p|v)$ is the probability of a phrase p co-occurring with a term v in a window of size w, C_p is a term set, each of which co-occurs with p by the window, and Z is the normalizing factor.

As done in [28], Z is typically set as the size of the vocabulary. Using these estimations, we can perform DSPApprox for identifying topic phrases, and the details of this approach are described in [65, 28]. Since DSPApprox is a simple greedy algorithm only considering topicality and predictiveness, to improve the identification process, in the next section, we propose a learning-to-rank framework that combines these two features with other features.

7.4.2 Learning-to-rank Topic Identification

7.4.2.1 Ranking Model

In order to obtain an effective set of topic phrases, we rank the candidate phrases extracted from the query document, i.e., P, and use the top k phrases as topic phrases. For this, our ranking model produces a ranked list of the phrases in descending order of their (predicted) effectiveness to derive more query topics. This is formally defined as follows.

Given a query document QD, let $T = t_1, t_2, \ldots, t_n$ be a set of relevant topics, and $P = p_1, p_2, \ldots, p_l$ be a set of candidate phrases extracted from QD, where l denotes the number of extracted phrases. Suppose that $Y = y_1, y_2, \ldots, y_l$ is a set of ranks, and the order of the ranks is given as: $y_1 \succ y_2 \succ y_l$ where $\succ$ indicates the preference between two ranks. For each phrase $p_j \in P$, some corresponding rank, $y_(p_j)$, is assigned. To learn a ranking function, we generate training examples as follows.

Let $R = r_1, r_2, \ldots$ be the set of relevant documents for QD, and we need to map each relevant document r_i to a topic $t_j \in T$. To do this, we can exploit manually labeled topic information for each relevant document. For example, in the patent domain, there are IPC[2] codes which are annotated to any patent documents and can classify a patent document into predefined topic classes. We first extract all IPC codes from the relevant documents, and assume that each IPC code can form a query topic in T. Then, we map the relevant documents to the corresponding topics thru their annotated IPC codes. We can rewrite T as:

$$T_R = \left\{ \{r_1^{t_1}, r_2^{t_1}, \ldots, r_{nr(t_1)}^{t_1}\}, \ldots, \{r_1^{t_n}, r_2^{t_n}, \ldots, r_{nr(t_n)}^{t_n}\} \right\} \tag{7.9}$$

where $nr(t)$ is the number of relevant documents assigned to t.

Using T_R, we can create rank labels of training examples, i.e., the ranked list of candidate phrases. For each phrase p, we calculate its clarity score [27] which indicates the effectiveness of p can derive the relevant documents of each topic. The calculation is given as:

$$\text{scr}(p) = \sum_{t_i} \sum_{r \in t_i} P(p|r) \cdot \log_2 \frac{P(p|r)}{P_c(p)} \tag{7.10}$$

where $P_c(p)$ is the collection probability.

In Eq. (7.10), we use the unigram language model to estimate each phrase's probability, and the candidate phrases highly generative for more query topics are ranked higher. By using this, we can generate the training ranked list as:

$$\hat{Y} = \{y_{p_1}, y_{p_2}, \ldots, y_{p_l}\} \text{ such that } y_{p_i} \succ y_{p_j} \text{ if } \text{scr}(p_i) > \text{scr}(p_j) \tag{7.11}$$

[2]International Patent Classification (http://www.wipo.int/classifications/ipc/en/)

For training, a set of query documents, $\mathbb{QD} = \{QD_1, QD_2, \ldots, QD_{|\mathbb{QD}|}\}$, are given, and a feature vector $x_{ij} = f(QD_i, p_{ij}) \in X_i$ is generated for the pair of a query document and its candidate phrase. Then, we use $< X_i, \hat{Y}_i >$ for learning a ranking function.

7.4.2.2 Ranking Features

To compose a feature vector in our ranking model (described in Chapter 7.4.2.1, we use four types of features: (1) relevance, (2) importance, (3) predictiveness, and (4) cohesiveness. Table 7.1 summarizes these four types of features, and we describe each type as follows.

Table 7.1: Four Types of Ranking Features

Type	**Description**
Relevance	Query Relevance (Rel_{QD}), Pseudo Relevance (Rel_{PR})
Importance	Query Clarity (i.e., Topicality) [27], Query Scope [42], Inverse Collection Term Frequency, Inverse Document Frequency, Word Count
Predictiveness [65]	Query Document-based (Predict_{QD}), Pseudo Relevance-based (Predict_{PR})
Cohesiveness	Query Document-based (Cohv_{QD}), Pseudo Relevance-based (Cohv_{PR})

Relevance: Relevance contains two features measuring some probabilistic relevance to the query document or pseudo-relevant documents. We consider as pseudo-relevant the top N documents ranked in the initial retrieval result, i.e., D. Given a phrase $p = \{w_1, w_2, \ldots, w_{|p|}\}$, its query relevance is calculated as:

$$\text{Rel}_{QD}(p) = \prod_{w \in p} P(w|QD) \tag{7.12}$$

where w is a unigram word in p, and $P(w|QD)$ is the probability by the smoothed language model [110] drived from QD.

Pseudo-relevance exploits the relevance model [64] estimated by the pseudo-relevant documents, and the calculation is given as:

$$\text{Rel}_{PR}(p) = \prod_{w \in p} P_R(w|QD) \tag{7.13}$$

where $P_R(w|QD) = \sum_{d \in PR} P(w|d) \cdot P(d|QD)$ and PR is the set of pseudo-relevant documents.

We use query relevance and pseudo-relevance as relevance features. Since these features use the query model derived from the entire text of the query document or pseudo-relevant documents, they would help to identify the phrases likely to be associated with the overall query topics.

Importance: Importance indicates effectiveness related to retrieving relevant documents. To measure this, we leverage the features for predicting query performance (e.g., [27, 42, 113]). Given a phrase, we calculate its query clarity score [27] based on the query model directly derived from the query document or the relevance model used above. In addition, we use query scope [42], inverse document frequency, inverse collection term frequency, and word count, which are generally used for measuring pre-retrieval effectiveness. Note that the contribution of the topicality feature used in DSPApprox is the same as that of the query clarity feature we use. Since the diversification algorithm (described in Chapter 7.3) mainly uses the topic phrases for diversification, identifying highly effective phrases for retrieving relevant documents is important to increase the retrieval effectiveness of the final retrieval result.

Predictiveness: Predictiveness [65] measures the extent to which a term predicts the occurrences of other terms in a query vocabulary. We use two different types of query vocabulary: 1) all terms in the query document and not numbers, and 2) the terms that appeared in at least two pseudo-relevant documents and not numbers. Note that stop-words and section terms (e.g., "background" and "summary") are removed. First, predictiveness using the query document vocabulary is given as:

$$\text{Predict}_{QD}(p) = \frac{1}{Z} \sum_{v \in C_p^{QD}} P_{\text{w}}(p|v) \tag{7.14}$$

where C_p^{QD} indicates the set of terms that a term v co-occurs within the windows recognized in QD, w is the size of each window, $P_{\text{w}}(p|v)$ indicates such co-occurrence probability using w, and Z is the normalization factor.

Similarly predictiveness using the pseudo-relevant vocabulary is given as:

$$\text{Predict}_{PR}(p) = \frac{1}{Z} \sum_{v \in C_p^{PR}} P_{\text{w}}(p|v) \tag{7.15}$$

where C_p^{PR} indicates the co-occurrence term set by the windows identified in PR.

For each feature, the normalization factor is set by the size of the corresponding vocabulary, and we empirically set w to be 20 (as done in [28]). These predictiveness features are effective for extracting diverse phrases that can represent the terms in each topic vocabulary.

Cohesiveness: Cohesiveness quantifies the coherence of the terms in a phrase. We assume that the terms more co-occurring in query document contexts can be keywords. As an example, for the patent "Method and apparatus for providing content on a computer system based on usage profile" the terms "usage" and "profile" would frequently co-occur and may be effective to find its relevant documents. To capture this, we estimate the cohesiveness of the terms in a phrase by measuring relative co-occurrences of the terms. Like the predictiveness features, we use two different resources to measure cohesiveness, i.e., the query document and the pseudo-relevant documents. The cohesiveness using the query document is calculated as:

$$\text{Cohv}_{QD}(p) = \frac{P_{\text{w}}(w_1, w_2, \ldots, w_{|p|}|QD)}{\prod_{w \in p} P_{\text{w}}(w|C)} \tag{7.16}$$

where $P_{\text{w}}(w|C)$ indicates the window-based probability in the collection, and the size of the window is set the same way in predictiveness.

In addition, the feature using the pseudo-relevant documents is given as:

$$\mathrm{Cohv}_{PR}(p) = \frac{P_{\mathrm{w}}(w_1, w_2, \ldots, w_{|p|}|PR)}{\prod_{w \in p} P_{\mathrm{w}}(w|C)} \tag{7.17}$$

The cohesiveness features are useful for phrases containing terms related (i.e., co-occurring) to each other. If the terms are coherent in the query contexts (e.g., query document or pseudo-relevant documents), such terms would be keywords, and it is probable that those also appear in relevant documents.

7.5 Evaluation

To evaluate our approach, we conduct the experiments as follows. For each query document, we generate a baseline query (e.g., EX-RM described in Chapter 3.5) to produce an initial retrieval result. Then, we apply the diversification framework, described in Chapter 7.3, with topic phrases. To generate the topic phrases, we use either DSPApprox or the learning-to-rank method (proposed in Chapter 7.4.2). In the rest of this chapter, we provide more details of the experiments as well as experimental results.

7.5.1 Experimental Setup

For evaluation, we use two different patent test collections: USPTO and EPO (described in Chapter 3.3). Queries and documents are stemmed using the Krovetz stemmer [60] and stop-words are removed. We adopt baseline retrieval models to generate initial retrieval results. Among several query generation methods (e.g., PriorArtQuery and ReductionQuery), we select EX-RM (see Chapter 3.5) which can significantly outperforms the others in our initial experiments using the USPTO collection. To develop baseline retrieval results for EPO, we use PATATRAS (described in Chapter 3.6) which performed the best in the CLEF-IP 2010 [33]. More details of the settings are provided as follows.

(**Evaluation Metrics**) Since we attempt to diversify search results, we use conventional IR evaluation metrics to measure retrieval effectiveness as well as diversity metrics which measure "diversity" on retrieval results. For measuring relevance, we utilize MAP, Precision, NDCG, and Recall, which are typically used for adhoc retrieval tasks. In addition, PRES [74] is adopted, which is particularly designed for recall-oriented search tasks. This metric reflects the normalized recall incorporated with the quality of ranks of relevant documents observed within the maximum numbers of documents that the user examines (see Chapter 3.7). As diversity metrics, α-NDCG [25], ERR-IA (a variant of ERR [21]), NRBP [23], and subtopic recall (S-Recall) are used. These metrics penalize redundancy in retrieval results, i.e., how much of the information in each retrieved relevant document the user has already obtained in earlier ranks. Note that these have been used as standard metrics for diversity tasks in TREC [24]. Since the experiments are conducted for the patent domain and patent examiners (i.e., the search users) typically examine 100 patents on average in the invalidity search processes [53], we assume that the top 100 ranked documents are used to calculate the value of each metric.

(**Topic Relevance Judgment**) Although we develop the list of relevant documents for each query document (i.e., patent), the diversity metrics require the identification of query aspects for the relevant documents. In other words, for each query document, we need to group relevant documents if they belong to the same topic. The manual judgments required for this would be too laborious, and domain experts are essential because they can fully understand domain-specific topics. To alleviate this, we devise a semi-automatic method.

Each patent document contains a list of IPC codes that classify the document into a hierarchical taxonomy. As an example, the IPC code "H01S 3/14" indicates the patents related to "lasers characterized by the material used as the active medium". So, we exploit these codes to generate the topics of each query patent as follows. Given

a query patent, we first extract all IPC codes from its relevant documents. We sort the codes in descending order of the number of corresponding relevant documents, i.e., $c_a \succ c_b$ if $\sharp\text{rel}(c_a) > \sharp\text{rel}(c_b)$ where $\sharp\text{rel}(c)$ indicates the number of relevant documents containing the code c. Then, we scan from the top and remove the code if it covers all relevant documents (i.e., $\sharp rel(c) = |R|$) because such a code is too general and does not help to measure true diversity. After this, we assume that each remaining code can represent a topic for the query patent, and map relevant documents to their corresponding topics. In our experiments, the queries in USPTO and EPO include 4.94 and 8.66 topics, respectively.

Since any patent documents contain IPC codes, it could be argued that diversification can be performed using IPC codes that appear in initial retrieval results. That is, the topic set for each query patent is directly estimated by the IPC codes, i.e., $T = \{t_1 = c_a, t_2 = c_b, \ldots\}$. However, the topics of IPC codes are very abstract and general, e.g., "H01F 1/01" means "magnetic bodies of inorganic materials". We assume that true topics in a query patent are more specific and concrete. Thus, we generate topic phrases for representing detailed topics (as described in Chapter 7.2). IPC codes are treated as a crude estimation for true topics, and used for evaluating diversity in retrieval results.

In training of our phrase ranking model (described in Chapter 7.4.2.1), we need to use the sets of relevant documents grouped by their IPC codes (i.e., relevant documents and their IPC codes are necessary). However, in testing, we do not require IPC codes because the trained ranking model automatically generates the ranks of given candidate phrases by using the features described in Chapter 7.4.2.2. Such training and testing scheme is typically used in many supervised learning frameworks (e.g., learning-to-rank document retrieval [17, 105]), and in real systems, only the ranking models trained using relevant documents are used to generate the ranked list of phrases.

(**Parameter Settings**) The diversification algorithm described in Chapter 7.3 is applied to the top 200 documents in initial retrieval results, i.e., $K = 200$. For web search tasks, the PM-2 performed better with $K = 50$ [29], but prior-art search requires the examination of more documents (e.g., top 100 documents). Thus, we empirically set $K = 200$, and consequently, the topic phrase identification techniques (i.e., DSPApprox and the learning-to-rank method) are also performed with these top 200 documents. In addition, we need to tune two free parameters for this algorithm, i.e., λ and μ (see Eq. (7.3) and Eq. (7.4)). For this, we consider each value in the range of $[0.1, 1.0]$ with an increment of 0.1, and 10-fold cross-validation is performed with random partitioning. The topic phrase identification techniques also require the free parameter k, which indicates the number of topic phrases to be extracted from the candidate pool. We use multiple values of k $= \{5, 10, 20, 40, 60, 80, 100\}$ and the 10-fold cross-validation is applied. The learning-to-rank topic identification is also performed using this 10-fold cross-validation. Note that the average number of phrases in the pool is 487.17 and 313.89 over USPTO and EPO query patents, respectively.

7.5.2 Retrieval Results

We evaluate our approach in terms of retrieval effectiveness and diversity. We first verify the retrieval effectiveness of the ranked results obtained by each method. Table 7.2 shows the evaluation results using both USPTO and EPO. In that, LTR and DSP denote diversification using the learning-to-rank topic identification and DSPApprox, respectively. In each row, a significant improvement over each method is marked by the first letter of the method, e.g., B indicates an improvement over Baseline, and the paired t-test is performed with $p < 0.05$. Also, the best performance is marked by bold. For each retrieval result, we measure overall performance (e.g., MAP and

Recall) as well as early precision (i.e., Precision at 20 and NDCG at 20), and report an average value of each metric over the query documents in each corpus.

Table 7.2: Retrieval Results using Relevance Metrics

Corpus	**Metric**	**Baseline**	**DSP**	**LTR**
USPTO	MAP	0.1221 (0.0%)	0.1337^{B} (+9.50%)	$\mathbf{0.1516}^{BD}$ (+19.46%)
	PRES	0.2766 (0.0%)	0.2789 (+0.81%)	$\mathbf{0.2985}^{BD}$ (+7.32%)
	Precision@20	0.1503 (0.0%)	0.1530 (+1.80%)	$\mathbf{0.1687}^{BD}$ (+10.91%)
	NDCG@20	0.2087 (0.0%)	0.2176^{B} (+4.26%)	$\mathbf{0.2527}^{BD}$ (+17.41%)
	Recall	0.4261 (0.0%)	0.4282 (+0.49%)	**0.4285** (+0.56%)
EPO	MAP	0.2414 (0.0%)	0.2482^{B} (+2.78%)	$\mathbf{0.2536}^{BD}$ (+5.30%)
	PRES	0.4148 (0.0%)	0.4184 (+0.86%)	$\mathbf{0.4292}^{BD}$ (+3.46%)
	Precision@20	0.2857 (0.0%)	0.2945^{B} (+3.08%)	$\mathbf{0.3010}^{B}$ (+5.36%)
	NDCG@20	0.3440 (0.0%)	0.3562^{B} (+3.55%)	$\mathbf{0.3630}^{B}$ (+5.52%)
	Recall	0.5159 (0.0%)	0.5166 (+0.14%)	**0.5209** (+0.97%)

First, our diversification framework can provide significant improvements relative to the baseline retrieval results on many relevance metrics, while recall does not significantly increase. That is, the diversification keeps the relevant documents that appear in the initial ranked results, and effectively promotes their ranks. This is important because, using the diversification, search users are more likely to find relevant documents in early ranks. In particular, the MAP and NDCG scores increase if we use either LTR or DSP for the topic phrase identification, which means that the diversification technique is useful for domain-specific search. Second, LTR is more effective than DSP. In USPTO, LTR significantly outperforms DSP in all cases (ex-

cept Recall), but in EPO it is better in terms of only MAP and PRES. Comparing to EPO, the values of early precision metrics (e.g., NDCG and Precision) in USPTO are dramatically lower, i.e., many relevant documents retrieved in USPTO are ranked out of 20. Thus, the baseline results of USPTO may provide more chances to promote relevant documents which are initially found below rank 20, and the topic phrases identified by LTR would effectively work for such relevant documents.

Next, we evaluate the "diversity" of retrieval results obtained by each method. Specifically, we measure the values of α-NDCG, ERR-IA, S-Recall at early ranks (i.e., top 20) and overall ranks. Table 3 presents the diversity-based evaluation results. Note that each retrieval result is truncated at rank 100.

First, for both collections, our diversification approach is effective for generating significantly more diversified results. The diversity performance in USPTO is especially improved, e.g., +26.10% is achieved in terms of NRBP. This result indicates that the diversification can increase the ranks of relevant documents related to diverse topics, and enabling the user to recognize the diverse aspects of query patents. Second, the sub-topic recall at rank 100 (i.e., S-Recall@100) is less improved by the diversification. We believe the cause of this result is that within rank 100, the baseline has already found sufficient amounts of each topic from retrieved relevant documents. Thus, the diversification may not find new topics not covered by the initial retrieval results. However, within rank 20, significantly more topics are extracted by the diversification, i.e., S-Recall@20. Third, the diversification performance in USPTO looks better than that in EPO whereas the retrieval effectiveness measured in EPO is much better than that measured in USPTO (see Table 7.2). This is because the relevant documents in EPO includes more topics, i.e., the (average) number of topics in relevant documents of USPTO and EPO is 4.94 and 8.66, respectively. Thus, the retrieval results for USPTO easily contain relatively more topics, i.e., the ratio of found topics to the whole topics. Lastly, different from the relevance results (Table

Table 7.3: Diversification Results

Corpus	**Metric**	**Baseline**	**DSP**	**LTR**
USPTO	NRBP	0.1662 (0.0%)	0.1814^{B} (+9.18%)	$\mathbf{0.2248}^{\mathrm{BD}}$ (+26.10%)
	α-NDCG@20	0.3441 (0.0%)	0.3596 (+4.53%)	$\mathbf{0.4179}^{\mathrm{BD}}$ (+17.70%)
	α-NDCG@100	0.4158 (0.0%)	0.4306 (+3.55%)	$\mathbf{0.4785}^{\mathrm{BD}}$ (+13.10%)
	ERR-IA@20	0.1948 (0.0%)	0.2089^{B} (+7.24%)	$\mathbf{0.2499}^{\mathrm{BD}}$ (+22.10%)
	ERR-IA@100	0.2015 (0.0%)	0.2155 (+6.95%)	**0.2557** (+21.20%)
	S-Recall@20	0.5299 (0.0%)	0.5443 (+2.72%)	**0.5678** (+6.70%)
	S-Recall@100	0.7074 (0.0%)	0.7088 (+0.19%)	**0.7186** (+1.50%)
EPO	NRBP	0.1312 (0.0%)	$\mathbf{0.1433}^{\mathrm{BL}}$ (+9.22%)	0.1368^{B} (+4.26%)
	α-NDCG@20	0.3439 (0.0%)	0.3601^{B} (+4.72%)	$\mathbf{0.3627}^{\mathrm{B}}$ (+5.47%)
	α-NDCG@100	0.4345 (0.0%)	0.4476^{B} (+3.00%)	$\mathbf{0.4493}^{\mathrm{B}}$ (+3.39%)
	ERR-IA@20	0.1576 (0.0%)	$\mathbf{0.1692}^{\mathrm{B}}$ (+7.37%)	0.1659^{B} (+5.26%)
	ERR-IA@100	0.1650 (0.0%)	$\mathbf{0.1766}^{B}$ (+7.01%)	0.1729^{B} (+4.77%)
	S-Recall@20	0.3815 (0.0%)	0.39020^{B} (+2.75%)	$\mathbf{0.4054}^{\mathrm{BD}}$ (+6.26%)
	S-Recall@100	0.6256 (0.0%)	0.6257 (+0.03%)	**0.6267** (+0.18%)

7.2), LTR is significantly better than DSP only when using the USPTO collection. In EPO, significant differences between the results obtained by both methods are rarely observed. This is because LTR uses the ranking model trained by relevant documents, which can select more effective phrases, whereas DSP only utilizes the topicality and predictiveness in an unsupervised manner. Since a supervised learning approach typically takes advantages from a labeled data (i.e., relevant documents), LTR can be useful when relevant documents are provided. In summary, the diversification approach we used can improve retrieval effectiveness as well as the diversity of patent search results.

7.5.3 Feature Analysis

We now provide an analysis of features used in the learning-to-rank topic identification (LTR) described in Chapter 7.4.2. As summarized in Table 7.1, we use four different types of features for LTR, and conduct another experiment to examine the influence of each feature type for diversification. Since calculating the effects of some features on the topic phrase identification is very difficult, we indirectly measure their effectiveness by performing diversification using the topic phrases generated by the target features.

We first extract topic phrases by LTR using all features with 10-fold cross-validation, and diversify initial retrieval results. Then, following the same partitions, we train the ranking model with all features except for one feature type, and run the diversification with the topic phrases extracted by this model. After this, we observe the final performance change by the feature drop, i.e., how much the topic phrase identification depends on the dropped feature type. Note that the parameters for this experiment are the same as used previously.

Table 7.4 shows the feature analysis using the USPTO collection where LTR is notably effective. In that, we use MAP and α-NDCG for the analysis, and like

the previous experiments, each retrieval result contains the top 100 documents. In each column, a * indicates a significant different from {All}, and the paired t-test is performed with $p < 0.05$.

Table 7.4: Feature Analysis Results

Features	**MAP**	**α-NDCG**
{All}	0.1516 (0.0%)	0.4785 (0.0%)
{All} − {Cohesiveness}	0.1472 (−2.90%)	0.4729 (−1.16%)
{All} − {Relevance}	0.1455 (−4.02%)	0.4606* (−3.75%)
{All} − {Predictiveness}	0.1438* (−5.12%)	0.4621* (−3.42%)
{All} − {Importance}	0.1415* (−6.66%)	0.4521* (−5.51%)

First, all the features we used seem to have positive effects on diversification. Whenever a feature is dropped, the value of every metric decreases. Second, the predictiveness and importance features look more influential than the others since these features can cause a significant decrease in MAP. Considering α-NDCG, the relevance features are also significant. Since we reuse the predictiveness already proposed for DSPApprox, it is somewhat obvious that the predictiveness features are important for the topic phrase identification. However, we additionally identify other significant features, i.e., relevance and importance that represent the relevance of phrases to query patents and their predicted effectiveness to retrieve relevant documents (i.e., query performance predictors).

7.5.4 Qualitative Analysis

We now provide a qualitative analysis of our topic phrase identification using an example. Table 7.5 shows the top 5 topic phrases generated for an example query patent (which is in the same as Table 1.1). The application in this patent provides

profiled information about computer system usage, and several modules such as Basic Input Output System (BIOS), Operating System (OS), and Profile Server make up the whole system. For this query patent, the baseline performs reasonably well (its

Table 7.5: Examples of Topic Phrase Identification

Query Patent		
Title: Method and apparatus for providing content on a computer system based on usage profile. **Abstract** A method and apparatus for determining a **computer system usage profile** ... A **basic input output system (BIOS) module** and/or an **operating system module** obtain computer system usage profile information by tracking events such as the frequency of re-boots, the time required to boot-up and shut-down the operating system ... data is collected and communicated to a **profile server** ...		
Initial Retrieval		
AvePrec	0.1288	
α-NDCG	0.4058	
Diversification		
	DSP	**LTR**
AvePrec	0.1409	0.1939
α-NDCG	0.4560	0.6155
Rank 1	computer device	usage profile information
Rank 2	event	BIOS module
Rank 3	execution	remote network
Rank 4	OS	OS profile module
Rank 5	microprocessor	boot process

average precision score is slightly higher than MAP over all queries (see Table 7.2)), and diversification is effective for improving the initial retrieval result.

One observation is that DSPApprox (DSP) can identify phrases that describe other query terms, i.e., phrases with high predictiveness. For example, "computer device" appears to be highly representative for the peripheral devices used for BIOS, e.g., printer and keyboard, and "event" stands for the actions recorded in the usage profile, e.g., re-boot and shut-down. On the other hand, our learning-to-rank method (LTR) can recognize key phrases that describe significant topics in the query patent

and that are more effective for retrieving relevant documents. As an example, "BIOS module" and "OS module" are important components for the application, and as discussed in Chapter 1.2 (using Table 1.1), we assume that such components may form query topics. In addition, these phrases are related to several relevant documents for this query patent. Moreover, the other phrases, e.g., "remote network" and "boot process", are also effective for retrieving relevant documents such as "Generic remote boot for networked workstations by creating local bootable code image" (the title of a relevant document for this query patent).

Another interesting observation is that DSPApprox favors unigram phrases. Although we use the same phrase pool for both methods, unigram phrases are more highly ranked by DSPApprox. This bias can be caused by the high predictiveness scores of one-word phrases since they tend to co-occur with more terms than multi-word phrases. The LTR method uses a supervised learning framework, and the weight on the predictiveness feature can be effectively controlled.

7.6 Summary

In this chapter, we addressed the problem of diversifying patent search results based on query patents. To solve this, we propose a result-level diversification approach using topic phrase identification. Given an initial retrieval result of each query document, we identify topic phrases to represent underlying query topics, and diversify based on the identified phrases. Through experiments, we showed that this phrase-level diversification can improve patent search results in terms of retrieval effectiveness and diversity. In addition, we devise a learning-to-rank method to identify topic phrases, and verify its effectiveness in comparison to the state-of-the-art topic term identification algorithm.

CHAPTER 8

CONCLUSIONS

8.1 Overview

This chapter summarizes the dissertation. In Chapter 8.2, we again describe the important problems for improving domain-specific searches and highlight key results that verify the effectiveness of our proposed approaches. Chapter 8.3 restates the main contributions of this dissertation, and Chapter 8.4 discusses the limitations of our approaches and the future directions for improvements.

8.2 Summary

In this dissertation, we propose IR techniques for improving the domain-specific users' search experiences. The techniques we introduce are designed for the unique characteristics of domain-specific searches (e.g., patent retrieval, academic literature search, and medical search). As discussed in Chapter 1.1, an important characteristic of domain-specific IR is that users manually formulate search queries after reading query documents (e.g., new patents in prior-art search and new project descriptions for literature search). To reduce the difficulty of formulating effective queries, we propose query generation methods based on query documents. In addition, we devise query suggestion methods to help users formulate more effective queries. To resolve the diversity issue (described in Chapter 1.2), we introduce two different diversification approaches: (1) query-side diversification that generates diverse queries and (2) result-level diversification that directly diversifies retrieval results.

In Chapter 4, to formulate effective queries for professional users, we propose a method to generate effective Boolean queries. We showed that this Boolean query generation approach can generate a substantial number of Boolean queries, i.e., about 200 queries are generated for each query document (as reported in Table 4.3). In addition, the best query among the top 10 suggested queries significantly improves retrieval effectiveness compared to the baseline (PriorArtQuery) (see Table 8.1). Note that a † indicates a significant improvement over the baseline, and the paired t-test is performed with $p < 0.05$.

Table 8.1: Boolean Query Retrieval Performance

Domain	**Method**	**F1@100**
Patent	Baseline	0.1184
	Best of top-10 queries	$0.1402^{\dagger}$
Medical	Baseline	0.2636
	Best of top-10 queries	$0.3089^{\dagger}$

In Chapter 5, we propose a method for generating phrasal-concept queries (e.g., {*"structural paraphrase generation" "large corpora" "multiple sequence alignment"*}) to improve academic literature search. In the retrieval experiments using the ACL collection (described in Chapter 3.3), we verified that phrasal-concept queries are significantly better than the baseline keyword queries (LCE and MLE) (see Table 8.2). Note that an $^{\text{LM}}$ denotes significant improvements over the baselines (i.e., LCE and MLE), and the paired t-test is performed with $p < 0.05$.

Table 8.2: Phrasal-Concept Query Retrieval Performance using ACL

Method	**NDCG@100**	**MAP**
LCE	0.4874	0.2638
MLE	0.5086	0.2744
PHRASAL-CONCEPT	0.5301^{LM}	0.2899^{LM}

In order to diversify search results, we propose a diverse query generation method that can suggest a list of diverse queries. In the experiments, we showed that our query-side diversification method is effective for retrieving more relevant documents. In Table 8.2, the baseline queries are generated by the decision tree-based method (described in Chapter 4), and the diverse queries are also generated by the same method but query aspects are considered in the generation; we first extract multiple query aspects from the query document, and then each query aspect is used to generate the queries by the decision tree-based method. Note that a † indicates a significant improvement over the baseline and the paired t-test is performed with $p < 0.05$.

Table 8.3: Retrieval Performance of Diverse Queries

Domain	**Metric**	**Baseline Queries**	**Diverse Queries**
Patent	SNR@100	0.1989	$0.2509^{\dagger}$
	NSDCG@100	0.0959	$0.1212^{\dagger}$
Academic	SNR@100	0.6351	$0.7392^{\dagger}$
	NSDCG@100	0.3099	$0.4457^{\dagger}$

In Chapter 7, we describe a phrase-level diversification framework that can identify topic phrases and directly diversify search results based on the identified phrases. By comparing with the baseline (EX-RM), we showed that our diversification method can improve the retrieval effectiveness and diversity of search results (see Table 8.4 & 8.5). Note that a † indicates a significant improvement over the baseline and the paired t-test is performed with $p < 0.05$.

Table 8.4: Diversification Performance by Relevance Metrics

Corpus	**Method**	**MAP**	**PRES**	**Recall**
USPTO	Baseline	0.1221	0.2766	0.4261
	Diversification	$0.1516^{\dagger}$	$0.2985^{\dagger}$	$0.4285^{\dagger}$
EPO	Baseline	0.2414	0.4148	0.5159
	Diversification	$0.2536^{\dagger}$	$0.4292^{\dagger}$	$0.5209^{\dagger}$

Table 8.5: Diversification Performance by Diversity Metrics

Corpus	**Method**	**NRBP**	**α-NDCG@100**	**ERR-IA@100**
USPTO	Baseline	0.1662	0.4158	0.2015
	Diversification	0.2248†	0.4785†	0.2557†
EPO	Baseline	0.1312	0.4345	0.1650
	Diversification	0.1368†	0.4493†	0.1729†

Based on these results, we examined the effectiveness of our query generation and diversification methods. By conducting retrieval experiments on various search domains, the evaluations were performed in more robust ways, and the proposed techniques were shown to be effective for enhancing the search quality of domain-specific IR.

8.3 Contributions

To recap, the major contributions of our work are as follows.

1. **Evidence showing that domain-specific searches are improved by resolving three issues: (1) query generation, (2) query suggestion and formulation, and (3) search result diversification**. As discussed in Chapter 1.2, these three issues are important for domain-specific searches, and our query generation and diversification methods are designed to resolve these issues. Moreover, the experimental results verified that the proposed approaches are effective for improving domain-specific searches.

2. **Methods to generate effective queries based on documents**. We proposed three different query generation methods based on query documents: (1) Boolean query generation, (2) phrasal-concept query generation, and (3) diverse query generation. We showed that these approaches are effective for improving

various domain-specific search tasks (e.g., prior-art search, academic literature search, and medical reference retrieval).

3. **Query formulation in user-preferred representations**. Our methods can generate effective Boolean queries preferred by professional users (e.g. patent examiners) and phrasal-concept queries useful for academic users (e.g., research scientists). Through the user experiments described in Chapter 5.5, we verified the effectiveness of phrasal-concept queries for academic literature search.

4. **Query-side diversification methods to generate diverse search results**. To resolve the diversity issue, we proposed the method to generate diverse queries, and in overall session-retrieval results, diverse queries can retrieve significantly more relevant documents than baseline queries (that do not consider the diversity).

5. **Search result diversification frameworks applied to domain-specific searches**. We exploited the term-level diversification framework (described in [28]) for diversifying domain-specific search results. To improve diversification performance in domain-specific searches, we modified the diversification algorithm (see Chapter 7.3) and proposed the learning-to-rank method to identify topic phrases.

6. **Algorithms to identify important topics (or aspects) from documents**. To identify query aspects (i.e., sets of query document terms; see Chapter 6.2), we used the term clustering method described in Chapter 6.3.1. In addition, we proposed a similarity learning method to predict the similarity between query terms. For extracting topic phrases (i.e., phrases to represent query topics), we proposed the learning-to-rank approach to rank topic phrases by considering topicality, predictiveness, and various features (see Chapter 7.4).

8.4 Future Work

We now describe the limitations of each proposed method and discuss further improvements.

8.4.1 Improvements for Boolean Query Generation

In this method, we primarily focus on using conjunction ('AND') and negation ('NOT') operators for generating Boolean queries because these operators have more impact on patent retrieval performance for very detailed documents. However, professional searchers often use the disjunction ('OR') operator for representing synonym groups. In fact, Boolean Conjunctive Normal Form (CNF), i.e., a conjunction of disjunctions where each disjunction contains a term and its synonyms, is effective to resolve term mismatch problems between queries and relevant documents in legal search [112]. Thus, adding synonym structure into the current suggestion framework may provide further improvements and is useful for extending to the legal domain. In addition to this, we can elaborate the decision tree-based query generation method (see Chapter 4.3) as follows.

First we can generate more effective queries by focusing on the terms discriminant in more important (pseudo) relevant documents. In our method, we equally treat each pseudo-relevant example regardless of its rank in the initial retrieval. However, we can consider different weights on the pseudo-relevant documents by their ranking scores, and this could help the generated queries to focus on more effective terms (i.e., effective to retrieve more (pseudo) relevant documents). Second, we can simply identify more effective Boolean queries without learning the Boolean query ranking model (described in Chapter 4.4). We can predict the retrieval effectiveness of each generated query by measuring its information gain on pseudo-relevant documents. In other words, more effective Boolean query would be more precise to imply more

pseudo-relevant documents. By doing this, we could reduce the complexity of our method and resolve the difficulty of optimizing the ranking model.

8.4.2 Improvements for Phrasal-Concept Query Generation

The merit of this approach is reproducibility and generalizability. To generate effective concept queries, we mainly use the concepts identified from pseudo-relevant documents, and similarities recognized within the corpus. In other words, external resources or manually constructed data are not required. However, as Bai et al. studied [7], query contexts mined from external ontologies may help to identify more effective concepts and their relationships. Thus, it can be useful to explore global information-based approaches applicable for the queries in academic literature search. In addition, using "semantic" concepts for query generation can be helpful because semantic entities (e.g., author names and domain-specific terminology) may be crucial to creating more effective and more "interesting" queries from the user's perspective. We observed that several author names are extracted as expansion terms in the MLE queries which use machine learning algorithms to select discriminant features (i.e., expansion terms).

8.4.3 Improvements for Diverse Query Generation

The complexity of our diverse query generation model can be significant for a practical system. In general, domain-specific users spend much more time to complete a single search task (e.g., patent examiners use about 12 hours to validate a new patent [53]), and the efficacy (rather than the efficiency) is more important in this method. However, reducing the complexity may help users to find more relevant documents because users can examine more retrieved documents in a given amount of time. The cost of running the diverse query generation model is mainly based on three different parts: (1) query aspect identification, (2) query generation, and (3) diverse query suggestion. The complexity of the latter two parts can be simply

improved by controlling the number of generated queries, the size of pseudo-relevant documents or query vocabulary. To reduce the cost of identifying query aspects from a query document, we may need to consider more efficient term clustering algorithms. Spectral clustering (currently used in our method) can be less efficient for a large set of aspect terms (i.e., a long query document). To alleviate this, we used the parallel spectral clustering approach implemented using distributed systems [22]. In addition, instead of using spectral clustering, we can consider other efficient clustering algorithms (e.g., hierarchical clustering) and further exploration may require to find optimal parameter settings (e.g., linkage criteria).

8.4.4 Improvements for Search Result Diversification

In our diversification framework, we mainly exploit human-labeled topic information (e.g., IPC[1] codes annotated in patents) for evaluating the diversity of search results and learning topic phrases. However, this approach forces us to use only patent test collections because only patent documents provide this type of manual coding. So, for applying this framework to other domains, we require a more general proxy to represent topics in relevant documents. One way to solve this is using document clustering techniques [2], i.e., clustering relevant documents by their topics. In this method, exploring effective algorithms to generate more accurate clusters is crucial, and domain-specific knowledge bases (e.g., medical term ontology) may help to improve the clustering accuracy by providing semantic features.

[1]International Patent Classification (http://www.wipo.int/classifications/ipc/en/)

BIBLIOGRAPHY

[1] Abdul-jaleel, Nasreen, Allan, James, Croft, W. Bruce, Diaz, O, Larkey, Leah, Li, Xiaoyan, Smucker, Mark D., and Wade, Courtney. UMass at TREC 2004: Novelty and hard. In *Proceedings of the 13th Text Retrieval Conference* (2004), TREC-13.

[2] Aggarwal, Charu C., and Zhai, ChengXiang. A survey of text clustering algorithms. In *Mining Text Data*, Charu C. Aggarwal and ChengXiang Zhai, Eds. Springer US, 2012, pp. 77–128.

[3] Agrawal, Rakesh, Gollapudi, Sreenivas, Halverson, Alan, and Ieong, Samuel. Diversifying search results. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining* (New York, NY, USA, 2009), WSDM '09, ACM, pp. 5–14.

[4] Andrew, Galen, and Gao, Jianfeng. Scalable training of l1-regularized log-linear models. In *Proceedings of the 24th International Conference on Machine Learning* (New York, NY, USA, 2007), ICML '07, ACM, pp. 33–40.

[5] Azzopardi, Leif, Vanderbauwhede, Wim, and Joho, Hideo. Search system requirements of patent analysts. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval* (New York, NY, USA, 2010), SIGIR '10, ACM, pp. 775–776.

[6] Baeza-Yates, Ricardo, Hurtado, Carlos, and Mendoza, Marcelo. Query recommendation using query logs in search engines. In *Proceedings of the 2004 International Conference on Current Trends in Database Technology* (Berlin, Heidelberg, 2004), EDBT'04, Springer-Verlag, pp. 588–596.

[7] Bai, Jing, Nie, Jian-Yun, Cao, Guihong, and Bouchard, Hugues. Using query contexts in information retrieval. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (New York, NY, USA, 2007), SIGIR '07, ACM, pp. 15–22.

[8] Balasubramanian, Niranjan, Allan, James, and Croft, W. Bruce. A comparison of sentence retrieval techniques. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (New York, NY, USA, 2007), SIGIR '07, ACM, pp. 813–814.

[9] Bashir, Shariq, and Rauber, Andreas. Improving retrievability of patents in prior-art search. In *Proceedings of the 32Nd European Conference on Advances in Information Retrieval* (Berlin, Heidelberg, 2010), ECIR'2010, Springer-Verlag, pp. 457–470.

[10] Basu, Chumki, Hirsh, Haym, and Cohen, William W. Technical paper recommendation: A study in combining multiple information sources. *Journal of Artificial Intelligence Research 14* (2001), 231–252.

[11] Bendersky, Michael, and Croft, W. Bruce. Discovering key concepts in verbose queries. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (New York, NY, USA, 2008), SIGIR '08, ACM, pp. 491–498.

[12] Bendersky, Michael, Croft, W. Bruce, and Smith, David A. Two-stage query segmentation for information retrieval. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval* (New York, NY, USA, 2009), SIGIR '09, ACM, pp. 810–811.

[13] Bethard, Steven, and Jurafsky, Dan. Who should I cite: Learning literature search models from citation behavior. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management* (New York, NY, USA, 2010), CIKM '10, ACM, pp. 609–618.

[14] Bhatia, Sumit, Majumdar, Debapriyo, and Mitra, Prasenjit. Query suggestions in the absence of query logs. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval* (New York, NY, USA, 2011), SIGIR '11, ACM, pp. 795–804.

[15] Bird, Steven, Dale, Robert, Dorr, Bonnie J., Gibson, Bryan, Joseph, Mark T., yen Kan, Min, Lee, Dongwon, Powley, Brett, Radev, Dragomir R., and Tan, Yee Fan. The ACL anthology reference corpus: A reference dataset for bibliographic research in computational linguistics. In *Proceedings of International Conference on Language Resources and Evaluation* (2008), LREC'08.

[16] Brin, Sergey, and Page, Lawrence. The anatomy of a large-scale hypertextual web search engine. *Comput. Netw. ISDN Syst. 30*, 1-7 (Apr. 1998), 107–117.

[17] Cao, Yunbo, Xu, Jun, Liu, Tie-Yan, Li, Hang, Huang, Yalou, and Hon, Hsiao-Wuen. Adapting ranking SVM to document retrieval. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (New York, NY, USA, 2006), SIGIR '06, ACM, pp. 186–193.

[18] Carbonell, Jaime, and Goldstein, Jade. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (New York, NY, USA, 1998), SIGIR '98, ACM, pp. 335–336.

[19] Carterette, Ben, and Chandar, Praveen. Probabilistic models of ranking novel documents for faceted topic retrieval. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management* (New York, NY, USA, 2009), CIKM '09, ACM, pp. 1287–1296.

[20] Cartright, Marc-Allen, Feild, Henry A., and Allan, James. Evidence finding using a collection of books. In *Proceedings of the 4th ACM Workshop on Online Books, Complementary Social Media and Crowdsourcing* (New York, NY, USA, 2011), BooksOnline '11, ACM, pp. 11–18.

[21] Chapelle, Olivier, Metlzer, Donald, Zhang, Ya, and Grinspan, Pierre. Expected reciprocal rank for graded relevance. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management* (New York, NY, USA, 2009), CIKM '09, ACM, pp. 621–630.

[22] Chen, Wen-Yen, Song, Yangqiu, Bai, Hongjie, Lin, Chih-Jen, and Chang, Edward Y. Parallel spectral clustering in distributed systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence 33*, 3 (2008), 568–586.

[23] Clarke, Charles L., Kolla, Maheedhar, and Vechtomova, Olga. An effectiveness measure for ambiguous and underspecified queries. In *Proceedings of the 2Nd International Conference on Theory of Information Retrieval: Advances in Information Retrieval Theory* (Berlin, Heidelberg, 2009), ICTIR '09, Springer-Verlag, pp. 188–199.

[24] Clarke, Charles L. A., Craswell, Nick, Soboroff, Ian, and Cormack, Gordon V. Overview of the TREC 2010 web track. In *Proceedings of the nineteenth Text Retrieval Conference* (2010), TREC-19.

[25] Clarke, Charles L.A., Kolla, Maheedhar, Cormack, Gordon V., Vechtomova, Olga, Ashkan, Azin, Büttcher, Stefan, and MacKinnon, Ian. Novelty and diversity in information retrieval evaluation. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (New York, NY, USA, 2008), SIGIR '08, ACM, pp. 659–666.

[26] Craswell, Nick, and Szummer, Martin. Random walks on the click graph. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (New York, NY, USA, 2007), SIGIR '07, ACM, pp. 239–246.

[27] Cronen-Townsend, Steve, Zhou, Yun, and Croft, W. Bruce. Predicting query performance. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (New York, NY, USA, 2002), SIGIR '02, ACM, pp. 299–306.

[28] Dang, Van, and Croft, Bruce W. Term level search result diversification. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval* (New York, NY, USA, 2013), SIGIR '13, ACM, pp. 603–612.

[29] Dang, Van, and Croft, W. Bruce. Diversity by proportionality: An election-based approach to search result diversification. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval* (New York, NY, USA, 2012), SIGIR '12, ACM, pp. 65–74.

[30] Dang, Van, Xue, Xiaobing, and Croft, W. Bruce. Inferring query aspects from reformulations using clustering. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management* (New York, NY, USA, 2011), CIKM '11, ACM, pp. 2117–2120.

[31] Efron, Bradley, Hastie, Trevor, Johnstone, Iain, and Tibshirani, Robert. Least angle regression. *The Annals of Statistics 32*, 2 (04 2004), 407–499.

[32] Fall, C. J., Törcsvári, A., Benzineb, K., and Karetka, G. Automated categorization in the international patent classification. *SIGIR Forum 37*, 1 (Apr. 2003), 10–25.

[33] Florina, Piroi. CLEF-IP 2010: Prior-art candidates search evaluation summary. Tech. rep., IRF-TR-2010-00003, Information Retreival Facility, 2010.

[34] Florina, Piroi, and Tait, John. Clef-ip 2010: Retrieval experiments in the intellectual property domain. Tech. rep., IRF-TR-2010-00005, Information Retreival Facility, 2010.

[35] Fonseca, Bruno M., Golgher, Paulo, Pôssas, Bruno, Ribeiro-Neto, Berthier, and Ziviani, Nivio. Concept-based interactive query expansion. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management* (New York, NY, USA, 2005), CIKM '05, ACM, pp. 696–703.

[36] Fujiii, Atsushi, Iwayama, Makoto, and Kando, Noriko. Overview of the patent retrieval task at the NTCIR-6 workshop. In *Proceedings of NTCIR-6 Workshop Meeting* (May 2007), NTCIR-6, pp. 359 – 365.

[37] Ganguly, Debasis, Leveling, Johannes, and Jones, Gareth J.F. United we fall, divided we stand: A study of query segmentation and PRF for patent prior art search. In *Proceedings of the 4th Workshop on Patent Information Retrieval* (New York, NY, USA, 2011), PaIR '11, ACM, pp. 13–18.

[38] Ganguly, Debasis, Leveling, Johannes, Magdy, Walid, and Jones, Gareth J.F. Patent query reduction using pseudo relevance feedback. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management* (New York, NY, USA, 2011), CIKM '11, ACM, pp. 1953–1956.

[39] Garey, Michael R., and Johnson, David S. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

[40] Gobeill, Julien, Teodoro, Douglas, Pasche, Emilie, and Ruch, Patrick. Report on the TREC 2009 experiments: Chemical IR track. In *Proceedings of the Eighteenth Text Retrieval Conference* (2009), TREC-18.

[41] Hasan, Mohammad Al, Parikh, Nish, Singh, Gyanit, and Sundaresan, Neel. Query suggestion for e-commerce sites. In *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining* (New York, NY, USA, 2011), WSDM '11, ACM, pp. 765–774.

[42] He, Ben, and Ounis, Iadh. Inferring query performance using pre-retrieval predictors. In *In Proc. Symposium on String Processing and Information Retrieval* (2004), Springer Verlag, pp. 43–54.

[43] He, Jing, Nie, Jian-Yun, Lu, Yang, and Zhao, Wayne Xin. Position-aligned translation model for citation recommendation. In *Proceedings of the 19th International Conference on String Processing and Information Retrieval* (Berlin, Heidelberg, 2012), SPIRE'12, Springer-Verlag, pp. 251–263.

[44] He, Qi, Pei, Jian, Kifer, Daniel, Mitra, Prasenjit, and Giles, Lee. Context-aware citation recommendation. In *Proceedings of the 19th International Conference on World Wide Web* (New York, NY, USA, 2010), WWW '10, ACM, pp. 421–430.

[45] Hearst, Marti A. Texttiling: Segmenting text into multi-paragraph subtopic passages. *Comput. Linguist. 23*, 1 (Mar. 1997), 33–64.

[46] Hersh, William, Buckley, Chris, Leone, T. J., and Hickam, David. Ohsumed: An interactive retrieval evaluation and new large test collection for research. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (New York, NY, USA, 1994), SIGIR '94, Springer-Verlag New York, Inc., pp. 192–201.

[47] Huang, Xiangji, Huang, Yan Rui, Wen, Miao, An, Aijun, Liu, Yang, and Poon, J. Applying data mining to pseudo-relevance feedback for high performance text retrieval. In *Data Mining, 2006. ICDM '06. Sixth International Conference on* (Dec 2006), pp. 295–306.

[48] Iwayama, Makoto, Fujii, Atsushi, Kando, Noriko, and Marukawa, Yuzo. An empirical study on retrieval models for different document genres: Patents and newspaper articles. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval* (New York, NY, USA, 2003), SIGIR '03, ACM, pp. 251–258.

[49] Järvelin, Kalervo, and Kekäläinen, Jaana. Cumulated gain-based evaluation of IR techniques. *ACM Trans. Inf. Syst. 20*, 4 (Oct. 2002), 422–446.

[50] Järvelin, Kalervo, Price, Susan L., Delcambre, Lois M. L., and Nielsen, Marianne Lykke. Discounted cumulated gain based evaluation of multiple-query IR sessions. In *Proceedings of the IR Research, 30th European Conference on Advances in Information Retrieval* (2008), ECIR'08, Springer-Verlag, pp. 4–15.

[51] Joachims, Thorsten. Optimizing search engines using clickthrough data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (New York, NY, USA, 2002), KDD '02, ACM, pp. 133–142.

[52] Joachims, Thorsten. Training linear SVMs in linear time. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (New York, NY, USA, 2006), KDD '06, ACM, pp. 217–226.

[53] Joho, Hideo, Azzopardi, Leif A., and Vanderbauwhede, Wim. A survey of patent users: An analysis of tasks, behavior, search functionality and system requirements. In *Proceedings of the Third Symposium on Information Interaction in Context* (New York, NY, USA, 2010), IIiX '10, ACM, pp. 13–24.

[54] Jones, Rosie, Rey, Benjamin, Madani, Omid, and Greiner, Wiley. Generating query substitutions. In *Proceedings of the 15th International Conference on World Wide Web* (New York, NY, USA, 2006), WWW '06, ACM, pp. 387–396.

[55] Kando, Noriko. Overview of the sixth NTCIR workshop. In *Proceedings of NTCIR-6 Workshop Meeting* (May 2007), NTCIR-6, pp. 1 – 9.

[56] Kharitonov, Eugene, Macdonald, Craig, Serdyukov, Pavel, and Ounis, Iadh. User model-based metrics for offline query suggestion evaluation. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval* (New York, NY, USA, 2013), SIGIR '13, ACM, pp. 633–642.

[57] Kim, Youngho, and Croft, W. Bruce. Diversifying query suggestions based on query documents. In *Submission* (2014).

[58] Kim, Youngho, Seo, Jangwon, and Croft, W. Bruce. Automatic Boolean query suggestion for professional search. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval* (New York, NY, USA, 2011), SIGIR '11, ACM, pp. 825–834.

[59] Kim, Youngho, Seo, Jangwon, Croft, W. Bruce, and Smith, David A. Automatic suggestion of phrasal-concept queries for literature search. *Information Processing & Management* (2014), (in press).

[60] Krovetz, Robert. Viewing morphology as an inference process. In *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (New York, NY, USA, 1993), SIGIR '93, ACM, pp. 191–202.

[61] Kumaran, Giridhar, and Allan, James. Effective and efficient user interaction for long queries. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (New York, NY, USA, 2008), SIGIR '08, ACM, pp. 11–18.

[62] Kumaran, Giridhar, and Carvalho, Vitor R. Reducing long queries using query quality predictors. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval* (New York, NY, USA, 2009), SIGIR '09, ACM, pp. 564–571.

[63] Lavrenko, Victor, and Croft, W. Bruce. Relevance based language models. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (New York, NY, USA, 2001), SIGIR '01, ACM, pp. 120 – 127.

[64] Lavrenko, Victor, and Croft, W.Bruce. Relevance models in information retrieval. *Language Modeling for Information Retrieval 13* (2003), 11 – 56.

[65] Lawrie, Dawn, Croft, W. Bruce, and Rosenberg, Arnold. Finding topic words for hierarchical summarization. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (New York, NY, USA, 2001), SIGIR '01, ACM, pp. 349–357.

[66] Lawrie, Dawn J., and Croft, W. Bruce. Generating hierarchical summaries for web searches. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval* (New York, NY, USA, 2003), SIGIR '03, ACM, pp. 457–458.

[67] Lee, Chia-Jung, and Croft, W. Bruce. Generating queries from user-selected text. In *Proceedings of the 4th Information Interaction in Context Symposium* (New York, NY, USA, 2012), IIIX '12, ACM, pp. 100 – 109.

[68] Lopez, Patrice, and Romary, Laurent. Patatras: Retrieval model combination and regression models for prior art search. In *Proceedings of the 10th Cross-language Evaluation Forum Conference on Multilingual Information Access Evaluation: Text Retrieval Experiments* (Berlin, Heidelberg, 2009), CLEF'09, Springer-Verlag, pp. 430–437.

[69] Lopez, Patrice, and Romary, Laurent. Patatras: Retrieval model combination and regression models for prior art search. *Multilingual Information Access Evaluation I. Text Retrieval Experiments 6241* (2010), 430–437.

[70] Luo, Gang, Tang, Chunqiang, Yang, Hao, and Wei, Xing. Medsearch: A specialized search engine for medical information retrieval. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management* (New York, NY, USA, 2008), CIKM '08, ACM, pp. 143–152.

[71] Lupu, Mihai, Piroi, Florina, jimmy Huang, Zhu, Jianhan, and Tait, John. Overview of the TREC 2009 chemical IR track. In *Proceedings of the Eighteenth Text Retrieval Conference* (2009), TREC-18.

[72] Lv, Yuanhua, and Zhai, ChengXiang. A comparative study of methods for estimating query language models with pseudo feedback. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management* (New York, NY, USA, 2009), CIKM '09, ACM, pp. 1895–1898.

[73] Ma, Hao, Lyu, Michael R., and King, Irwin. Diversifying query suggestion results. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence* (2010), AAAI '10, AAAI Press.

[74] Magdy, Walid, and Jones, Gareth J.F. Pres: A score metric for evaluating recall-oriented information retrieval applications. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval* (New York, NY, USA, 2010), SIGIR '10, ACM, pp. 611–618.

[75] Mahdabi, Parvaz, Andersson, Linda, Keikha, Mostafa, and Crestani, Fabio. Automatic refinement of patent queries using concept importance predictors. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval* (New York, NY, USA, 2012), SIGIR '12, ACM, pp. 505–514.

[76] Mahdabi, Parvaz, Gerani, Shima, Huang, Jimmy Xiangji, and Crestani, Fabio. Leveraging conceptual lexicon: Query disambiguation using proximity information for patent retrieval. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval* (New York, NY, USA, 2013), SIGIR '13, ACM, pp. 113–122.

[77] Mase, Hisao, Matsubayashi, Tadataka, Ogawa, Yuichi, Iwayama, Makoto, and Oshio, Tadaaki. Proposal of two-stage patent retrieval method considering the claim structure. *ACM Transactions on Asian Language Information Processing (TALIP) 4*, 2 (June 2005), 190–206.

[78] Mei, Qiaozhu, Zhou, Dengyong, and Church, Kenneth. Query suggestion using hitting time. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management* (New York, NY, USA, 2008), CIKM '08, ACM, pp. 469–478.

[79] Metzler, Donald, and Croft, W. Bruce. A Markov random field model for term dependencies. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (New York, NY, USA, 2005), SIGIR '05, ACM, pp. 472–479.

[80] Metzler, Donald, and Croft, W. Bruce. Latent concept expansion using Markov random fields. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (New York, NY, USA, 2007), SIGIR '07, ACM, pp. 311–318.

[81] Metzler, Donald, Strohman, Trevor, Turtle, Howard, and Croft, W. Bruce. Indri at TREC 2004: Terabyte track. In *Proceedings of the 13th Text Retrieval Conference* (2004), TREC-13.

[82] Mitra, Mandar, Singhal, Amit, and Buckley, Chris. Improving automatic query expansion. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (New York, NY, USA, 1998), SIGIR '98, ACM, pp. 206–214.

[83] Ozertem, Umut, Velipasaoglu, Emre, and Lai, Larry. Suggestion set utility maximization using session logs. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management* (New York, NY, USA, 2011), CIKM '11, ACM, pp. 105–114.

[84] Ponte, Jay M., and Croft, W. Bruce. A language modeling approach to information retrieval. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (New York, NY, USA, 1998), SIGIR '98, ACM, pp. 275–281.

[85] Qiu, Yonggang, and Frei, Hans-Peter. Concept based query expansion. In *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (New York, NY, USA, 1993), SIGIR '93, ACM, pp. 160–169.

[86] Radlinski, Filip, Szummer, Martin, and Craswell, Nick. Inferring query intent from reformulations and clicks. In *Proceedings of the 19th International Conference on World Wide Web* (New York, NY, USA, 2010), WWW '10, ACM, pp. 1171–1172.

[87] Rafiei, Davood, Bharat, Krishna, and Shukla, Anand. Diversifying web search results. In *Proceedings of the 19th International Conference on World Wide Web* (New York, NY, USA, 2010), WWW '10, ACM, pp. 781–790.

[88] Ritchie, Anna, Teufel, Simone, and Robertson, Stephen. Creating a test collection for citation-based IR experiments. In *Proceedings of the Main Conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics* (Stroudsburg, PA, USA, 2006), HLT-NAACL '06, Association for Computational Linguistics, pp. 391–398.

[89] Robertson, Stephen, and Zaragoza, Hugo. The probabilistic relevance framework: BM25 and beyond. *Found. Trends Inf. Retr. 3*, 4 (Apr. 2009), 333–389.

[90] Rocchio, J. J. Relevance feedback in information retrieval. In *The Smart retrieval system - experiments in automatic document processing* (1971), G. Salton, Ed., Englewood Cliffs, NJ: Prentice-Hall, pp. 313–323.

[91] Russell, Stuart J., Norvig, Peter, Candy, John F., Malik, Jitendra M., and Edwards, Douglas D. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1996.

[92] Santos, Rodrygo L.T., Macdonald, Craig, and Ounis, Iadh. Exploiting query reformulations for web search result diversification. In *Proceedings of the 19th International Conference on World Wide Web* (New York, NY, USA, 2010), WWW '10, ACM, pp. 881–890.

[93] Smucker, Mark D., and Allan, James. Find-similar: Similarity browsing as a search tool. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (New York, NY, USA, 2006), SIGIR '06, ACM, pp. 461–468.

[94] Song, Yang, Zhou, Dengyong, and He, Li-wei. Post-ranking query suggestion by diversifying search results. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval* (New York, NY, USA, 2011), SIGIR '11, ACM, pp. 815–824.

[95] Song, Yang, Zhou, Dengyong, and He, Li-wei. Query suggestion by constructing term-transition graphs. In *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining* (New York, NY, USA, 2012), WSDM '12, ACM, pp. 353–362.

[96] Strohman, Trevor, Croft, W. Bruce, and Jensen, David. Recommending citations for academic papers. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (New York, NY, USA, 2007), SIGIR '07, ACM, pp. 705–706.

[97] Strohman, Trevor, Metzler, Donald, Turtle, Howard, and Croft, W. Bruce. Indri: A language-model based search engine for complex queries (extended version). Tech. rep., CIIR, University of Massachusetts Amherst, 2005.

[98] Tseng, Yuen-Hsien, and Wu, Yi-Jen. A study of search tactics for patentability search: a case study on patent engineers. In *Proceedings of the 1st ACM workshop on Patent information retrieval* (2008), ACM, pp. 33–36.

[99] Turtle, Howard. Natural language vs. Boolean query evaluation: A comparison of retrieval performance. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (New York, NY, USA, 1994), SIGIR '94, Springer-Verlag New York, Inc., pp. 212–220.

[100] Turtle, Howard, and Croft, W. Bruce. Evaluation of an inference network-based retrieval model. *ACM Trans. Inf. Syst. 9*, 3 (July 1991), 187–222.

[101] von Luxburg, Ulrike. A tutorial on spectral clustering. *Statistics and Computing 17*, 4 (2007), 395–416.

[102] Wang, Jun, and Zhu, Jianhan. Portfolio theory of information retrieval. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval* (New York, NY, USA, 2009), SIGIR '09, ACM, pp. 115–122.

[103] Weng, Linkai, Li, Zhiwei, Cai, Rui, Zhang, Yaoxue, Zhou, Yuezhi, Yang, Laurence T., and Zhang, Lei. Query by document via a decomposition-based two-level retrieval approach. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval* (New York, NY, USA, 2011), SIGIR '11, ACM, pp. 505–514.

[104] Xu, Jinxi, and Croft, W. Bruce. Query expansion using local and global document analysis. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (New York, NY, USA, 1996), SIGIR '96, ACM, pp. 4–11.

[105] Xu, Jun, and Li, Hang. Adarank: A boosting algorithm for information retrieval. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (New York, NY, USA, 2007), SIGIR '07, ACM, pp. 391–398.

[106] Xue, Xiaobing, and Croft, W. Bruce. Automatic query generation for patent search. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management* (New York, NY, USA, 2009), CIKM '09, ACM, pp. 2037–2040.

[107] Xue, Xiaobing, and Croft, W. Bruce. Transforming patents into prior-art queries. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval* (New York, NY, USA, 2009), SIGIR '09, ACM, pp. 808–809.

[108] Yang, Yin, Bansal, Nilesh, Dakka, Wisam, Ipeirotis, Panagiotis, Koudas, Nick, and Papadias, Dimitris. Query by document. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining* (New York, NY, USA, 2009), WSDM '09, ACM, pp. 34–43.

[109] Zhai, ChengXiang, Cohen, William W., and Lafferty, John. Beyond independent relevance: Methods and evaluation metrics for subtopic retrieval. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval* (New York, NY, USA, 2003), SIGIR '03, ACM, pp. 10–17.

[110] Zhai, ChengXiang, and Lafferty, John. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (New York, NY, USA, 2001), SIGIR '01, ACM, pp. 334–342.

[111] Zhang, Junte, and Kamps, Jaap. Search log analysis of user stereotypes, information seeking behavior, and contextual evaluation. In *Proceedings of the Third Symposium on Information Interaction in Context* (New York, NY, USA, 2010), IIiX'10, ACM, pp. 245 – 254.

[112] Zhao, Le, and Callan, Jamie. Automatic term mismatch diagnosis for selective query expansion. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval* (New York, NY, USA, 2012), SIGIR '12, ACM, pp. 515–524.

[113] Zhao, Ying, Scholer, Falk, and Tsegay, Yohannes. Effective pre-retrieval query performance prediction using similarity and variability evidence. In *Proceedings of the IR Research, 30th European Conference on Advances in Information Retrieval* (2008), ECIR'08, Springer-Verlag, pp. 52–64.

[114] Zhu, Xiaojin, and Ghahramani, Zoubin. Learning from labeled and unlabeled data with label propagation. Tech. rep., CMU-CALD-02-107, Carnegie Mellon University, 2002.