Foundations and Trends® in Information Retrieval
Vol. 9, No. 3-4 (2015) 209–354
© 2015 M. Gupta and M. Bendersky
DOI: 10.1561/1500000050

Information Retrieval with Verbose Queries

Manish Gupta
Microsoft
gmanish@microsoft.com

Michael Bendersky
Google
bemike@google.com

Contents

Abstract

Recently, the focus of many novel search applications has shifted from short keyword queries to verbose natural language queries. Examples include question answering systems and dialogue systems, voice search on mobile devices and entity search engines like Facebook's Graph Search or Google's Knowledge Graph. However the performance of textbook information retrieval techniques for such verbose queries is not as good as that for their shorter counterparts. Thus, effective handling of verbose queries has become a critical factor for adoption of information retrieval techniques in this new breed of search applications.

Over the past decade, the information retrieval community has deeply explored the problem of transforming natural language verbose queries using operations like reduction, weighting, expansion, reformulation and segmentation into more effective structural representations. However, thus far, there was not a coherent and organized survey on this topic. In this survey, we aim to put together various research pieces of the puzzle, provide a comprehensive and structured overview of various proposed methods, and also list various application scenarios where effective verbose query processing can make a significant difference.

M. Gupta and M. Bendersky. *Information Retrieval with Verbose Queries.* Foundations and Trends® in Information Retrieval, vol. 9, no. 3-4, pp. 209–354, 2015.

DOI: 10.1561/1500000050.

Preface

Information retrieval with verbose natural language queries has gained a lot of interest in recent years both from the research community and the industry. Search with verbose queries is one of the key challenges for many of the current most advanced search platforms, including question answering systems (Watson or Wolfram Alpha), mobile personal assistants (Siri, Cortana and Google Now), and entity-based search engines (Facebook Graph Search or Knowledge Graph). Therefore, we believe that this survey is very timely and should be interesting to readers from both academia as well as industry.

Scope of the Survey

We cover an exhaustive list of techniques to handle verbose queries. Intuitively verbose queries are long. Also empirical observations show that often times long queries are verbose in nature. We use the terms “verbose” queries and “long” queries interchangeably in this survey.

In order to stay focused, following is a list of related topics that we do not cover as part of this survey.

- Automatic Speech Recognition (ASR)
- Processing null queries other than verbose queries

- Methods (e.g., [Yang et al., 2009] and [Tsagkias et al., 2011]) and applications (e.g., [Yih et al., 2006]) which consider documents as queries
- Query processing tasks for short queries which do not need any non-trivial modification to be applicable to long queries
- Community-based question-answering systems

Development of the Survey

Many tutorials and surveys dedicated to general query handling or query log analysis have been conducted by researchers in information retrieval and web mining. However, all of them focus on short queries; none of these have explicitly focused on long verbose queries. This survey is based on a full-day tutorial offered by the authors at the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2015). The slides for the tutorial can be obtained from `http://research.microsoft.com/pubs/241895/gupta15_verbose.pptx`.

This survey is entirely based on previously published research and publicly available datasets, rather than the internal practices of the respective employers of the authors. As such, it should prove useful for both practitioners and academic researchers interested in reproducing the reported results.

Audience

Researchers in the field of information retrieval will benefit the most, as this survey will give them an exhaustive overview of the research in the direction of handling verbose web queries. We believe that the survey will give the newcomers a complete picture of the current work, introduce important research topics in this field, and inspire them to learn more. Practitioners and people from the industry will clearly benefit from the discussions both from the methods perspective, as well as from the point of view of applications where such mechanisms are starting to be applied.

After reading the survey, the audience will be able to appreciate and understand the following.

- What are the interesting properties of complex natural language verbose queries
- Challenges in effective information retrieval with verbose queries
- State-of-the-art techniques for verbose query transformations that yield better expected search performance
- State-of-the-art ranking methods for verbose queries, including supervised learning-to-rank methods
- What user/industry segments can be affected by better retrieval with verbose queries and what are the possible applications

Writing Style

We have tried to make the survey as self-contained as possible. However, for some sections, we have deliberately adopted a reference paper writing style, to enable a holistic overview of the research field. In such cases, we discuss those pieces of work from a more general and abstract standpoint, and advise the readers to go through the referenced papers for details. We provide a basic introduction to preliminary information retrieval concepts, graphical models and dependency parsing in the Appendices.

1

Introduction

Web search has matured significantly in the past two decades. Beyond the ten blue links, search engines display a large amount of heterogeneous information including direct factual answers, task panes, image answers, news answers, video answers, social results, related searches, etc. Broadly, queries to a search engine can be divided into two parts: head and tail. Head queries are the highly popular queries while the tail queries occur with a low frequency in the query log. Although the head queries are handled very elegantly by the popular search engines, there is a large room for improvement when handling the tail queries, a part of which return no results.

1.1 Null Queries

Null queries are queries for which the search engine returns zero results. This could be because of the following reasons.

- Query verbosity
- Mismatch between the searcher and the publisher vocabulary

- Unavailability of relevant documents (temporally, or general rarity)
- Inability of the naïve users to formulate appropriate queries

In this survey, we focus on the verbosity aspect of such "null" or difficult to handle queries. We use the terms "verbose" queries and "long" queries interchangeably. This work focuses on verbose queries as well as on long queries which may or may not be verbose.

1.2 Verbose Queries are Frequent

As shown in Figure 2.1, the percentage of the total query traffic follows a power law distribution with respect to the query length [Arampatzis and Kamps, 2008, Bailey et al., 2010], i.e., for a query Q,

$$p(|Q|) = C|Q|^{-s},\ for\ |Q| \geq k_0 \tag{1.1}$$

where $|Q|$ is the query length in words, C is a normalizing constant, s is the slope, k_0 is the lower bound from which the power law holds.

We consider queries with five or more words as verbose or long queries. In 2006, Yahoo! claimed that 17% of the queries contained five or more words.[1]. Figure 2.1 shows that ∼15% queries contain five or more words.

Popular usage of speech-based personal assistants like Cortana, Siri, and Google Now attract an even higher percentage of verbose queries. Crestani and Du [2006] and Yi and Maghoul [2011] analyzed the properties of written versus spoken queries which were manually generated by participants to satisfy TREC topic information needs. They found that while written queries had an average length of 9.54 and 7.48 words with and without stop words respectively, spoken queries had an average length of 23.07 and 14.33 words respectively. Voice queries were considerably longer than the typed mobile queries.

While most of the verbose queries are explicitly asked by the users, some of them are implicit. Users ask verbose queries explicitly in a large

[1]`http://www.zdnet.com/blog/micro-markets/yahoo-searches-more-sophisticated-and-specific/27`

number of scenarios. Advanced users searching for an exhaustive list of relevant documents in medical literature or patent documents often use verbose comprehensive queries. Naïve users like children or the elderly are not trained to ask short queries to search engines and hence end up using full sentence queries. Community-based question answering platforms also attract long queries. Sometimes users end up using long queries implicitly. Long queries could be an outcome of cut-and-paste behavior. For example, a user just found some text on some topic (say a news headline) and fires it as a query to find related news articles. Similarly, to find a relevant image for a paragraph in a textbook, one may fire the entire paragraph as a query to the search engine. We discuss both the implicit and explicit examples of verbose queries in more details in §9.

1.3 Search Engine Performance for Verbose Queries

Past research in information retrieval found that long queries increase the retrieval performance. However, for web search queries, many researchers have observed that search engines perform poorly on verbose queries. The reasons for poor performance are as follows.

- High degree of query specificity. To satisfy their specific (or narrow) needs, users put additional non-redundant information in verbose queries. But since there are not many web-pages to satisfy such highly specific information needs, it is difficult for search engines to surface the right results.

- Term redundancy or extraneous terms (lot of noise). Often times, verbose queries contain a lot of noise, such as extraneous terms that users believe are important to conveying their information needs, but in fact are confusing to automatic systems.

- Rarity of verbose queries. Most search engines optimize for highly popular (or head) queries. Since verbose queries are rare, search engine algorithms are not tweaked to always perform well for them.

- Lack of sufficient natural language parsing. Longer queries can be answered more effectively if the semantics can be understood using natural language understanding techniques. However, search engines currently do not perform such deep parsing because (a) they are optimized for short queries for which deep natural language parsing is not required, and (b) such deep parsing has performance implications.

- Difficulty in distinguishing between the key and complementary concepts. A verbose query can have multiple concepts. The performance can be improved if the results that contain key concepts are shown at the top. However, identifying key concepts from a verbose query is challenging.

Hence, a large number of efforts have been made to understand such long queries in a more effective manner.

1.4 Datasets

Most of the papers in this area have used the TREC datasets for evaluating their approaches. ROBUST04, W10g, GOV2, ClueWeb-09-Cat-B, TREC123, and CERC are the most popular TREC[2] datasets. ROBUST04 is a Newswire collection, while W10g, GOV2 and ClueWeb-09-Cat-B are web collections. TREC123 is a collection of documents from TREC disks 1 and 2. CERC is the CSIRO Enterprise Research Collection (CERC), a crawl of *.csiro.au (public) web sites conducted in March 2007 and used in the 2007 edition of the TREC Enterprise track. Table 1.1 gives a summary of the dataset statistics. Each of these datasets contain relevance judgments for multiple topics (or queries). The judgments are for multiple documents and are binary or graded (e.g., non-relevant, relevant, highly relevant). TREC topics illustrate the difference between a keyword query and a description query. A TREC topic consists of several parts, each of which corresponds to a certain aspect of the topic. In the example at Figure 1.1, we consider the title (denoted ⟨title⟩) as a keyword query on the topic, and the de-

[2] `http://trec.nist.gov`

Collection	Content	#Docs	Topics
Robust04	Newswire	528155	250
W10g	Web	1692096	100
GOV2	Web	25205179	150
ClueWeb-09-Cat-B	Web	50220423	150
TREC123	TREC disks 1 and 2	742611	150
CERC	Enterprise Documents from *.csiro.au	370715	50

Table 1.1: Statistics for TREC Datasets

scription of the topic (denoted ⟨desc⟩) as a natural language description of the information request. In general, the description field is intended to model what a searcher might first say to someone who will actually help them with their search. The verbose description is therefore often used as the verbose query. Another popular similar dataset is the NTCIR-4/5 English-English ad-hoc IR tasks dataset with an average length of 14 query words for description queries.

Some of the recent papers have also used real web query logs [Balasubramanian et al., 2010, Parikh et al., 2013, Yang et al., 2014]. A few researchers have also used document paragraphs or passages as verbose queries [Agrawal et al., 2011, Lee and Croft, 2012, Gupta, 2015].

1.5 Metrics

A variety of standard information retrieval metrics have been used to evaluate the methods for verbose query processing. Most of the researchers that use TREC datasets evaluate their methods using Mean Average Precision (MAP), Mean Reciprocal Rank (MRR), and Precision@K measures against the relevance judgments. Researchers using query logs also use Normalized Discounted Cumulative Gain (NDCG) with respect to the original long query as a metric. We provide a short description of these metrics in §A.6.

```
<num> Number 829
<title> Spanish Civil War support
<desc> Provide information on all kinds of material
international support provided to either side in the
Spanish Civil War.
```

Figure 1.1: An Example of ⟨title⟩ and ⟨desc⟩ Parts of a TREC Topic

1.6 Organization of the Survey

In this survey we present an organized summary of efforts towards improved information retrieval for verbose queries. We begin with a study of the specific properties of verbose queries (§2) which makes them especially challenging in information retrieval applications. Next, we discuss six main ways of handling long queries – query reduction to a single sub-query, query reduction to multiple sub-queries, query weighting, query expansion, query reformulation, and query segmentation in §3 to §8. Table 1.2 shows examples of each of the techniques.

Long verbose queries can be reduced to a single sub-query which could be, for example, the most important noun phrase in the query (§3). Or the long query could be processed to extract multiple short queries (§4). Rather than reducing queries by dropping terms from long queries, each term could be assigned a weight proportional to its importance (§5). Another way to handle long queries is to add concept words to the original query to make the intent clearer (§6). If the words used in the long queries are very specific, they could be completely reformulated to a new query which could potentially match a larger number of documents (§7). Finally, a verbose query can contain multiple pieces of the user information need. Such a query could be segmented and then each such segment can be reduced, weighted, expanded or reformulated to get desired results (§8). For each of these techniques, we group together related methods and present comparisons of these methods. We put together various domains in which verbose queries are frequent, and also discuss how various verbose query processing techniques have been used to handle them (§9). We conclude this survey with a brief

Technique	Original Query	Modified Query
Query Reduction to a Single Sub-query (§3)	ideas for breakfast menu for a morning staff meeting	breakfast meeting menu ideas
Query Reduction to a Multiple sub-queries (§4)	identify any efforts proposed or undertaken by world governments to seek reduction of iraqs foreign debt	reductions iraqs foreign debt, iraqs foreign debt
Query Weighting (§5)	civil war battle reenactments	civil:0.0889, war:0.2795, battle:0.1310, reenactments:0.5006
Query Expansion (§6)	staining a new deck	staining a new deck Shopping/Home and Garden/Home Improvement
Query Reformulation (§7)	how far is it from Boston to Seattle	distance from Boston to Seattle
Query Segmentation (§8)	new ac adapter and battery charger for hp pavilion notebook	new, ac adapter, and, battery charger, for, hp pavilion notebook

Table 1.2: Examples of Various Techniques for Handling Verbose Queries

Notation	Meaning
$Q = \{q_1, q_2, \ldots, q_n\}$	Original verbose query
P^Q	Power set of Q
P	A sub-query of Q
C	Collection
$\|C\|$	Number of words in C
N	Number of documents in C
$m(P, M)$	Target measure of effectiveness of ranking function M for query P
$tf(q_i)$	Term frequency of q_i in C.
$tf_d(q_i)$	Term frequency of q_i in document or document collection d.
$df(q_i)$	Document frequency of q_i in C.
$T_M(Q)$	Top M relevant documents for query Q.

Table 1.3: Table of Notations

overview of future research directions (§10). Table 1.3 presents a list of frequent notations that we use in this survey.

1.7 Summary

Query verbosity is one of the main reasons for zero results returned by search engines. Verbose queries occur in multiple domains and are increasing with increase in usage of speech-based personal assistants. Currently, search engines perform poorly for such long verbose queries. Hence, a large number of efforts have been made to understand such long queries in a more effective manner. In this survey we present an organized summary of efforts towards improved information retrieval for verbose queries.

Suggested Further Reading: [Arampatzis and Kamps, 2008]: Query length analysis and distribution fitting for multiple datasets; [Crestani and Du, 2006]: Comparison between written and spoken queries in terms of length, duration, part-of-speech, aptitude to describe relevant documents, and retrieval effectiveness; `http://trec.nist.gov/`: Details of the various TREC datasets.

2

Properties of Verbose Queries

In this chapter, we study various properties of verbose queries. We begin with a discussion on the performance of search engines on verbose queries. Verbose queries can be categorized with respect to topics or from a grammatical perspective. We discuss the distribution of verbose queries across these categories. Finally, we discuss properties related to the representation of verbose queries in the entire query log traffic with respect to top k coverage and repetition.

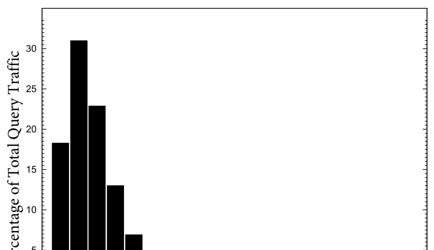


Figure 2.1: Power Law Distribution of Query Length vs Frequency [Bailey et al., 2010]

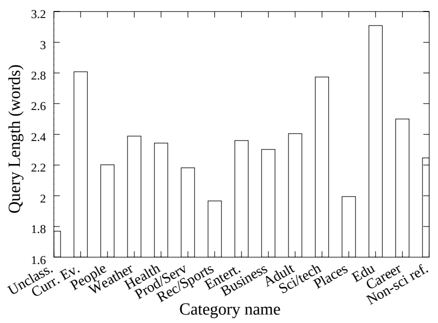


Figure 2.2: Query Length for different Categories [Lau and Horvitz, 1999]

	ROBUST04	W10g	GOV2
title	25.28	19.31	29.67
desc	24.50	18.62	25.27

Table 2.1: Retrieval Effectiveness Comparison (Mean Average Precision) for Title and Desc Queries on several TREC Collections

2.1 Performance for Verbose Queries

2.1.1 Retrieval Effectiveness for TREC Description Queries

Bendersky and Croft [2008] compared the retrieval effectiveness for short queries versus long queries. For short queries, they used the title part of the TREC datasets, while for the long ones they used the description part. The comparison was done using hundreds of queries from three TREC datasets. Table 2.1 shows that title queries consistently perform better on a variety of TREC collections.

2.1.2 Click Data Analysis

As shown in Figure 2.3, Bendersky and Croft [2009] observed that search engine performance drops significantly with increase in the query length. They found a 29% decrease in the expected reciprocal rank of the first click between the shortest (length = 1) and the longest (length = 12) queries in the MSN query log. Also, Figure 2.4 shows that the average click position is closer to the top position for short (SH) queries but much lower down for all types of verbose queries. They also observed that the abandonment rate for users (i.e., the fraction of queries with no clicks) has a high correlation with the length of the query. The verbose queries had an average abandonment rate of 0.6, while short queries had an abandonment rate of just 0.4. Singh et al. [2012] studied search trails of various eBay users and the impact of null queries[1] on purchase rate. They observed a degradation of purchase rate for null search trails compared to the non-null search trails. They found

[1]Singh et al. [2012] found that null queries are on average long in nature and hence we believe that indirectly the observations should hold for verbose queries too.

that the purchase rate for both classes of users (power users as well as novices) is lowered when null recall situations are encountered on their trails.

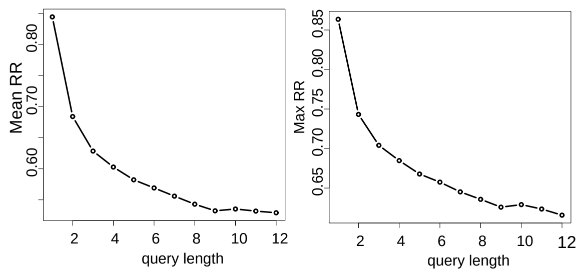


Figure 2.3: Mean and Max Reciprocal Rank by Query Length [Bendersky and Croft, 2009]

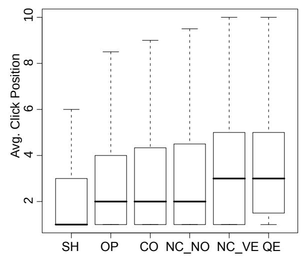


Figure 2.4: Click Positions Distribution by Query Type [Bendersky and Croft, 2009]

2.1.3 Smoothing and Tokenization

Zhai and Lafferty [2001, 2004], Cummins et al. [2015] studied the smoothing aspect for verbose queries. Zhai and Lafferty [2001] found that irrespective of the smoothing method, the performance of longer queries is much more sensitive to the choice of the smoothing parameters than that of short queries. Verbose queries generally require more aggressive smoothing compared to short queries. While Zhai and Lafferty [2004] found that Dirichlet prior method is good for short queries, but Jelinek-Mercer is better for longer queries; [Cummins et al., 2015] found that Dirichlet prior method is the best way of smoothing even for longer queries. Further, Zhai and Lafferty [2004] found that a two-stage smoothing approach is better where first the document language model is smoothed using Dirichlet prior method and then further using Jelinek Mercer. Jiang and Zhai [2007] found that stemming improves performance more for verbose queries compared to short queries. Also, they found that replacing special characters (e.g., hyphens, slashes and brackets) with spaces is the best way of tokenizing verbose queries. In addition, Smucker and Allan [2006] found that augmenting smoothing with inverse collection frequency weighting can improve performance of retrieval with verbose queries by more than 6%.

2.2 Categories of Verbose Queries

2.2.1 Query Category vs Length

Figure 2.2 shows the distribution of query length for different query categories. There is a huge variation in the query length with respect to the query category. On average, queries are much longer for the education, current events and science and technology categories. We suspect that this is because the users in these categories are usually “researchers” and hence tend to put in more keywords.

2.2.2 Types of Verbose Queries

Using a one-month MSN search query log with ∼15M queries, Bendersky and Croft [2009] studied categorization of verbose queries from a different perspective. They classified verbose queries into the following five types.

- Questions: Queries that begin with (what, who, where, when).
- Operators: Queries that contain at least one Boolean operator (AND, OR, NOT), one phrase operator (+, words within double quotation marks), or one special web search operator (contains:, filetype:, intitle:, etc.).
- Composite: Queries which are a composition of short query segments.
- Noun phrases: Non-composite queries that contain a noun phrase but not a verb phrase.
- Verb Phrases: Non-composite queries that contain a verb phrase.

They observed that 7.5% of the verbose queries are questions (QE), 5.5% are operators (OP), 64% are composite (CO), 14.7% are noun

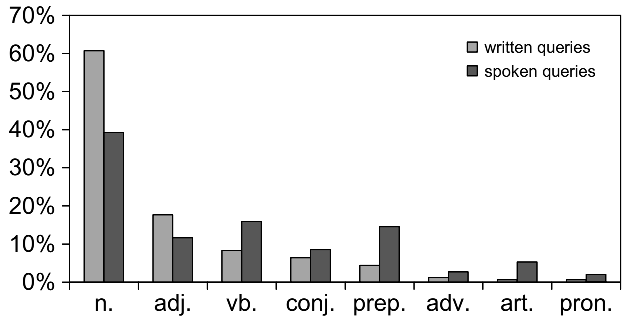


Figure 2.5: Percentages of Part-of-Speech Tags in Written and Spoken Queries [Crestani and Du, 2006]

phrases (NC_NO), and 8.3% are verb phrases (NC_VE). Thus a majority of verbose queries are composite in nature signifying the use of query segmentation as an important step in handling verbose queries.

2.2.3 Part-of-speech Distribution

Figure 2.5 shows the part-of-speech distribution for written and spoken queries. Usually nouns are the most important information conveying words in queries. Spoken queries contained far less number of nouns compared to the written queries. Some popular prepositions (like 'in' and 'at') appeared twice as frequently in the voice sample data set as in the written samples.

2.3 Query Log Traffic Representation

2.3.1 Repetition Factor

Singh et al. [2012] studied the repetitiveness of queries across time on an e-commerce query log from eBay. They found the repetition factor to be as low as 1.45 for null queries versus 19.57 for non-null queries. Even the most popular null queries do not repeat more than tens of thousands of times within a month. But the most popular non-null query repeats more than millions of times. Figure 2.6 shows percent of null queries that repeated x days in the past. This analysis shows that ∼30% queries repeated within a month in the past. Also, it was observed that any two separate days have only 7% null queries in common. A low repetition factor makes it difficult to use query log-based signals to improve the performance of null queries.

2.3.2 Top k Coverage

As shown in Figure 2.7, only 30% of null query traffic is covered by 10% of the most popular null queries. But 90% of non-null query traffic is covered by 10% most popular queries.

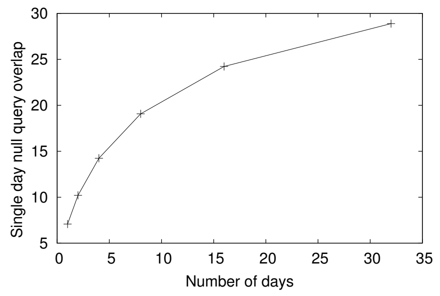


Figure 2.6: Single Day Null Query Overlap with Historical Queries [Singh et al., 2012]

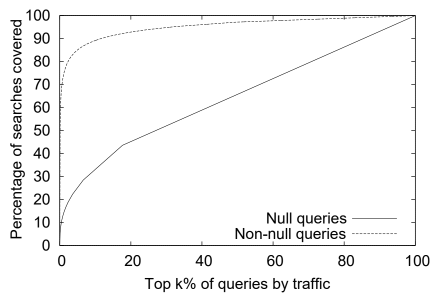


Figure 2.7: Percentage of Searches Covered by Top k% Queries [Singh et al., 2012]

2.4 Other Properties

2.4.1 Information Need Specificity

Phan et al. [2007] studied the variation of information need specificity of queries versus the query length. Queries were manually labeled as broad or narrow on a 4-point scale. Results in Figure 2.8 show a strong correlation between decreasing query length and increasing broadness or generality of queries with a cross-over at ∼3 words/query across four different datasets.

2.4.2 Effect of User Interfaces

Agapie et al. [2013], Franzen and Karlgren [2000] found that the user interface has a significant impact on the length of the query entered

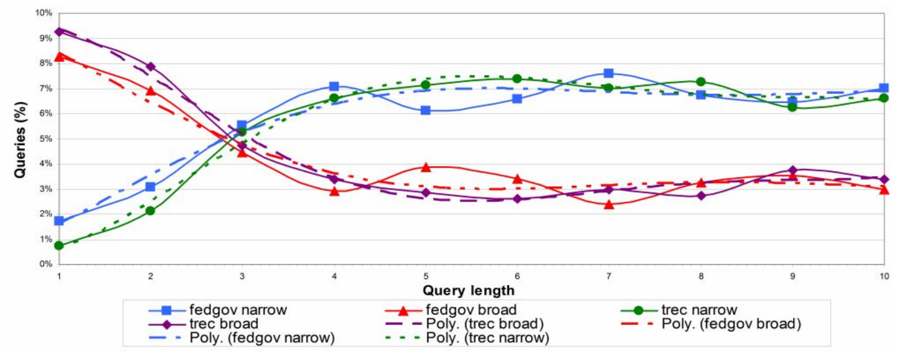


Figure 2.8: Information Need Specificity with respect to Query Length [Phan et al., 2007]

by the user. Franzen and Karlgren [2000] found that a longer query field motivates users to enter longer queries. Agapie et al. [2013] found that a halo with color change around the query field also leads people to writing long queries. Their color change design was as follows. To begin with, an empty query box has a pink halo. As the person starts to type, red hue starts to fade. As the query gets longer, the halo becomes progressively bluer. Long queries are displayed with a blue halo.

2.5 Summary

In this chapter, we studied various properties of verbose queries.

- Short queries consistently show a better performance compared to longer queries as measured both in terms of mean average precision for TREC datasets, as well as in terms of mean reciprocal rank for MSN query logs.

- Dirichlet prior has been found to be the best smoothing method. Also, stemming improves performance for verbose queries.

- A majority of verbose queries are a composition of short query segments signifying the use of query segmentation as an important step in handling verbose queries.

- POS distribution analysis of written versus spoken queries shows that spoken verbose queries contain a lesser number of nouns and more prepositions than written ones.
- Verbose queries do not repeat very often. Unlike short queries, most popular verbose queries do not cover majority of the traffic by verbose queries.
- Verbose queries usually have a narrow specific intent.
- Certain types of user interfaces can be instrumental in soliciting verbose queries.

Verbose queries show very different properties compared to the short queries. These differences bring in new challenges in handling verbose queries. The low frequency of verbose queries as well as low repetition factor are a major reason for search engines to not fine tune for them and hence perform poorly. In the remaining part of the survey we will discuss various techniques for handling such verbose queries and their applications.

Suggested Further Reading: [Bendersky and Croft, 2008]: Analysis of long queries from MSN search log with respect to length distribution, query types, and click data analysis; [Singh et al., 2012]: Properties of null queries on e-commerce portals. How to rewrite them to improve user experience; [Phan et al., 2007]: Relationship between information need specificity and search query length.

3

Query Reduction to a Single Sub-Query

3.1 Introduction

As discussed in §1 and §2, without any special processing, search engine performance for verbose queries is often poor. In this chapter, we will focus on one way to improve this performance by translating the original verbose query into a language that the search engines "understand" and work well with. Namely, instead of searching with the original verbose query, it will be reduced to a shorter version.

The query reduction problem can be formally defined as follows. Given an arbitrary query $Q = \{q_1, q_2, \ldots, q_n\}$, let P^Q denote the power set of Q. Let M be a ranking function (or a retrieval model) which ranks various documents for any query P from the collection C with $|C|$ words and N documents. Let $m(P, M)$ denote a target measure of effectiveness of ranking function M for the query P. The query reduction problem aims at finding a sub-query $P^* = \arg\max_{P \in P^Q} m(P, M)$. Note that the problem of identifying a sub-query is equivalent to the problem of deciding whether to keep or drop a word from Q, or deciding whether to select or reject a group of words from Q, or ranking various sub-queries in P^Q.

While dropping information from the original query may sound counter-intuitive at first, it is important to note that such practice of reducing the amount of information available to the algorithm is not limited to verbose query processing. Case in point, query reduction can be viewed as an instance of feature subset selection [Kohavi and John, 1997], a technique often used in machine learning. In machine learning and statistical NLP problems, feature subset selection often improves model fit to previously unseen data by enhancing generalization and avoiding overfitting.

Similarly, in retrieval problems, dropping too specific, redundant or obscure terms may significantly increase search engine effectiveness. Search engines often employ multiplicative combination of query term weights (e.g., the query likelihood model – see §A.2 for details) or weakly conjunctive logic [Broder et al., 2003] for document scoring. In the case of verbose queries, this may overly benefit documents with potentially spurious matches and cause topic drift. The query reduction process penalizes such documents, promotes documents that contain the most important query terms and lessens topic drift.

Absolute query reduction (one in which the original query is completely discarded and only the reduced query is used for retrieval) is a very hard problem, since it requires high term selection precision – removing an important query term can substantially hurt performance. The most successful absolute query reduction techniques tackled the problem via supervised approaches that combine a multitude of features from the query, documents and search logs (see §3.4 for more details).

In addition to the absolute query reduction, prior work demonstrated that retrieval can be further improved by interpolating the original query with the reduced query (see §3.5). This lessens the risk of completely dropping the important query terms associated with the absolute query reduction. Finally, as another way to mitigate this risk, the researchers also explored user interaction as a way to select the best query reductions (see §3.7).

In addition, in §3.6 we also discuss the efficiency aspects of query reduction methods. Since many of these methods operate over an ex-

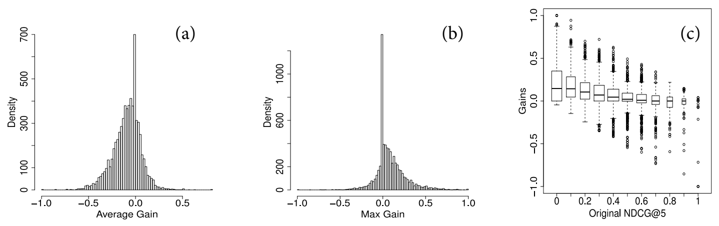


Figure 3.1: Distribution of Potential Gains in NDCG@5 [Balasubramanian et al., 2010]

ponentially large set of sub-queries, it is important to focus on ways to reduce this set, to make the query reduction techniques usable in practical applications.

3.2 Will Query Reduction help?

Empirical analysis by Kumaran and Carvalho [2009] and Balasubramanian et al. [2010] showed that query reduction could be very useful in improving performance for verbose queries. Perfectly reducing long TREC description queries can lead to an average improvement of up to ∼23% in the MAP. Note that perfect reduction is chosen by an oracle, and it means that the best sub-query was selected to replace the original long query. Figure 3.1 shows that (a) on average a randomly chosen sub-query may not perform well; but (b) the best sub-query provides a positive gain over the original verbose query; and (c) gains obtained using best sub-query are higher for verbose queries for which the performance was originally very poor.

Here are a few examples of reducing verbose queries to a single sub-query.

- "ideas for breakfast menu for a morning staff meeting" → "breakfast meeting menu ideas" [Kumaran and Carvalho, 2009]

- "provide information on all kinds of material international support provided to either side in the spanish civil war" → "spanish civil war" [Bendersky and Croft, 2008]

- "define argentina and britain international relations." $\rightarrow$ "britain argentina" [Kumaran and Allan, 2007]

When selecting the best sub-query the following questions need to be answered. What could be the candidates for a sub-query (§3.3)? What could be the features identifying an appropriate sub-query (§3.4)? What are the ways to combine these features in order to find the best sub-query (§3.5)? How can we make the search for the best sub-query efficient (§3.6)? Does asking for user input help (§3.7)? We address these questions in this chapter.

3.3 Candidates for Sub-queries

Various units of Q could be used to extract a sub-query. The following sub-units have been used in the literature.

- Each of the individual words [Allan et al., 1996, Lee et al., 2009b, Park and Croft, 2010, Park et al., 2011]: The simplest way of choosing candidates is to consider each word individually. In this case, one or more top few words could be chosen to form the sub-query.

- All two-word combinations (unordered) [Lee et al., 2009a]: Considering words individually could ignore correlations between importance of words. Hence, two word combinations could also be used as candidates.

- All word subsets [Kumaran and Allan, 2007, Kumaran and Carvalho, 2009, Datta and Varma, 2011, Cummins et al., 2011, Kumaran and Allan, 2008]: To incorporate correlations between importance of multiple words, all subsets of the set of words in the original query could be considered as candidates. However, this leads to a large number of candidates, and hence the approach is very inefficient.

- All word subsets with one word deleted [Balasubramanian et al., 2010, Jones and Fain, 2003, Yang et al., 2014]: This approach assumes that the original query can be improved by removing one specific keyword only. Balasubramanian et al. [2010] observed that dropping just a single (and correct) term from the original long query can result in a 26% improvement in NDCG@5.

- All one to three word queries without stop words [Maxwell and Croft, 2013]: Stop words usually are not important. Hence, it could be better to consider candidates ignoring the stop words.

- Right part of the query [Huston and Croft, 2010]: Many verbose queries contain unimportant words in the beginning of the query. The performance for such queries can be significantly improved by removing the first few unimportant words (also called as the stop structure).

- All noun phrases [Bendersky and Croft, 2008]: Noun phrases have proven to be reliable for key concept discovery and natural language processing, and are flexible enough to naturally distinguish between words, collocations, entities and personal names among others. Hence, noun phrases could act as effective sub-query candidates.

- All named entities [Kumaran and Allan, 2007, Kumaran and Carvalho, 2009]: Named entities (names of persons, places, organizations, dates, etc.) are known to play an important anchor role in many information retrieval applications. Hence, named entities could act as effective sub-query candidates.

- All matching queries from personal query log [Chen and Zhang, 2009]: Query reduction by selecting a random subset of words from the original query may lead to non-meaningful queries. The past short queries may provide us with what specific aspects interest users, and we may discover the users' interesting topics in a given long query. Moreover, since the short queries are created by users, they should contain the significant and meaningful terms for describing the topics.

- Most frequent Part-Of-Speech (POS) blocks extracted from language samples and mapped to the query [Lioma and Ounis, 2008]:

The syntactic arrangement in language can be captured using recurrent arrangements of parts of speech, namely POS blocks. In line with the principle of least effort, one can assume that the most frequently occurring POS blocks in language must be the ones that capture the most information content possible in the least effort-consuming way. This assumption has been known to be valid in the field of linguistics, where complicated and/or difficult syntactic structures and features are used less frequently with time, until they become extinct from language as a whole. It is this exact relation between syntax and information content that can be modeled by considering popular POS blocks as candidates.

3.4 Features to Extract a Single Sub-query

In this section, we provide an exhaustive list of features that have been used for sub-query identification. Note that some features are defined for individual words of the verbose query Q, some for groups of words, and others for sub-queries P.

3.4.1 Statistical Features

These features are mainly based on counts from the query, corpus or external data sources.

- TF-IDF Features [Bendersky and Croft, 2008, Cummins et al., 2011, Kumaran and Carvalho, 2009, Huston and Croft, 2010, Park et al., 2011, Xue et al., 2010, Church and Gale, 1999]: This feature set includes the following features.
  - TF, i.e. the term frequency in corpus
  - IDF, i.e., the inverted document frequency in the corpus
  - Average IDF of query terms, maximum IDF of query terms
  - Sum, standard deviation, max/min, arithmetic mean, geometric mean, harmonic mean, coefficient of variation of IDF of query terms

  - RIDF, i.e., Residual IDF in corpus which is defined as the difference between the observed IDF and the value predicted by a Poisson model.

    $$ridf(P) = idf(P) - \log \frac{1}{1 - e^{-tf(P)/N}} \tag{3.1}$$

    This is based on the assumption that the Poisson distribution only fits the distribution of non-content concepts.

  - TF extracted from Google N-Grams counts [Brants and Franz, 2006]

  - TF in matching Wikipedia titles[1]

  - TF in MSN query logs[2]

  - Count of passages containing the sub-query

- Simplified Clarity Score [Kumaran and Carvalho, 2009, Cummins et al., 2011]: This is the KL divergence between the query model and the collection model. It is the pre-retrieval equivalent of the Query Clarity. It is defined as follows.

  $$SCS(q_i) = \sum_{q_i \in Q} \frac{tf_Q(q_i)}{|Q|} \times \log_2 \frac{\frac{tf_Q(q_i)}{|Q|}}{\frac{tf_C(q_i)}{|C|}} \tag{3.2}$$

  where $tf_Q(q_i)$ and $tf_C(q_i)$ are the number of occurrences of the word q_i in the query Q and collection C respectively.

- Similarity Collection/Query-based (SCQ) Score [Kumaran and Carvalho, 2009, Cummins et al., 2011]: Queries that have higher similarity to the collection as a whole are of higher quality. Hence, the SCQ score feature is defined as follows.

  $$SCQ(q_i) = \left[1 + \log \frac{tf(q_i)}{N}\right] \times \log \left[1 + \frac{N}{df(q_i)}\right] \tag{3.3}$$

[1] `https://dumps.wikimedia.org/`

[2] `http://research.microsoft.com/en-us/um/people/nickcr/wscd09/`

where $tf(q_i)$ is the frequency of q_i in the collection, and $df(q_i)$ is the number of documents in which q_i is present. Given a query, a binary feature could be defined for every word indicating if the word has the maximum SCQ score among all query words.

- Dictionary-based Features [Yang et al., 2014]: Domain-specific dictionaries could be used to define presence features. For example, does the word indicate a brand name, probability of the word occurring in product titles, etc.

- Mutual Information (MI) between Words [Kumaran and Allan, 2007, 2008, Kumaran and Carvalho, 2009, Yang et al., 2014]: This feature is used to rank various sub-queries of a verbose query. For a sub-query P, they first form a graph with words as vertices; edge weight is set to mutual information between terms. Mutual information between two words q_i and q_j is defined as follows.

$$I(q_i, q_j) = \log \left[\frac{n(q_i, q_j)/|C|}{\frac{tf(q_i)}{|C|} \times \frac{tf(q_j)}{|C|}} \right] \tag{3.4}$$

  where $n(q_i, q_j)$ is the number of times words q_i and q_j occurred within a window of 100 words across the collection C. Based on such a graph, the following two features can be derived.

  – Average mutual information between words

  – Weight of the maximum spanning tree

  Also, one can use mutual information between a word and the query category as a feature for each query word.

3.4.2 Linguistic Features

These features are based on the linguistic properties of the queries and also those of individual words themselves.

- POS Tags [Park et al., 2011, Lee et al., 2009a,b, Lioma and Ounis, 2008, Yang et al., 2014, Xue et al., 2010]: Whether the word is a noun, verb, adjective, adverb, conjunction, or numeric; ratio of nouns, adjectives and verbs in a query per query length.

- Named Entities [Balasubramanian et al., 2010, Kumaran and Allan, 2007, Lee et al., 2009a,b, Xue et al., 2010]: Is the query word a person name, a location, an organization, or a time string? Does the query contain a location?
- Combination of POS and Named Entities for a Pair of Words as Candidate [Lee et al., 2009a]: For example, if both words q_i and q_j are nouns (or person names), the feature pos_nn=1 (or ne_pp=1).
- Acronyms [Lee et al., 2009a,b]: Is the query word an acronym?
- Syntactic Features [Park et al., 2011, Xue et al., 2010]: Number of noun phrases in the query, average depth of the key-concept (noun phrases) terms in the parse tree of the query, height of the parse tree for the query.
- Lexical Forms of Neighboring Words [Yang et al., 2014, Lease et al., 2009].

3.4.3 Query Features

These features are based on various properties of the query alone without any other context.

- Query Length [Kumaran and Carvalho, 2009, Park et al., 2011, Balasubramanian et al., 2010]
- Similarity Original Query [Kumaran and Carvalho, 2009]: Cosine similarity between TF-IDF vectors representing each sub-query and the original long query.
- Presence of Stop Words [Park et al., 2011, Balasubramanian et al., 2010]: Ratio of stop words in sub-query.
- Presence of URL [Balasubramanian et al., 2010]

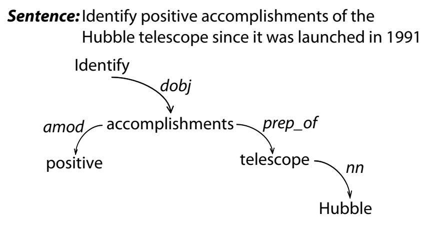

Sentence: Identify positive accomplishments of the Hubble telescope since it was launched in 1991
Identify
dobj
amod
accomplishments
prep_of
positive
telescope
nn
Hubble

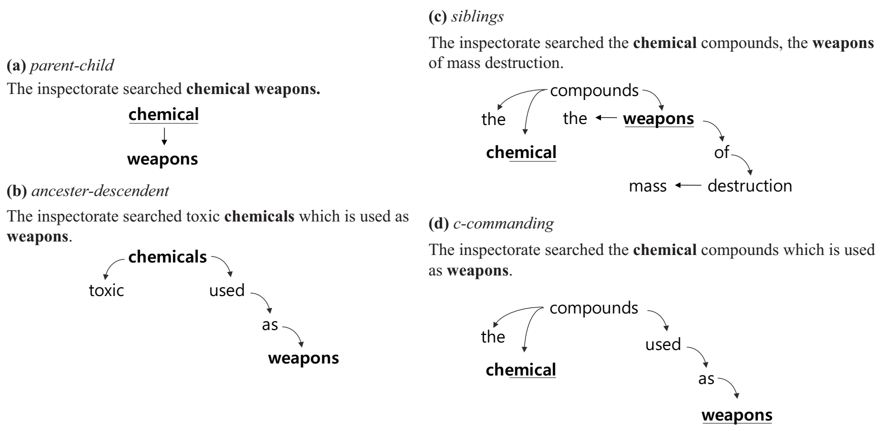
Figure 3.2: An Example of Dependency Parse Trees [Park and Croft, 2010]

- IsRightMost [Yang et al., 2014]: Users tend to put a modifier word as the rightmost word in the query.
- IsLeftMost [Yang et al., 2014]: Users tend to put optional adjectives on the left and key noun phrases on the right.
- Is the query a wh-question? [Park et al., 2011]
- Is the query a question? [Park et al., 2011]
- Category of the query [Yang et al., 2014]
- Location of the word in the query [Lease et al., 2009]
- Is the word trailed by a comma? [Lease et al., 2009]

3.4.4 Word Dependency Features

These features capture the dependencies between query words. We provide a basic introduction to dependency parsing in §C.

- Binary Dependencies [Park and Croft, 2010]: Dependency parse trees help to illustrate the binary dependencies existing between various query words as shown in Figure 3.2. Based on these, three types of syntactic features can be defined as follows.
  - An original syntactic feature with the specific word and the particular dependency



(a) parent-child
The inspectorate searched chemical weapons.
chemical
weapons
(b) ancester-descendent
The inspectorate searched toxic chemicals which is used as weapons.
chemicals
toxic
used
as
weapons
(c) siblings
The inspectorate searched the chemical compounds, the weapons of mass destruction.
compounds
the
chemical
the
weapons
of
mass
destruction
(d) c-commanding
The inspectorate searched the chemical compounds which is used as weapons.
compounds
the
chemical
used
as
weapons

Figure 3.3: Four Types of Dependencies [Park et al., 2011]

- Feature with the particular dependency and any word (i.e., the word is generalized to a '*')
- Feature with the specific word and any type of dependency (i.e., the type of dependency is generalized to a '*')

- Quasi-synchronous Dependencies [Park et al., 2011]: These features capture the dependencies as shown in Figure 3.3. Features include the number of dependent clauses in the query; and the ratio of the dependent term pairs which have parent-child, ancestor-dependent, siblings and c-commanding relations in the query. We provide details about dependency parsing in §C.

3.4.5 Query Log-based Features

These features are computed using a query log in context along with the current query.

- Query log frequency [Bendersky and Croft, 2008]: the number of times q_i was used as part of a query, and the number of times q_i was used as an exact query.

- Similarity with Past Queries [Chen and Zhang, 2009]: This can be defined in two ways as follows.

  - Count of common terms between the long query Q and sub-query P.

    $$R(Q, P) = \sum_{i=1}^{s} \frac{|S_i \cap P|}{|P|} \tag{3.5}$$

    where S_i is the i^{th} sentence in Q.

  - Number of common noun phrases between Q and noun phrases from sentences around sub-query terms in clicked result page.

    $$R(LQF, STF) = \frac{2|LQF \cap STF|}{|LQF| + |STF|} \tag{3.6}$$

    where LQF and STF are features (or noun phrases) from long query Q and sub-query context respectively.

- Deletion History [Jones and Fain, 2003, Yang et al., 2014]: the number of times word q_i was deleted, the number of times word q_i was deleted/the number of times q_i was seen in the query log for a particular category c, conditional deletion which is P(deleting q_i for similar long query earlier).

- Rareness in Query Log [Yang et al., 2014]: This is defined as the ratio of the number of queries in category c to the number of queries in category c containing q_i.

3.4.6 Post Retrieval Features

These features are computed by observing the results when sub-queries are fired to the search engine. Hence, these are expensive to compute. But they have been proven to be very effective.

- Query-document Relevance Scores [Balasubramanian et al., 2010, Chen and Zhang, 2009]: LambdaRank and BM25 scores of top

K documents (position-wise as well as aggregated as minimum/ maximum/ average/ standard deviation/ variance), click through counts of top K documents, Page-rank scores of top K documents.

- Term/Topic Co-occurrence/Context Features [Lee et al., 2009a,b, Yang et al., 2014]: Features in this group include the following.
  - Term-topic: Co-occurrence of word q_i and $\{Q - q_i\}$
  - Term-term: Co-occurrence of word q_i and $q_j \in \{Q - q_i\}$ where $i \neq j$
  - Term-term context: Cosine similarity between context vectors of q_i and $q_j \in \{Q - q_i\}$ where $i \neq j$
  - Term-topic context: Cosine similarity between context vectors of q_i and $\{Q - q_i\}$

Here the context vector for q_i is defined as a list $\langle$docID, relevance score$\rangle$. Recall that N is the total number of documents. Let a be the number of documents with q_i and q_j, b be the number of documents with q_i but not q_j, c be the number of documents with q_j but not q_i, and d be the number of documents with neither q_i nor q_j. Co-occurrence for two words q_i and q_j can be measured using three different measures as follows.

- Point-wise mutual information

$$PMI(q_i, q_j) = \log \frac{aN}{(a+b)(a+c)} \tag{3.7}$$

- Chi square statistic

$$\chi^2(q_i, q_j) = \frac{N(ad - bc)^2}{(a+b)(a+c)(b+d)(b+c)} \tag{3.8}$$

– Log likelihood ratio

$$\begin{aligned} & -2\log \ LLR(q_i, q_j) \\ = \quad & a \ \log \frac{aN}{(a+b)(a+c)} + b \ \log \frac{bN}{(a+b)(b+d)} \\ + \quad & c \ \log \frac{cN}{(c+d)(a+c)} + d \ \log \frac{dN}{(c+d)(b+d)} \end{aligned} \tag{3.9}$$

- Word Co-occurrence in Pseudo-relevant Documents [Maxwell and Croft, 2013].

- Query Scope [Kumaran and Carvalho, 2009]: Query Scope of a sub-query P is the size of the retrieved document set relative to the size of the collection. We can expect that high values of query scope are predictive of poor-quality queries as they retrieve far too many documents.

$$QS(P) = -\log \frac{N_P}{N} \tag{3.10}$$

where N_P is the number of documents containing at least one query term.

- Weighted Information Gain [Bendersky and Croft, 2008]: Change in information about the quality of the retrieval (in response to P) from a state where only the average document is retrieved to a state where the actual results are observed.

$$wig(P) = \frac{\frac{1}{M} \sum_{d \in T_M(P)} \log(p(P|d)) - \log(p(P|C))}{-\log(p(P|C))} \tag{3.11}$$

where $T_M(P)$ is the set of top M documents retrieved in response to P from collection C, and $p(P|\cdot)$ is the query likelihood probability calculated using the maximum likelihood estimate (MLE).

- Query Clarity [Kumaran and Carvalho, 2009, Cummins et al., 2011]: It is defined as the KL divergence of the query model from the collection model and has been found to be better than the IDF and the MI features.

$$QC(P) = \sum_{q_i \in P} \frac{tf_{T_M(P)}(q_i)}{|P|} \times \log_2 \frac{\frac{tf_{T_M(P)}(q_i)}{|P|}}{\frac{tf_C(q_i)}{|C|}} \tag{3.12}$$

where $tf_{T_M(P)}(q_i)$ and $tf_C(q_i)$ are the number of occurrences of the word q_i in the top M documents retrieved in response to the query P and the number of occurrences of the word q_i in the collection C respectively.

- Query Drift among Results [Cummins et al., 2011]: A set of features can be used to incorporate the query results drift signal as follows.
  - Standard deviation of the relevance scores at 100 documents
  - A normalized version of standard deviation at 100 documents: Standard deviation normalized by relevance score of corpus (treat corpus as a big document) for the query.
  - The maximum standard deviation in the ranked-list
  - Standard deviation using a variable cut-off point where cut-off is defined by the document whose score is at least 50% of the score of the top ranked document.
  - Query length normalized standard deviation using a variable cut-off point.

3.5 Methods to Combine the Features for Query Reduction

In this section, we discuss in detail a few methods to combine the various features that we discussed in the previous section.

3.5.1 Classification/Regression

A large number of research efforts have been made towards combining the features using a classification or a regression model. Typically the classification problem is to classify whether a word should be included in the sub-query or discarded. Typically the regression problem is to learn a weight for each word denoting its importance score or to learn a weight for a sub-query; the top few words or the sub-query with highest weight is then chosen. RankSVM [Balasubramanian et al., 2010, Kumaran and Carvalho, 2009, Park and Croft, 2010], decision trees, AdaBoost, logistic regression [Yang et al., 2014] are some of the popular classification methods. Random forests [Balasubramanian et al., 2010] is the most popular regression model. Besides this, Lee et al. [2009a] proposed a multi-level regression for individual words and also for pairs of words. Lee et al. [2009b] studied both generation and reduction approaches based on classification and regression. The generation approach greedily selects "effective" terms until k terms are chosen while the reduction approach greedily removes "in-effective" terms until k terms have been removed. k is an empirical value given by the users. They found the generation approaches to be better than the reduction approaches.

3.5.2 Core Term Identification using Heuristic Rules

The earliest work [Allan et al., 1996] in the area of query reduction was performed using simple heuristic rules. Allan et al. [1996] proposed the following method to identify the most important sub-query from a long query.

- Discard query words that are present in the stop word list.
- Weigh words (weight=ω) based on rules like (a) "purpose clause" is the most important part of query, (b) for queries of the form "What is the X of Y?" and "How X is Y?", Y is more important than X, etc.
- Rank words by $\omega \times avg\_tf^{0.7} \times idf$ where avg_tf is the term's average frequency in documents where it occurs. The top ranked term is called as the core term.
- If the core term is part of some phrase, the phrase becomes the core term.
- Cluster the query terms. If the cluster with the core term has multiple terms, replace the core term with a proximity operator (window) containing all terms in the cluster.

- If the term with highest ω is not in the core term, add it to the core term.
- If the proximity operator matches $<$10 documents, relax it by discarding the terms with low weights until $\geq$10 documents are matched.

3.5.3 Clustering and Rules-based Approach

This method [Chen and Zhang, 2009] uses clustering to perform segmentation of the query and then chooses the sub-query from among those clusters. The detailed method is as follows.

- Retrieve several short queries P related to a long query Q from the user's query history. Relatedness is measured in terms of common words between the sub-query and the long query as follows.
$$R(Q, P) = \sum_{i=1}^{n} \frac{|S_i \cap P|}{|P|} \tag{3.13}$$
where S_i is the i^{th} sentence in the long query Q.
- Filter the non-relevant results by comparing contexts from search results with contexts from the original long query. Context for the long query is expressed in terms of the noun phrases in the long query. Similarly, context for the sub-query is expressed in terms of the noun phrases extracted from sentences around the sub-query terms in the clicked result page. Comparison is performed using the following measure.
$$R(LQF, STF) = \frac{2|LQF \cap STF|}{|LQF| + |STF|} \tag{3.14}$$
where LQF and STF are features (or noun phrases) from the long query Q and sub-query context respectively.
- Construct sub-query clusters. Clustering is performed using an iterative method on a bipartite graph constructed using noun phrases (features) from the long query in one layer and the sub-queries as nodes in the second layer. Two noun phrases are clustered together if they are contained in many sub-queries. Two

sub-queries are clustered together if they contain many common noun phrases from the long query.

- Select the most representative sub-query to substitute the long query based on this score.

$$S(P) = \frac{1}{n}\sum_{i=1}^{|P|}\left[\frac{num_j(P_i)}{\sum\limits_k num_j(P_k)} \times \log\frac{|c|}{|\{c : P_i \in c\}|}\right] \tag{3.15}$$

where $num_j(P_i)$ is the number of occurrences of term P_i in the cluster c_j, $|c|$ is the number of clusters, and $|\{c : P_i \in c\}|$ is the number of clusters containing P_i.

3.5.4 Key Concept (KC) Discovery

Given a long query Q, the following modified ranking principle can be followed to rank documents. This is a linear combination of the traditional matching combined with the concept (or a sub-query)-based matching.

$$rank(d) \propto \lambda p(Q|d) + (1-\lambda)\sum_i p(P_i|Q)p(P_i|d) \tag{3.16}$$

where P_i corresponds to a sub-query/concept (or a noun phrase) and can be a key concept (KC) or not. Bendersky and Croft [2008] use AdaBoost algorithm with many statistical features to estimate $h_k(P_i)$, i.e., the confidence that $P_i \in KC$. Then an estimate of $p(P_i|Q)$ is computed as follows.

$$p(P_i|Q) = \frac{h_k(P_i)}{\sum\limits_{P_i \in Q} h_k(P_i)} \tag{3.17}$$

For a query, only one or two of the most key concepts are chosen. They observed that their approach is better than the Sequential Dependency Model [Metzler and Croft, 2005] (SD, described in §5) and much better than using the verbose query Q directly. Note that Park and Croft [2010] studied the binary dependency features and found their model to be better than the KC model.

3.5.5 Query Reduction Using Stop Structures

Many verbose queries contain unimportant words in the beginning of the query. The performance for such queries can be significantly improved by removing the first few unimportant words (also called as the stop structure). For example, “My husband would like to know more about cancer” can be reduced to “cancer”, “if i am having a lipid test can i drink black coffee” can be reduced to “lipid test can i drink black coffee”. Stop structure identification can be performed using sequential taggers like CRF++ and Yamcha. Huston and Croft [2010] found that Yamcha provided better accuracy compared to CRF++. They also observed that removing stop structures could improve the performance for verbose queries from Yahoo! API and Bing API datasets almost 1.5–2 times in terms of NDCG@5 and NDCG@10.

3.5.6 Query Reduction using Random Walk

Query reduction can also be performed by using linkage-based importance of query words in a word-word co-occurrence graph (PhRank). Given a query $Q = \{q_1, \dots, q_n\}$, let $T_M(Q)$ be the set of top M relevant documents (pseudo-relevant document set). Maxwell and Croft [2013] build a co-occurrence graph of word stems such that adjacent stems in $T_M(Q)$ are linked. Edge weights are computed as follows.

$$l_{ij} = r_{ij} \times \sum_{d_k \in T_M(Q)} p(d_k|Q)(\lambda c_{ijw_2} + (1-\lambda)c_{ijw_{10}}) \tag{3.18}$$

where c_{ijw_2} is the number of stem co-occurrences within window of size 2, and r_{ij} is computed as follows.

$$r_{ij} = \log_2 \frac{\sum\limits_{ij \in T_M(Q)} c_{ijw_2}}{1 + c_{ijw_2}} \tag{3.19}$$

Next, word scores (π_{q_i}) are computed using random walk on this graph. Each vertex score is weighed by exhaustiveness and global salience in the collection as follows.

$$\pi_{q_i} = \pi_{q_i} \times f_{avg}(q_i) \times idf(q_i) \tag{3.20}$$

where $f_{avg}(q_i)$ is the frequency of q_i averaged over all documents in $T_M(Q)$ and normalized by the maximum average frequency of any term in $T_M(Q)$; $idf(q_i) = \log_2 \frac{|C|}{1+df(q_i)}$ where $|C|$ is the size of the collection vocabulary. Candidate terms (or reduced query candidates) are all combinations of 1–3 words in a query that are not stop-words. Term scores are computed using the average word score for words in a term, combined with global discrimination weights.

$$f(x,Q) = z_x \times \sum_{q_i \in x} \frac{\pi_{q_i}}{n} \tag{3.21}$$

where

$$z_x = f_x \times idf(x) \times l_x \tag{3.22}$$

and

$$l_x = |x|^{|x|} \tag{3.23}$$

where $|x|$ is the number of words in term x, f_x is the number of times x appears in a window of size $4|x|$ in C and $idf(x) = \log_2 \frac{|C|}{1+df(x)}$.

PhRank achieves significant performance gains with a small number of compact terms. It outperforms the following baselines by a significant margin in terms of both MAP and R-Precision[3]: Query Likelihood [Ponte and Croft, 1998] (QL, §A.2), Sequential Dependency Model [Metzler and Croft, 2005] (SD, §5.5), Key Concept Discovery [Bendersky and Croft, 2008] (KC, §3.5.4) and Sub-query Distributions [Xue et al., 2010] (§4.2).

3.5.7 Different Formulations to Optimize

The problem of identifying the best sub-query for a given long query can be modeled in many ways. Balasubramanian et al. [2010] study the following three ways. Given a long query Q and one of its reduced versions P, let $h(P)$ be the predicted performance of sub-query P, and $T(P)$ is the true performance of sub-query P measured as the effectiveness of the retrieved results.

[3]R-Precision is precision at cut-off R where R is the number of relevant documents for the query.

- Independent Prediction: This method learns a regressor h^* such that it minimizes the mean squared error.

$$h^* = \arg\min_{h} \sqrt{\sum_{\forall Q_t \in \bar{Q_t}, P \in P_1^{Q_t}} (h(P) - T(P))^2} \tag{3.24}$$

  Then use this model to predict the performance of each sub-query independently and select the sub-query with the highest predicted performance.

$$P^* = \arg\max_{P \in P_1^Q} h^*(P) \tag{3.25}$$

  Here, $\bar{Q_t}$ is the collection of training verbose queries, Q_t is a particular training verbose query, and P_1^Q is a set of all sub-queries obtained by dropping one word from Q.

- Difference Prediction: This method learns the regressor h_d^* such that it minimizes the mean squared difference between the true and the predicted performance differences as follows.

$$h_d^* = \arg\min_{h_d} \sqrt{\sum_{Q_t \in \bar{Q_t}} \sum_{P \in P_1^{Q_t} \wedge P \neq Q_t} (h_d(Q_t, P) - D(Q_t, P))^2} \tag{3.26}$$

  Then, use this model to predict the difference in performance between each reduced version and its original query, and then select the query that has the highest positive difference.

$$P^* = \arg\max_{P \in P_1^Q} h^*(Q, P) \tag{3.27}$$

  Note that here h_d and D are the predicted and true performance differences.

- Ranking Queries: The goal is to rank the original long query Q and its reduced versions P in order to select the top ranking query. Ranking model h_r^* is learned by training on pairwise preferences between queries.

$$\begin{aligned} h_r^* = \arg\min_{h_r} \sum_{Q_t \in \bar{Q_t}} \sum_{P \in P_1^{Q_t}} & I[sign(h(P) - h(Q_t)) \\ & \neq sign(T(P) - T(Q_t))] \end{aligned} \tag{3.28}$$

where I is the indicator function. Use the model to predict the best sub-query as follows.

$$P^* = \arg\max_{P \in P_1^Q} h_r^*(P) \tag{3.29}$$

They observed that among all these formulations, the Independent formulation with appropriate thresholding provided the best NDCG gain.

3.6 Efficiency Aspect of Query Reduction Methods

In the previous section, we discussed various methods to rank all sub-queries of a long query. However, a verbose query has an exponentially large number of sub-queries and so performing such evaluations across all sub-queries could be computationally intensive for long queries. Hence, in this section we discuss various ways of choosing a limited number of sub-query candidates rather than considering all sub-queries as candidates. Following are the popular ways of making the query reduction process efficient.

- Consider sub-queries with a small fixed length only [Kumaran and Carvalho, 2009], say between 3 and 6 terms.
- Consider only those sub-queries that contain named entities [Kumaran and Carvalho, 2009].
- Consider candidates of a fixed type only like named entities or noun phrases.
- Consider sub-sequences with no gaps only.
- One word deletion [Jones and Fain, 2003, Yang et al., 2014]: Balasubramanian et al. [2010] observed that dropping just a single (and correct) term from the original long query can result in a 26% improvement in NDCG@5.
- Randomly pick up a few sub-queries [Datta and Varma, 2011]: Discard the k^{th} query word with probability=$l_{opt}/|Q|$ where $|Q|$

is the length of the query and l_{opt} is the optimal length of queries obtained using a training query set. This approach was found to be better than [Balasubramanian et al., 2010] with sub-query sample size as small as $3|Q|$.

3.7 Ask for User Input to Guide Query Reduction

Kumaran and Allan [2007, 2008] observed that applying automated ways to reduce long queries using simple mutual information-based statistical features did not give much gains. Hence, they proposed a solution where sub-queries are ranked automatically, followed by seeking user input (Interactive Query Reduction). The interface to seek user input consists of (1) the description (long query) and narrative portion of the TREC topic, and (2) the list of candidate sub-queries along with result snippets. They observed that user input helped and the selected sub-queries showed better performance compared to the original queries.

However, showing too many sub-query options to the user can irritate the user. Hence, they proposed two tricks to reduce the number of options shown to the user.

- Overlapping search results (or set cover pruning): Let X be the union of sets of 10 documents retrieved across options. Find minimal set of options that cover X. It is an NP hard problem and hence they propose a greedy algorithm for the same.

- Identical snippets (or snippet-based pruning): Retain one option from the set of sub-queries that retrieve the same snippet.

It was observed that for Interactive Query Reduction (IQR), an average decrease of two options per query using set cover pruning does not cause any significant drop in performance. In case of Interactive Query Expansion (IQE), an average reduction of six options per query is achieved using set cover pruning without any performance loss. Snippet-based pruning also leads to reduction of one option on average without any performance loss for both IQR and IQE. An inter-leaving of the top results from IQR and IQE (i.e., Selective Interactive Reduction and Expansion) with set cover pruning performed better than either IQR or IQE with significantly less number of options.

3.8 Summary

Reducing the long query to a smartly selected shorter version can provide better performance compared to just using the original query. Various units of the original query could be used to extract a sub-query. A large number of various types of features have been explored for reducing verbose queries to single sub-queries. These include statistical features, linguistic features, query features, word dependency features, query log-based features, and post retrieval features. While most of the papers suggest the use of plain classification or regression models to find the best sub-query, other innovative methods like random walks and stop structures have been found to be effective too. Finally, the efficiency aspect has been addressed by smartly selecting a few candidate sub-queries. Soliciting help from the user by showing a few nearly optimal sub-query options without causing overload seems to be promising.

Suggested Further Reading: [Kumaran and Carvalho, 2009]: Detailed description of statistical and query features for sub-query quality prediction; [Balasubramanian et al., 2010]: Three learning formulations that combine query performance predictors to perform automatic query reduction; [Yang et al., 2014]: Reduction using query term deletion using large-scale e-commerce search logs; [Maxwell and Croft, 2013]: Query reduction using random walks; [Huston and Croft, 2010]: Query reduction using stop structures; [Kumaran and Allan, 2007, 2008]: Efficient and effective user interaction for query reduction.

4

Query Reduction by Choosing Multiple Sub-Queries

4.1 Introduction

In the previous chapter, we examined ways to reduce a verbose query into a single best sub-query. However, instead of selecting the best sub-query to represent a long query, one can take a more general approach and model the query reduction problem as a distribution over the space of all possible sub-queries. All these sub-queries can then be submitted to the search engine and the results obtained from all these queries are merged to obtain the final list of results for the long query.

For example, consider the query "give information on steps to manage control or protect squirrels." Using the techniques mentioned in the previous chapter, one can reduce the query to a single sub-query: "steps manage control protect squirrels." Instead, reducing it to the following sub-query distribution would be more general:

"steps protect squirrels":0.621
"steps control squirrels":0.324
"steps control protect squirrels":0.048
"steps manage squirrels":0.002.

Methods based on such sub-query distributions have been shown to outperform single query reduction methods. This may be accounted to the fact that these methods lessen the risk of completely dropping important query terms as discussed in the previous chapter.

In sub-query distribution methods, the sub-queries are generally modeled as random variables, and their probabilities are assigned according to their expected retrieval performance. In this chapter, we discuss various mechanisms to define such probability distributions over sub-queries, and also how to merge results from multiple sub-queries.

In §4.2 we discuss a method that uses a variant of Conditional Random Fields for probability assignment, while §4.3 discusses a method that uses a list-wise approach, ListNet. Finally in §4.4, we discuss a tree-based method that constructs a sub-query tree based on reformulation operations. In such a way, reformulation trees are able to encode the dependency between the different sub-queries, which yields better performance compared to the previous methods that treat sub-queries as independent random variables.

4.2 Sub-query Distributions using CRF-perf

To learn distributions over sub-queries, the training set consists of $\{Q, \{P, m(P, M)\}\}$ where Q is the original query, P is a sub-query, $m(P, M)$ is the retrieval performance measure value for query P under the retrieval model M. It is also denoted by $m(P)$ when M is clear in the context. The sub-query selection problem is to decide for every word whether to include it in the sub-query or not. Neighboring words generally tend to have the same labels, either keep all of them or drop all of them. Therefore, it is reasonable to model sub-query selection as a sequential labeling problem.

When there is a single label associated with every instance, a CRF (conditional random field) optimizes the labeling accuracy based on the training set of input sequences and their corresponding gold-standard label sequences. We provide a basic introduction to CRFs in §B.2. To address the case of multiple labels each with a certain weight, CRF-perf was proposed by Xue et al. [2010]. CRF-perf directly optimizes the expected retrieval performance over all the sub-queries.

For a typical CRF, the distribution over sub-queries is given by the following equation.

$$P(P|Q) = \frac{exp\left[\sum\limits_{k=1}^{K} \lambda_k f_k(Q,P)\right]}{Z(Q)} \tag{4.1}$$

where the partition function $Z(Q)$ is computed as follows.

$$Z(Q) = \sum_{P \in P^Q} exp\left[\sum_{k=1}^{K} \lambda_k f_k(Q,P)\right] \tag{4.2}$$

Here, f_k are the feature functions, and λ_k is the weight of the k^{th} feature.

For CRF-perf, the distribution over sub-queries is given by the following equation.

$$P_m(P|Q) = \frac{exp\left[\sum\limits_{k=1}^{K} \lambda_k f_k(Q,P)\right] m(P)}{Z_m(Q)} \tag{4.3}$$

where the partition function $Z(Q)$ is computed as follows.

$$Z_m(Q) = \sum_{P \in P^Q} exp\left[\sum_{k=1}^{K} \lambda_k f_k(Q,P)\right] m(P) \tag{4.4}$$

A large number of features as described in §3.4 were used as f_k to train the CRF. Average Precision (AP) was the optimized retrieval performance measure m. Xue et al. [2010] experimented with the following four retrieval models M to compute the relevance score for a document D.

- SubQL

$$score_{QL}(D,P) = \sum_{p_i \in P} \log(P(p_i|D)) \tag{4.5}$$

where $P(p_i|D)$ is estimated using the language modeling approach with Dirichlet smoothing (§A.1 and §A.2).

- SubDM

$$score_{DM}(D,P) = \lambda_T \sum_{p_i \in P} \log(P(p_i|D))$$
$$+ \lambda_O \sum_{o \in O(P)} \log(P(o|D)) + \lambda_U \sum_{u \in U(P)} \log(P(u|D)) \quad (4.6)$$

  where $O(P)$ denotes a set of ordered bigrams extracted from P, and $U(P)$ denotes a set of unordered bigrams extracted from P. Usually λ_T, λ_O, and λ_U are set to 0.85, 0.1 and 0.05 respectively. This is the same as the SD model (§5.5).

- QL+SubQL

$$score(D,Q,P) = \alpha score_{QL}(D,Q) + (1-\alpha) score_{QL}(D,P) \quad (4.7)$$

  where α is a constant.

- DM+SubQL

$$score(D,Q,P) = \alpha score_{DM}(D,Q) + (1-\alpha) score_{QL}(D,P) \quad (4.8)$$

  where α is a constant.

The retrieval models above are used to compute $m(P,M)$ for each sub-query P which is useful for training the CRF. Once the parameters of the CRF are learned, given a new query Q, a distribution $P(P|Q)$ is computed using the CRF. The following two strategies can be used to compute the final results for Q.

- Top 1: Select the sub-query with the highest CRF probability and feed it to the retrieval model M.

- Top K: Select top k sub-queries, feed them to the retrieval model M and compute the combined score as follows.

$$score_{QL}(D,\{P\}) = \sum_{i=1}^{k} P(P_i|Q) score_{QL}(D,P_i) \quad (4.9)$$

  where P_i is the i^{th} sub-query.

Xue et al. [2010] observed that the sub-query distributions method performs better than all the four baselines: QL [Ponte and Croft, 1998] (§A.2), SD [Metzler and Croft, 2005] (§5.5), SRank [Kumaran and Carvalho, 2009], and KC [Bendersky and Croft, 2008] (§3.5.4). Note that SRank considers sub-query selection as a ranking problem and uses Rank SVM as the ranking model and features as discussed in §3.4. Among all the methods for computing sub-query distributions they observed that DM+SubQL(K) performed best at K=10.

4.3 Sub-query Distributions using ListNet

Xue and Croft [2011] took a different approach to estimate $P(P|Q)$. They assume that it is a linear combination of a variety of query features. To learn the combination parameter for each query feature, they generate the corresponding retrieval feature by calculating the sum of the retrieval scores of using all sub-queries weighted by their query feature values. They use the ListNet [Cao et al., 2007] Learning to Rank approach to learn combination parameters of these generated retrieval features. ListNet is a listwise ranking algorithm. Unlike pairwise approaches such as Rank SVM and RankBoost using pairs of objects in learning, lists of objects are used as instances in the listwise approch. ListNet employs cross entropy loss as the listwise loss function in gradient descent.

Again they implement the subset query distributions using either QL or SD as the retrieval model. The two models are thus called QDist-QL and QDist-DM respectively.

Xue and Croft [2011] found that QDist-DM performed better than QDist-QL as well as previously proposed QL+SubQL and DM+SubQL methods [Xue et al., 2010].

4.4 Reformulation Trees Method

Xue and Croft [2012] proposed a more sophisticated way of handling multiple sub-queries of a verbose query. It organizes these multiple sub-queries in the form of a tree called the reformulation tree. A reformulation tree (as shown in Figure 4.1(c)) organizes multiple sequences of

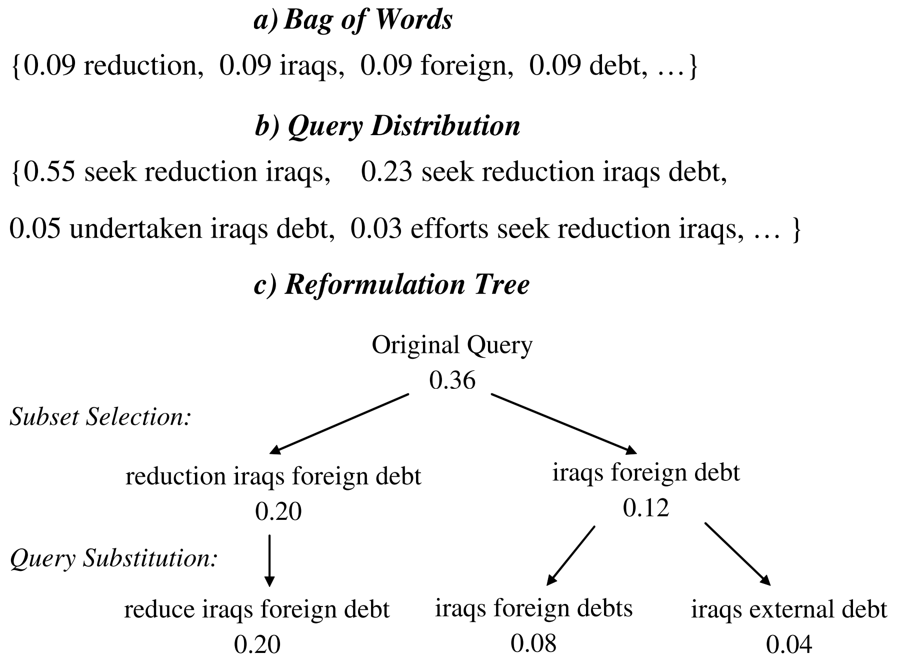

a) Bag of Words
{0.09 reduction, 0.09 iraqs, 0.09 foreign, 0.09 debt, …}
b) Query Distribution
{0.55 seek reduction iraqs, 0.23 seek reduction iraqs debt,
0.05 undertaken iraqs debt, 0.03 efforts seek reduction iraqs, … }
c) Reformulation Tree
Original Query 0.36
Subset Selection:
reduction iraqs foreign debt 0.20
iraqs foreign debt 0.12
Query Substitution:
reduce iraqs foreign debt 0.20
iraqs foreign debts 0.08
iraqs external debt 0.04

Figure 4.1: Different Query Representations for a Verbose Query "identify any efforts proposed or undertaken by world governments to seek reduction of iraqs foreign debt" [Xue and Croft, 2012]

reformulated queries as a tree structure, where each node is a reformulated query and each path of the tree corresponds to a sequence of reformulated queries. A two-level reformulation tree combines two query operations, i.e., sub-query selection and query substitution, within the same framework. A weight estimation approach assigns weights to each node of the reformulation tree by considering the relationships with other nodes and directly optimizing the retrieval performance. Reformulation trees are richer than sub-query distributions as the latter do not consider relationships between the nodes. When the tree T is used for retrieval, the score of document D is computed as follows.

$$sc(T,D) = \sum_{q_r \in T} w(q_r) sc(q_r, D) \tag{4.10}$$

where $w(q_r)$ is the weight of node corresponding to the reformulated query q_r and is computed as follows.

$$w(q_r) = w(parent(q_r)) \sum_{k} \lambda_k f_k(q_r) \tag{4.11}$$

The tree is constructed as follows. Given the original long query, remove stop words, select top 10 words by IDF, and generate subsets of length 3 to 6. Select $SubNum$ sub-queries as nodes in the tree. Next, $ModNum$ nodes at first level need to be substituted. Substitution can then be performed using one of the following three ways.

- Morph: Morphologically similar words
- Pattern: Words matching patterns extracted from original query
- Wiki: Wikipedia redirect pairs

Weight assignment is then done as follows. $w(Q)$ is set to 1 for the original query Q. Weights for the sub-queries are computed using ListNet by aggregating features across various sub-queries as follows.

$$w(q_{sub}) = \sum_{k} \lambda_k^{sub} f_k^{sub}(q_{sub}) \tag{4.12}$$

where the λs are estimated using features which are defined as follows.

$$F_k^{sub}(\{q_{sub}\}, D) = \sum_{q_{sub}} f_k^{sub}(q_{sub}) sc(q_{sub}, D) \tag{4.13}$$

Further, weights for substituted queries are computed as follows.

$$w(q_{mod}) = w(q_{sub}) \sum_k \lambda_k^{mod} f_k^{mod}(q_{mod}) \tag{4.14}$$

where the λs are estimated using features which are defined as follows.

$$F_k^{mod}(\{q_{mod}\}, D) = \sum_{q_{mod}} w(q_{sub}) f_k^{mod}(q_{mod}) sc(q_{mod}, D) \tag{4.15}$$

SD [Metzler and Croft, 2005] (§5.5) is used to compute $sc(q_r, D)$, $sc(q_{sub}, D)$ and $sc(q_{mod}, D)$.

Reformulation trees were found to provide significantly better performance in terms of MAP, Precision@10 and NDCG@10 compared to QL (§A.2), SD (§5.5), KC (§3.5.4), $QL + SubQL$ (§4.2, Eq. 4.7) as well as $DM + SubQL$ (§4.2, Eq. 4.8).

4.5 Summary

A more general way of reducing verbose queries is to reduce it to multiple sub-queries and assign a weight to each of the these reduced queries. We discussed three main methods to obtain such sub-query distributions: CRF-perf, ListNet and Reformulation Trees. The CRF-perf method proposed learning sub-query distributions by directly optimizing the expected retrieval performance over all sub-queries using an adaptation of the traditional CRF model. The ListNet method assumes that the probability of each sub-query can be learned as a linear combination of a variety of query features where the combination parameters are learned using ListNet. In the reformulation trees also, weight of the sub-queries is learned using ListNet by aggregating features across various sub-queries. However, the sub-queries are organized in a tree fashion and the method also incorporates the notion of query substitution followed by sub-query extraction. All of these sub-query distributions methods have been found to perform better than the methods which reduce the verbose query to a single sub-query. Reformulation trees is the most general framework amongst these. Both ListNet and Reformulation Trees have been found to perform better compared to the CRF-perf method.

Suggested Further Reading: [Xue et al., 2010]: Good motivation for reduction to multiple sub-queries, and the CRF-perf-based sub-query distributions method; [Xue and Croft, 2011]: ListNet-based method to model sub-query distributions; [Xue and Croft, 2012]: Two-level reformulation tree that effectively combines two query operations, i.e., subset selection and query substitution, within the same framework.

5

Weighting Query Words and Query Concepts

5.1 Introduction

Given a verbose query, we discussed (1) selecting a single sub-query, and (2) selecting multiple weighted sub-queries in Chapters 3 and 4 respectively. In this chapter, we will discuss techniques that go beyond making a binary decision (include / exclude) per query word, and instead assign weights to individual query words or concepts.

Term weighting approaches that were used in prior information retrieval research were targeted mostly at keyword queries and employed either simple collection statistics or pseudo-relevance feedback for term weighting. Both of these approaches have disadvantages when applied in the context of retrieval with verbose queries.

Term collection statistics, such as commonly used inverse document frequency weighting, ignore the context of the current query. For instance, consider the two queries:

- Q_1="Term limitations for US Congress members"
- Q_2="Insurance Coverage which pays for Long Term Care"

The word "term" has much more importance in query Q_1 rather than in query Q_2. In Q_1 assigning a low weight to the word "term" would clearly not produce desired search results as it is a central word in the query. However in Q_2, assigning a low weight to the word "term" would still be able to produce desired results. However, inverse document frequency-based weighting will provide the same weight to the word "term" in both cases, which will hurt the retrieval performance of query Q_1. Such scenarios are especially common for verbose search queries, where term weights tend to be highly dependent on their surrounding context. Hence, we start this chapter by discussing various methods of assigning weights to words which can provide better overall retrieval performance for verbose queries. Usually, the document language model (θ_D) and the query language model (θ_Q) are simply computed using the word frequency-based Maximum Likelihood Estimate (MLE) approach combined with smoothing using the corpus. Using various methods as described in §5.2, §5.3 and §5.4, the query model can be estimated more accurately leading to better matching.

Standard pseudo-relevance feedback based weighting may also adversely affect the performance of verbose queries. Since, as demonstrated in the previous chapters, verbose queries' performance is often inferior to short keyword queries, using pseudo-relevance feedback can cause topic drift in the retrieved result set, which in turn leads to incorrect term weight assignment. In this chapter, we describe methods that overcome the query drift via better concept weighting at the initial retrieval stage (§5.9) or via retrieval using multiple corpora (§5.10).

Finally, much of the prior work on retrieval with keyword queries treated individual words independently, which can further hurt performance of verbose queries due to spurious word matches. Instead, in this chapter we describe methods that assign weights to word dependencies or "concepts". These concepts may be defined syntactically (§5.7) or statistically (§5.8).

In §5.11 we further extend the discussion to concept dependencies (or dependencies between word dependencies). Such concept dependencies can yield very rich and expressive query representations. For instance, Q_1 in the example above can be modeled via the concepts "term limitations", "US congress" and "members", with further dependency between these concepts that will give preference to documents where these concepts co-occur within a close proximity.

Word and concept weighting, expansion and dependencies provide powerful and flexible mechanisms for dealing with verbose queries. They can be applied in a variety of retrieval scenarios, including, among others, web search [Bendersky et al., 2010], image search [Nie et al., 2012] and medical information retrieval [Choi et al., 2014] and have been shown to provide state-of-the-art retrieval performance.

5.2 A Fixed-Point Method

Paik and Oard [2014] perform query weighting by drawing on an idea from text summarization, in which centrality is also a key issue. They iteratively estimate which words are most central to the query using power iterations. They use an initial set of retrieval results to define a recursion on the query word weight vector that converges to a fixed point representing the vector that optimally describes the initial result set. Their approach is based on the following two key intuitions.

- Important words are more frequent than the less important words in the segment of the collection where original query words are densely present.
- Importance of a word increases if it is more frequent than other important words.

Let $A(q_i)$ denote the centrality of word q_i. Let D be the initial set of documents retrieved for query Q. Word centrality is then defined as follows.

$$A(q_i) = \sum_{j=1, i \neq j}^{|Q|} \sum_{d \in D} RF(q_i|q_j, d) A(q_j) \tag{5.1}$$

where RF is computed as follows.

$$RF(q_i|q_j, d) = \begin{cases} \frac{\log_2(1+tf_d(q_i))}{\log_2(1+tf_d(q_j))}, & \text{if } tf_d(q_j) > 0 \\ \log_2(1 + tf_d(q_i)), & \text{otherwise} \end{cases}$$

Importance of the word q_i is then computed as a combination of the centrality factor with an IDF factor as follows.

$$I(q_i) = A(q_i).\frac{idf(q_i)}{c + idf(q_i)} \tag{5.2}$$

where c is a constant. The fixed point method was found to be better than QL (§A.2), SD (§5.5), KC (§3.5.4) and WSD (§5.8) models. Also, the computation time was found to be much less than SD, KC and WSD.

5.3 Word Necessity Prediction using Regression

Zhao and Callan [2010] propose that SVD-based similarity could be used to compute intuitive features that can help predict word necessity. Let $S(t, w)$ denote the cosine similarity between words t and w in Singular Value Decomposition (SVD) concept space. We provide a basic introduction to SVD in §A.5. SVD is computed using a TF-IDF weighted word-document matrix where the documents are the top K relevant documents for the query. Let w_i be the words in descending order of similarity to query word t, i.e., $S(t, w_1) \geq S(t, w_2) \geq \ldots$ and so on. Based on the SVD space, they propose the following features.

- If a word is not central to a topic, it is unlikely to be necessary. To capture this intuition, they define $Topic\ Centrality(t) = S(t, w)$ where w is the word with highest $S(t, w)$. In most cases, $t = w$ and $S(t, w)$ indicates the weight of the word preserved after SVD.
- Searchonyms (synonyms, antonyms, hyponyms) may replace the original query word in relevant documents, lowering the necessity of the query word. Hence, they define the Synonymy feature as follows.

$$Synonymy(t) = \sum_{i=2}^{c+1} \frac{S(t, w_i)}{c} \tag{5.3}$$

  where c is a constant.
- If a word does not often co-occur with its synonyms, then it is often replaceable by the synonyms, and the word is less necessary.

Thus, Replaceability is defined as follows.

$$Replaceability(t) = \sum_{i=1...6}^{w_i \neq t} \frac{df(w_i) - n(t, w_i)}{df_i} \times \frac{S(t, w_i)}{S(t, t)} \tag{5.4}$$

where $n(t, w_i)$ is the number of documents in the collection matching both t and w_i.

- Many abstract but rare words are unnecessary, because they are at a different abstraction level than the relevant documents. Rareness is thus captured as a feature.

$$Rareness(t) = idf(t) = \log \frac{N}{df(t)} \tag{5.5}$$

Necessity enhanced Language modeling (LM) and Okapi methods were found to be better than the LM and Okapi models without “necessity”.

5.4 Regression Rank

Lease et al. [2009] propose a regression method to estimate the query language model rather than the usual MLE approach. A key idea of their approach is that one can generalize knowledge of successful query models from past queries to predict effective query models for novel queries. Regression Rank is a framework which uses regression to weigh query words, i.e., estimate more effective query models. Given a dataset of queries with labeled relevant and non-relevant documents, one needs to generate training data such that each instance corresponds to a query word. In order to do this, we must have query models to generalize from (i.e., to train the regression model). Various statistical, linguistic, and query features (as discussed in §3.4) are used to characterize each query word and form the features for each instance. The regression label for each training instance is computed by estimating θ_Q using a weighted combination of frequency-based language models of its subqueries where the weights are defined by query performance. For every training query Q, estimate $\hat{\theta_Q} = \sum_s [Metric(\theta_s)\theta_s]$. The language models θ_s are selected using grid search in the space of all possible query language models. Grid points are query reduction candidates of size up to 6.

Regression Rank was found to be significantly better than query likelihood with maximum likelihood estimates, SD (§5.5) and KC (§3.5.4) models.

5.5 Sequential Dependence (SD) Model using Markov Random Fields

It is well known that dependencies exist between terms in a collection of text. Occurrences of certain pairs of terms are correlated. The fact that either one occurs provides strong evidence that the other is also likely to occur. The SD model [Metzler and Croft, 2005] weighs query words and word pairs by modeling such dependencies between query words using a Markov Random Field (MRF) network. We provide a basic introduction to MRF in §B.1. The network consists of words in a query and a document node. Then, in this model, the query-document match can then be computed using the posterior as follows.

$$P(D|Q) \equiv \sum_{c \in C(G)} \log\ \psi(c) = \sum_{c \in C(G)} \lambda_c f(c) \tag{5.6}$$

where $C(G)$ is the set of cliques in the network G, $\psi(c)$ is a potential function defined over clique c, $f(c)$ is the feature function, λ_c is the weight for feature $f(c)$ and $\equiv$ denotes rank equivalence. Three variants of such a network are shown in Figure 5.1. They are Full Independence, Sequential Dependence, and Full Dependence. The Full Independence variant makes the assumption that query words q_i are independent given some document D. The Sequential Dependence variant assumes a dependence between neighboring query words. The Full Dependence variant assumes all query words are in some way dependent on each other.

Potentials (defined over cliques) can then be grouped into three types as follows.

- Potential function over the query word-document clique

$$\psi_T(c) = \lambda_T \log P(q_i|D) = \lambda_T \log \left[(1-\alpha_D)\frac{tf_{q_i,D}}{|D|} + \alpha_D \frac{cf_{q_i}}{|C|}\right] \tag{5.7}$$

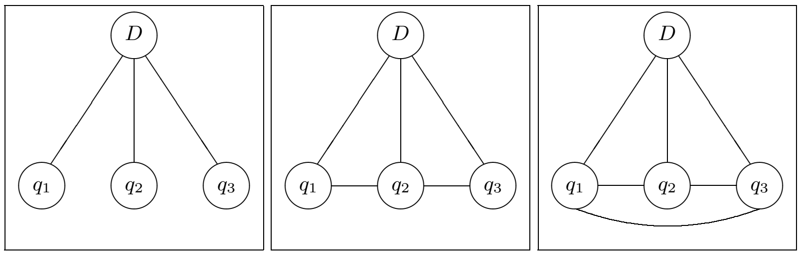


Figure 5.1: Three Variants of the Markov Random Field Network [Metzler and Croft, 2005]

- Ordered potential function over cliques with more than two query words

$$\begin{aligned}\psi_O(c) &= \lambda_O \log P(\#1(q_i, \ldots, q_{i+k}), D)\\ &= \lambda_O \log \left[(1-\alpha_D)\frac{tf_{\#1(q_i,\ldots,q_{i+k}),D}}{|D|} + \alpha_D \frac{cf_{\#1(q_i,\ldots,q_{i+k})}}{|C|}\right] \quad (5.8)\end{aligned}$$

where $tf_{\#1(q_i,\ldots,q_{i+k}),D}$ denotes the number of times the exact phrase $q_i, \ldots, q_{i+k}$ occurs in document D with an analogous definition for $cf_{\#1(q_i,\ldots,q_{i+k})}$.

- Unordered potential function over cliques with more than two query words

$$\begin{aligned}\psi_U(c) &= \lambda_U \log P(\#uwN(q_i, \ldots, q_j), D)\\ &= \lambda_U \log \left[(1-\alpha_D)\frac{tf_{\#uwN(q_i,\ldots,q_j),D}}{|D|}\right.\\ &\left. +\alpha_D \frac{cf_{\#uwN(q_i,\ldots,q_j)}}{|C|}\right] \quad (5.9)\end{aligned}$$

where $tf_{\#uwN(q_i,\ldots,q_j),D}$ is the number of times the words $q_i, \ldots, q_j$ appear ordered or unordered within a window of N words. $cf_{\#uwN(q_i,\ldots,q_j)}$ is defined similarly.

Relevance of the document D for the query Q is then computed as follows.

$$\begin{aligned} P(D|Q) &\equiv \sum_{c \in C(G)} \lambda_c f(c) \\ &= \sum_{c \in T} \lambda_T f_T(c) + \sum_{c \in O} \lambda_O f_O(c) + \sum_{c \in O \cup U} \lambda_U f_U(c) \quad (5.10) \end{aligned}$$

where T, O and U are the set of all words, all ordered bigrams and all unordered bigrams within a window in Q respectively. Note that the parameters are constrained by setting $\lambda_T + \lambda_O + \lambda_U = 1$. Coordinate-level hill climbing algorithm is used to fix λs where the objective is to optimize the MAP. The performance was seen to be the best when the window for the unordered pairs was set to 8 words. The Full Dependent model and Sequential Dependence Model were found to be better than the Full Independent model. The number of cliques in the FD model is exponential in the number of query terms which limits the application of this variant to shorter queries. SD and FD provided similar accuracy in most cases. However, since FD considers a much larger set of dependencies compared to SD, FD takes significantly more execution time. Thus, considering both accuracy and execution time, SD was found to be better than the FI and FD models for verbose queries. The best parameter values for SD were found to be (0.85, 0.10 and 0.05) across the TREC datasets.

5.6 Integrating Regression Rank with Markov Random Fields (MRFs)

Although the weights for each feature class are learned from data in SD [Metzler and Croft, 2005], feature weights within each class are estimated by the same uniform assumption as the standard unigram. That is, the MRF uses the maximum likelihood estimate (MLE)-based Dirichlet-smoothed unigram. Regression Rank provides a metric-optimized θ_Q rather than MLE. Lease [2009] observed that MRF's first term computation could use Regression Rank rather than simple maximum likelihood. Weights for bigrams, λ_O and λ_U, are still computed using the same ML estimates. Such a combination of MRF and Regression Rank framework was shown to be clearly better than either of the two.

5.7 Quasi-synchronous Dependency (QSD) Language Model

The SD model captures word dependencies by allowing differences in order and promixity of words in queries versus the documents. However, it ignores word dependence based on syntactic relationships in queries. Also, if one uses binary dependencies with the head-modifier relation in which two dependent words are directly linked in a dependency structure, long-distance word dependencies can get ignored. Quasi-synchronous word dependence models support multiple syntactic configurations and hence allow flexible inexact query-document matching. Hence, Park et al. [2011] proposed the QSD model which influences the query-document matching process using the word dependence based on quasi-synchronous syntactic relationships in queries. We provide a basic introduction to dependency parsing in §C. We discussed four types of such quasi-synchronous linguistic dependencies in §3.4.

Let T_Q and T_D be the dependency trees for the query and the document respectively. A synchronous model allows dependent pair of (e.g., parent-child) words in the query to be aligned with words having different syntactic configurations in the document tree. Let A be a loose alignment between the query and the document. A is a set of all possible combinations of four syntactic configurations between a query and a document. It has 4×7 elements: parent-child, child-parent, ancestor-descendent, descendent-ancestor, siblings, and two for c-commanding. For example, word 'a' is parent of word 'b' in the query dependency tree could match with word 'a' is a parent of word 'b' in the document or word 'b' is a parent of word 'a' in the document. Using this notion of quasi-synchronous dependencies, the probability that the query is generated by the document model can be written as follows.

$$P(Q|D) = P(T_Q, A|T_D) = P(A|T_D)P(T_Q|A, T_D) \tag{5.11}$$

Let N be the number of elements in A. Let syn_D and syn_Q be one of the four syntactic configurations. Let T_{D,syn_D} represent a set of dependent terms having syn_D dependency and $P(syn_D, syn_Q|T_{D,syn_D})$

be the probability that a syntactic relation syn_D in a document is used in the form of syn_Q in the query. Then we can write the following.

$$\begin{aligned} & P(A|T_D)P(T_Q|A, T_D) \\ & = \sum_{(syn_D, syn_Q) \in A} P(syn_D, syn_Q|T_{D,syn_D})P(T_{Q,syn_Q}|T_{D,syn_D}) \\ & = \sum_{(syn_D, syn_Q) \in A} \frac{1}{N} P(T_{Q,syn_Q}|T_{D,syn_D}) \end{aligned} \tag{5.12}$$

For the first term, we assumed $P(syn_D, syn_Q|T_{D,syn_D})$ to be $\frac{1}{N}$. The second term of the above equation can be computed as follows.

$$P(T_{Q,syn_Q}|T_{D,syn_D}) = \sum_{(q_i,q_j) \in T_{Q,syn_Q}} \lambda(q_i, q_j)P(q_i, q_j|T_{D,syn_D}) \tag{5.13}$$

where q_i and q_j are dependent terms with relation as syn_Q in the parse tree of a query. Here $\lambda(q_i, q_j)$ is the mean value of the query term ranking scores for q_i and q_j as computed by Park and Croft [2010]. Further, the joint probability of a pair of query terms q_i and q_j given the dependency can be computed as follows.

$$P(q_i, q_j|T_{D,syn_D}) = (1 - \alpha_D)\frac{tf_{q_i,q_j,syn_D}}{|D|} + \alpha_D \frac{cf_{q_i,q_j,syn_D}}{|C|} \tag{5.14}$$

where $tf(q_i, q_j, syn_D)$ and $cf(q_i, q_j, syn_D)$ are the frequency of word pairs q_i and q_j with the syntactic relation syn_D in a document D and in the collection C respectively. The model can be combined with the Sequential Dependency Model [Metzler and Croft, 2005] to lead to the following query-document match score.

$$P(Q, D) = \lambda SD(Q, D) + (1 - \lambda)QuasiSync(Q, D) \tag{5.15}$$

where λ is learned using regression such that it maximizes the overall average precision of the interpolated retrieval model. SD model along with the Quasi-Sync interpolation was found to be better than the SD model.

5.8 Weighted Sequential Dependence (WSD) Model

One of the primary limitations of the SD model is the fact that all matches of the same type (e.g., words, ordered window, or unordered window) are treated as being equally important. This is the result of the massive parameter tying that is done in Eq. 5.10. Instead, it would be desirable to weight, a priori, different terms (or bigrams) within the query differently based on query-level evidence. For example, in a verbose query, there will likely be a few concepts (terms or phrases) within the query that will carry the most weight. While the sequential dependence model would treat all of the concepts as equally important, we would like to be able to weight the concepts appropriately, with regard to each other. The WSD model aims to address this drawback [Bendersky et al., 2010]. For example, weighted concepts for the query "civil war battle reenactments" using WSD were "civil:0.0619, war:0.1947, battle:0.0913, reenactments:0.3487, civil war:0.1959, war battle:0.2458, battle reenactments:0.0540".

As an extreme, one way is to have a different λ for every word q_i and $\lambda_{i,i+1}$ for every bigram (q_i, q_{i+1}). But these are too many parameters. Each λ now depends on 1 (or 2) query words. Bendersky et al. [2010] proposed the WSD model which finds a middle path by parameterizing λs as follows.

$$\lambda(q_i) = \sum_{j=1}^{k_u} w_j^u g_j^u(q_i) \tag{5.16}$$

and

$$\lambda(q_i, q_{i+1}) = \sum_{j=1}^{k_b} w_j^b q_j^b(q_i, q_{i+1}) \tag{5.17}$$

where $g^u(q_i)$ and $g^b(q_i, q_{i+1})$ are features defined over unigrams and bigrams respectively. The retrieval model is then expressed as follows.

$$\begin{aligned} P(D|Q) = \sum_{i=1}^{k_u} w_i^t \sum_{q \in Q} g_i^u(q) f_T(q, D) \\ + \sum_{i=1}^{k_b} \sum_{q_j, q_{j+1} \in Q} g_i^b(q_j, q_{j+1}) f_O(q_j, q_{j+1}, D) \\ + \sum_{i=1}^{k_b} w_i^b \sum_{q_j, q_{j+1} \in Q} g_i^b(q_j, q_{j+1}) f_U(q_j, q_{j+1}, D) \end{aligned} \tag{5.18}$$

Here, g^u and g^b are concept importance features: endogenous (collection dependent) and exogenous (dependent on external data sources like Google N-Grams, MSN Query log, Wikipedia titles – see Section 3.4 for more details on these sources). They are document independent and capture importance of a concept within the query. In Eq. 5.18, k_u and k_b are the number of unigram and bigram importance features respectively. Also, w^u and w^b are free parameters estimated so as to optimize for the retrieval metric of interest such as MAP or NDCG. Since the ranking function is linear with respect to w, coordinate level ascent algorithm is used to update each parameter w_i one at a time and iteratively.

On all the three TREC datasets (ROBUST04, W10g and GOV2) and a web corpus dataset, WSD was found to be better than QL (§A.2) and SD (§5.5). Also, WSD using both unigrams and bigrams was better than WSD with only unigrams or WSD with only bigrams for both the TREC and the web datasets. Similarly, WSD with all features was better than WSD with only endogenous features or WSD with only exogenous features on TREC datasets. However, for the web corpus, the exogenous features did not improve the accuracy.

5.9 Parameterized Query Expansion (PQE) Model

Previous work focused on few concept types only: noun phrases [Bendersky and Croft, 2008], terms [Lease, 2009], query term spans [Svore et al., 2010]. WSD extended the SD model to include generic term concepts like bigrams with individual weights. The PQE model, proposed by Bendersky et al. [2011b], focuses both on explicit query concepts as well as on latent concepts that are associated with the query through pseudo-relevance feedback (PRF). We provide a basic introduction to PRF in §A.3. PQE is a unification of the concept weighting and the query expansion models. It considers four concept types as follows.

- QT-concepts: words
- PH-concepts: bigrams

- PR-concepts: bigrams from queries that are within a window of size 8 in documents
- ET-concepts: top-K terms associated with the query through pseudo-relevance feedback

For example, for the query: "What is the current role of the civil air patrol and what training do participants receive?", the following are the weights assigned to the query words, query bigrams and expansion terms.

- Query Words: 0.1064 patrol, 0.1058 civil, 0.1046 training, 0.0758 participants.
- Query Bigrams: 0.0257 civil air, 0.0236 air patrol, 0.0104 training participants, 0.0104 participants receive.
- Expansion Terms: 0.0639 cadet, 0.0321 force, 0.0296 aerospace, 0.0280 cap.

Query-document scoring is then done as follows.

$$sc(Q, D) = \sum_{T \in \mathcal{T}} \sum_{\kappa \in T} \lambda_\kappa f(\kappa, D) \tag{5.19}$$

where $\mathcal{T}$ is the set of concept types and $f(\kappa, D)$ is a matching function that defines how related the concept κ is to D. It is computed as follows.

$$f(\kappa, D) = \log \frac{tf_{\kappa,D} + \mu \frac{tf_{\kappa,C}}{|C|}}{|D| + \mu} \tag{5.20}$$

In Eq. 5.19, the λs are parameterized in the same way as done in the WSD model. Thus, each λ_κ is represented as a linear weighted combination of importance features Φ^T of type T.

$$\lambda_\kappa = \sum_{\psi \in \Phi^T} w_\psi \psi(\kappa) \tag{5.21}$$

The initial set of ET-concepts are obtained using Latent Concept Expansion (LCE) as follows. First rank documents in the collection including only the concept types manifested in the query itself (QT-concepts, PH-concepts and PR-concepts). Then, all the words in the pseudo-relevant set of documents R (top ranked documents) are weighted by $w_{LCE}(\kappa)$ which is computed as follows.

$$w_{LCE}(\kappa) = \sum_{D \in \mathcal{R}} exp\left[\gamma_1 sc(Q, D) + \gamma_2 f(\kappa, D) - \gamma_3 \log \frac{tf_{\kappa,C}}{|C|}\right] \quad (5.22)$$

Top K words with highest LCE form the set of ET-concepts.

Each type is associated with the following six importance features.

- $GF(\kappa)$: Frequency of κ in Google N-Grams
- $WF(\kappa)$: Frequency of κ in Wikipedia titles
- $QF(\kappa)$: Frequency of κ in a search log
- $CF(\kappa)$: Frequency of κ in the collection
- $DF(\kappa)$: Document frequency of κ in the collection
- $AP(\kappa)$: A priori concept weight. AP is set to LCE weight for ET concepts, 1 for other types

Unlike WSD, in this case the parameter estimation needs to be done using a two stage optimization technique.

- Use coordinate ascent to learn feature importance weights for explicit concept types.
- Compute the top K ET-concepts. Again, use coordinate ascent to learn feature importance weights for explicit and latent concept types

The authors compared PQE with many other models and found PQE to be the best. The following models were compared: QL (§A.2), SD (§5.5), RM[10] [Lavrenko and Croft, 2003], LCE[10] (Eq. 5.22), WSD (§5.8), and WRM[10] (weighted version of RM). RM and LCE are both Pseudo-relevance feedback (PRF) expansion models. RM[10] means that 10 expansion concepts were used. Figure 5.2 compares these models with respect to the concepts they consider.

		Concept Types			
		QT	PH	PR	ET
Non-parameterized methods	QL	$\mathcal{N}$			
	SD	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	
	RM	$\mathcal{N}$			$\mathcal{N}$
	LCE	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$
Parameterized methods	WSD	$\mathcal{P}$	$\mathcal{P}$	$\mathcal{P}$	
	WRM	$\mathcal{P}$			$\mathcal{P}$
	PQE	$\mathcal{P}$	$\mathcal{P}$	$\mathcal{P}$	$\mathcal{P}$

Figure 5.2: Comparison of Concepts across Models ($\mathcal{N}$ means Non-parameterized, $\mathcal{P}$ means Parameterized) [Bendersky et al., 2011b]

5.10 Multiple Source Formulation (MSF)

Bendersky et al. [2012] further extended the PQE model to include more data sources which were used to compute richer features. The following features were used for concept weighting from external sources.

- Google N-Grams: Frequency of concept κ
- MSN Query Log: Frequency of concept κ
- Wikipedia Titles: Frequency of concept κ
- Retrieval Corpus: Document frequency of concept κ

The following features were used for query expansion from external sources.

- ClueWeb Heading Text: Single line of heading text (as defined by the $<h1>$–$<h6>$ tags)
- ClueWeb Anchor Text: Single line of anchor text (as defined by the $<a>$ tag)
- Wikipedia Corpus: Single article
- Retrieval Corpus: Single document

Compared to the PQE model, the initial set of expansion concepts are computed per source. These expansion concepts are combined across sources to get the initial pool of expansion concepts. Coordinate ascent is used in a two stage procedure to estimate all parameters as in PQE. Comparisons with SD (§5.5), WSD (§5.8), LCE[10] (Eq. 5.22), LCE-WP[10], PQE[10] (§5.9) show that the MSF[10] is better than all of these. Note that LCE-WP performs the pseudo-relevance feedback on Wikipedia, rather than using the retrieval corpus.

5.11 Query Hypergraphs

Various MRF models discussed till now (SD, WSD, PQE, MSF) model multiple dependencies between various query words. The query hypergraphs model, proposed by Bendersky and Croft [2012], is even more general as it models arbitrary higher order term dependencies (a dependency between term dependencies) as concepts. Vertices in a hypergraph correspond to individual query concepts. Dependency between a subset of concepts is modeled using a hyperedge. Query hypergraphs use passage-level evidence to model dependencies between concepts. They assign weights to both concepts and concept dependencies, proportionate to the estimate of their importance for expressing the query intent. Here are the examples of the possible structures and the concepts they might contain for the query "members of the rock group nirvana" (stop words removed).

- Words: members, rock, group, nirvana
- Bigrams: members rock, rock group, group nirvana
- Noun phrases: members, rock group nirvana
- Named entities: nirvana
- Dependencies: members nirvana, rock group

Figure 5.3 shows an example hypergraph representation. Let K^Q be the set of all concepts. Then the hypergraph has $V = K^Q \cup \{D\}$ and $E = \{(k, D) : k \in P^{K^Q}\}$ where P^{K^Q} stands for the power set of

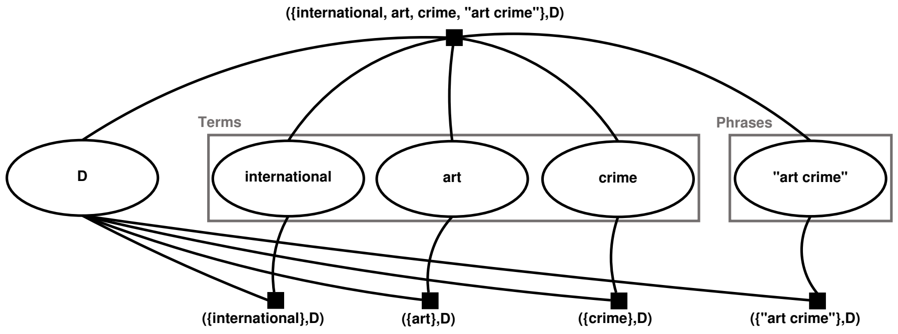


Figure 5.3: Example of a Hypergraph Representation for the Query "international art crime" [Bendersky and Croft, 2012]

K^Q. Similar to the other MRF models, relevance score to a document D for query Q is computed as follows.

$$sc(Q, D) = \prod_{e \in E} \psi_e(k_e, D) \equiv \sum_{e \in E} \log(\psi_e(k_e, D)) \tag{5.23}$$

where $\equiv$ means "rank equivalent".

Note that since the graph contains hyperedges, the potentials are defined over concept sets rather than single concepts. QT, PH and PR concepts were used as nodes in the hypergraph. The following two types of hyperedges were used.

- Local Edges: They link every concept to D. They represent the contribution of the concept κ to the total document relevance score, regardless of the other query concepts.
- Global Edges: A single global hyperedge (K^Q, D) is defined over the entire set of query concepts. It represents the contribution of each concept given its dependency on the entire set of query concepts.

Local potential functions are defined on local hyperedges and global potential functions on the global hyperedge. For both types of potential functions, the same matching function $f(\kappa, D)$ is used as in the PQE model [Bendersky et al., 2011b]. The local potential functions are defined as follows.

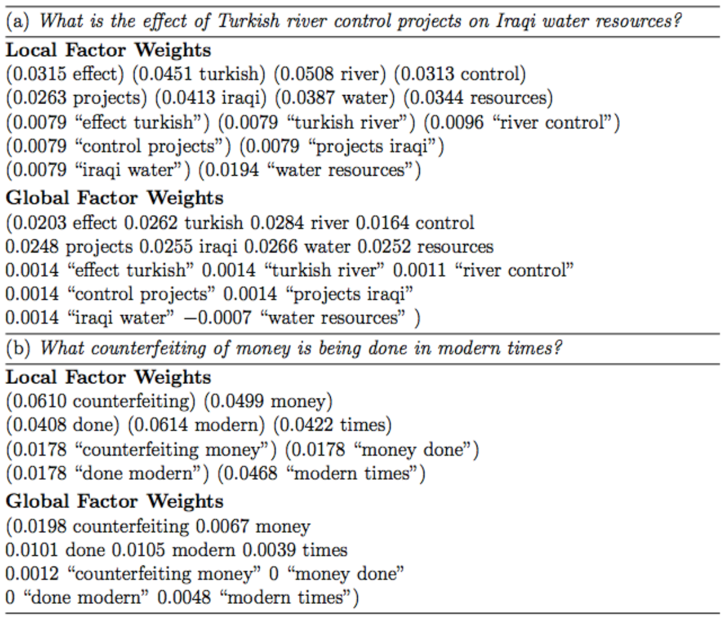
$$\psi(\kappa, D) = exp(\lambda(\kappa) f(\kappa, D)) \tag{5.24}$$

where $\lambda(\kappa)$ is the importance weight for concept κ. For the global potential function, the document D is divided into multiple overlapping passages Π_D and each passage $\pi \in \Pi_D$ is ranked with respect to the query. The potential function value is computed based on a match with the best matching passage (also called as the Max-Passage retrieval model).

$$\psi(K^Q, D) = exp\left[\max_{\pi \in \Pi_D} \sum_{\kappa \in K^Q} \lambda\left(\kappa, K^Q\right) f(\kappa, \pi)\right] \tag{5.25}$$

where $\lambda\left(\kappa, K^Q\right)$ is the importance weight of concept κ in the context of entire set of query concepts K^Q, and $f(\kappa, \pi)$ is a matching function between concept κ and a passage $\pi \in \Pi_D$. Intuitively, the global function assigns a higher relevance score to a document that contains many important concepts in the confines of a single passage. $\lambda(\kappa, D)$ and $\lambda(\kappa, K^Q)$ could be computed by parameterization by structure (tying all concepts of the same type together, as in SD (§5.5)) or by parameterization by concepts (as in PQE (§5.9)). Parameter estimation is done using a two stage coordinate ascent algorithm.

It was observed that the hypergraph-based versions of various models like QL, SD (§5.5), FD (§5.5), WSD (§5.8) were better than the original models. It is important to note, however, that relative improvements were diminishing as the baseline grew stronger. For instance, Bendersky and Croft [2012] note that while improvements over the QL model are around 5%, the improvements over the more sophisticated WSD model are around 1%, which demonstrates that WSD already captures some of the dependencies modeled by the global potential function in the query hypergraph. In addition, query hypergraph-based models optimize twice as many parameters as the original models, which might require a significant amount of training data, obtained either through labeling or implicit user feedback such as click data. However, the hypergraph approach is indeed helpful for many complex queries. For instance, Bendersky and Croft [2012] report that for the GOV2 collection, the hypergraph-based method improves the performance (in terms of MAP) for 60% of the queries compared to the WSD baseline, while hurting only 30% of the queries. Figure 5.4 demonstrates



(a) *What is the effect of Turkish river control projects on Iraqi water resources?*

Local Factor Weights
(0.0315 effect) (0.0451 turkish) (0.0508 river) (0.0313 control)
(0.0263 projects) (0.0413 iraqi) (0.0387 water) (0.0344 resources)
(0.0079 "effect turkish") (0.0079 "turkish river") (0.0096 "river control")
(0.0079 "control projects") (0.0079 "projects iraqi")
(0.0079 "iraqi water") (0.0194 "water resources")

Global Factor Weights
(0.0203 effect 0.0262 turkish 0.0284 river 0.0164 control
0.0248 projects 0.0255 iraqi 0.0266 water 0.0252 resources
0.0014 "effect turkish" 0.0014 "turkish river" 0.0011 "river control"
0.0014 "control projects" 0.0014 "projects iraqi"
0.0014 "iraqi water" −0.0007 "water resources")

(b) *What counterfeiting of money is being done in modern times?*

Local Factor Weights
(0.0610 counterfeiting) (0.0499 money)
(0.0408 done) (0.0614 modern) (0.0422 times)
(0.0178 "counterfeiting money") (0.0178 "money done")
(0.0178 "done modern") (0.0468 "modern times")

Global Factor Weights
(0.0198 counterfeiting 0.0067 money
0.0101 done 0.0105 modern 0.0039 times
0.0012 "counterfeiting money" 0 "money done"
0 "done modern" 0.0048 "modern times")

Figure 5.4: Analysis of the Best Performing Queries for Hypergraph-based Models [Bendersky, 2012]

some of the queries with the highest improvements, along with the local and global factor weights.

5.12 Summary

In this chapter, we discussed multiple methods to assign weights to the query words. Broadly we studied four kinds of methods: one using power iterations, another using features in the SVD space, regression rank and the fourth using Markov Random Fields. Within the Markov random field-based methods, we studied Sequential Dependence (SD), Quasi-Synchronous Dependency (QSD), Weighted Sequential Dependence (WSD), Parameterized Query Expansion (PQE), Multiple Source Formulation (MSF) and Query Hypergraphs. SD model considers dependencies between words only. QSD models the word

dependence based on syntactic relationships in queries. WSD models query word dependencies and weighs generic word concepts (e.g., unigrams, bigrams, etc.). The PQE model focuses both on explicit query concepts as well as on latent concepts that are associated with the query through pseudo-relevance feedback. Multiple external document sources have been helpful to extract robust statistical features in MSF. The query hypergraphs model is even more general as it models arbitrary higher order term dependencies (a dependency between term dependencies) as concepts. Only a few types of features have been used to define potential functions for the query hypergraphs model; more complicated feature functions from §3 could be used to further improve the accuracy. Overall, the dependency models have been shown to be significantly better than the models proposed in the previous chapters.
Suggested Further Reading: [Paik and Oard, 2014]: A fixed-point method; [Zhao and Callan, 2010]: Regression using features in the SVD space; [Lease et al., 2009]: Regression using knowledge of successful query models from past queries; [Metzler and Croft, 2005]: First work to model dependencies between query words; [Park et al., 2011]: Models long-distance syntactic word dependencies between query words; [Bendersky et al., 2010]: Parameterizing weights for generic word concepts; [Bendersky et al., 2011b]: Parameterizing weights for generic word concepts as well as latent concepts using query expansion; [Bendersky et al., 2012]: Using multiple external document sources; [Bendersky and Croft, 2012]:Model arbitrary higher order term dependencies.

6

Query Expansion by Including Related Concepts

6.1 Introduction

As discussed in Chapters 3 and 4, query reduction helps improve the performance for a verbose query. Better results were obtained using query weighting methods as discussed in Chapter 5. Although it sounds counter-intuitive, can we increase the performance of a verbose query by adding words?

Indeed, query expansion has a rich and successful history in information retrieval, where it has been shown to improve performance in a diverse set of applications. However, it needs to be applied with some caveats, since adding erroneous expansion terms to the query may hurt its performance. This is especially true in the case of verbose queries, which already contain redundant terms as discussed in §3.

Therefore, we start this chapter with a discussion on how to find out whether query expansion indeed could be helpful for verbose queries (§6.2). Then, we explore various query expansion techniques like adding a category label to queries (§6.3), adding weighted latent concepts to the query (§6.4), and interactive query expansion using pseudo-relevance feedback (§6.6). We also review a query expansion method where the user pro-actively provides guidance in the form of relevant

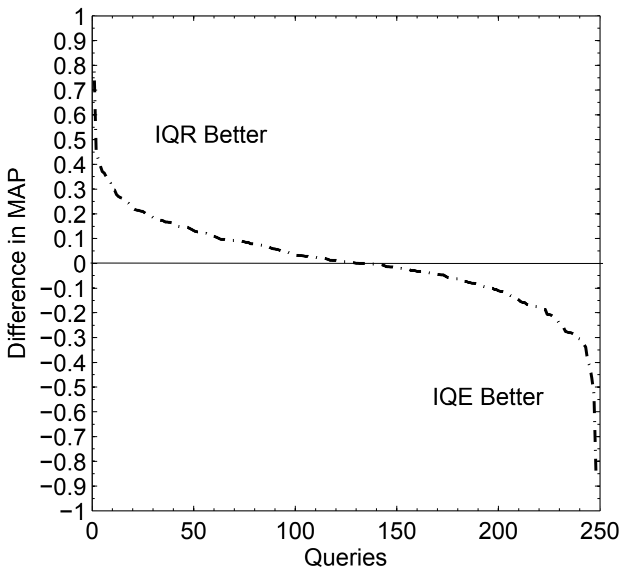


Figure 6.1: Some queries are better suited for IQR, while others can be better improved through IQE [Kumaran and Allan, 2008]

reference documents (§6.5). For interested readers, we provide a basic introduction to pseudo-relevance feedback (PRF) in §A.3.

6.2 When Could Query Expansion Help?

Figure 6.1 shows that there are certain queries for which query reduction is useful while for other queries, query expansion is good. Query expansion (QE) does not work in all cases. Intuitively, QE is helpful when the initial set of top k retrieved documents have a high precision. QE is also useful when the query is relatively short. How to define a measure to decide if QE will fail or succeed? Amati et al. [2004] study this problem and come up with two important measures: $Info_{Bo2}$ and $Info$ based on the DFR (Divergence From Randomness) framework. We provide a basic introduction to DFR in §A.4.

Let Q be a query, C be the document collection, and R be the set of top documents retrieved for Q from R using language models. The Divergence From Randomness (DFR) models are based on this simple idea: "The more the divergence of the within-document term-frequency from its frequency within the collection, the more the information carried by the word w in the document d." Using the Bose-Einstein distribution-based DFR framework, $Info_{Bo2}$ is defined as follows.

$$Info_{Bo2}(q_i) = -\log_2\left[\frac{1}{1+\lambda}\right] - tf_R(q_i).\log_2\left[\frac{\lambda}{1+\lambda}\right] \tag{6.1}$$

where $\lambda = |R|.\frac{tf_C(q_i)}{|C|}$, $|R|$ and $|C|$ are the total number of terms in R and C respectively, and $tf_R(q_i)$ is the frequency of q_i in R. We refer the reader to [Amati, 2003] for details of this formulation for the $Info_{Bo2}$ measure. When summed across all terms in the query, $Info_{Bo2}$ was found to be significantly positively correlated with Average Precision (correlation=0.52).

Let $InfoPrior(Q)$ be defined as follows.

$$InfoPrior(Q) = \sum_{q_i \in Q} -\log_2 \frac{tf(q_i, C)}{tf(C)} \tag{6.2}$$

Also let M_Q be defined as follows.

$$M_Q = \max\left[\frac{InfoPrior(Q) - \mu_{InfoPrior(Q)}}{\sigma_{InfoPrior(Q)}}, \max_{M \in DFR} \frac{Info_M(Q) - \mu_{Info_M(Q)}}{\sigma_{Info_M(Q)}}\right] \tag{6.3}$$

where the mean μ and standard deviation σ are computed across different query words. Note that here M denotes various DFR models like Divergence approximation of the binomial, Bose-Einstein distribution, Inverse Document Frequency model, etc. Finally, using these two, $Info(Q)$ can be defined as follows.

$$Info(Q) = \frac{1}{|Q|}\left[\frac{InfoPrior(Q) - \mu_{InfoPrior(Q)}}{\sigma_{InfoPrior(Q)}} + M_Q\right] \tag{6.4}$$

where $|Q|$ is the length of the query Q in words. The $Info$ measure was found to be related to Average Precision increase after QE activation.

They found that low $Info_{Bo2}$ is an indicator of a possible low outcome of average precision for the query, attesting thus when a query is possibly difficult. On the other hand, the information content measure $Info$ is negatively correlated to the increase in average precision after application of query expansion.

6.3 Adding a Category Label to Queries

The Open Directory Project (ODP) [1] is a multilingual open-content directory of World Wide Web links which uses a hierarchical ontology scheme for organizing site listings. Listings on a similar topic are grouped into categories which can then include smaller categories. Bailey et al. [2010] propose an approach that leverages past queries for which ODP category labels have been assigned, mines these past queries that are similar to rare long queries of interest, and propagates the labels from them to the rare long queries. For queries with successful URL trails, they assign a URL to one of the ODP categories using a back-off strategy as follows. If the URL does not match with an ODP category, remove successive path fragments from the end of URL until there is a match. Thus a query is assigned a distribution over ODP categories. This is then called as the aggregated labeled query trails (ALQT) for the query.

Given a long query, it is matched to obtain multiple similar ALQTs to obtain the ODP label for this long query. Four matching methods were proposed as follow.

- Term dropping: Drop least frequent term with respect to the query log.
- Named entities: Match any past query with at least one named entity from the long query.
- Language modeling: The long query is evaluated for match against all past queries using smoothed language model scores.
- BM25 scoring.

All matching ALQTs for the long query are further aggregated to compute an ALQT for the long query. Using the above strategies,

[1] `http://www.dmoz.org/`

it was possible to correctly identify the ODP category with about 20% accuracy even for long queries with no exact match with any past query. BM25 provided the best accuracy among the four techniques. These ODP category labels were then used to reorder the top-ranked search results. The following retrieval methods were compared: BM25F, Click, ODP, BM25F+Click, BM25F+ODP, Click+ODP, All (BM25F+ODF+Click). The findings suggest that re-ranking based on the ODP labels alone can outperform BM25F and click-through methods. Combining BM25 with ODP and also Click with ODP also appeared to help.

6.4 Parameterized Latent Concept Expansion

As discussed in Section 5.9, the Parameterized Query Expansion (PQE) model [Bendersky et al., 2011b] learns in a supervised setting the importance weights for explicit query concepts as well as latent concepts that are associated with the query through pseudo-relevance feedback. It was shown to be better than similar techniques that did not use the latent concepts. Bendersky et al. [2012] proposed an extension of the PQE model called the Multiple Source Formulation (MSF) (see Section 5.10) which performs query weighting and query expansion across multiple information sources. MSF was found to be better than many other query weighting models. Table 6.1 shows how different external sources could help in obtaining expansion terms for the query from different perspectives. For more detailed technical discussion of the PQE and MSF methods see Sections 5.9 and 5.10, where these models were introduced in the context of concept weighting techniques.

6.5 Expansion using User-supplied Reference Documents

Balog et al. [2008] address a specific enterprise document search scenario, where the information need is expressed in an elaborate manner. User information needs are expressed using a short query (of a few keywords) together with examples of key reference pages. Given this set of reference documents S, they propose methods for query-dependent as well as query-independent query expansion.

Table 6.1: Comparison between the lists of expansion terms derived from the individual external information sources for the query “toxic chemical weapon” and the combined list produced by MSF.

Source	Expansion Terms
Retrieval Corpus	chemical, weapon, toxic, convention, substance, gas, destruction, product, plant, mirzayanov
Wikipedia	chemical, agent, gas, weapon, warfare, war, poison, mustard, disseminate, nerve
Anchor Text	toxic, chemical, cigarette, tobacco, terrorist, tts, weapon, leach, terror, wwf
Heading Text	toxic, chemical, weapon, terrorist, terror, assess, biology, behavior, incinerate, emission
Combined	weapon, agent, gas, russia, convention, mustard, warfare, substance, destruction, product

Query likelihood model ranks documents as follows.

$$\log P(d|Q) \propto \log P(d) + \sum_{q_i \in Q} P(q_i|\theta_Q). \log P(q_i|\theta_D) \tag{6.5}$$

The document model is often smoothed with the collection statistics as follows.

$$P(q_i|\theta_D) = (1-\lambda).P(q_i|D) + \lambda.P(t|C) \tag{6.6}$$

One way to use set of reference documents S is to influence setting of the smoothing parameter λ. Balog et al. [2008] propose the following two methods to compute a query-dependent λ_Q.

- Maximizing Average Precision: This approach maximizes the average precision assuming that S is the only set of relevant documents given query Q.

- Maximizing Query Log Likelihood: This approach tries to maximize the log-likelihood of the query Q, given the set of sample documents S.

Besides influencing the smoothing parameter of the document model, the set S can also be used to influence the pseudo-relevance feedback based query expansion. Given any word t, the probability of the word t given the document set S can be computed as follows for query expansion.

$$P(t|S) = \sum_{D \in S} P(t|D).P(D|S) \tag{6.7}$$

The above model has two main components, one for estimating (expansion) word importance, and one for estimating the importance of the documents from which expansion words are selected.

The first part $P(t|D)$ can be computed in three different ways as follows.

- Maximum likelihood estimate: $P(t|D) = P_{ML}(t|D) = \frac{n(t,D)}{\sum_{t'} n(t',D)}$.
- Smoothed estimate of a word: $P(t|D) = (1 - \lambda).P_{ML}(t|D) + \lambda.P_{ML}(t|C)$.
- Unsupervised query expansion [Ponte and Croft, 1998]: $s(t) = \log \frac{P_{ML}(t|D)}{P_{ML}(t|C)}$ and $P(t|D) = \frac{s(t)}{\sum_{t'} s(t')}$.

The second part $P(D|S)$ can be computed in three different ways as follows.

- Uniform: $P(D|S) = 1/|S|$, i.e., all sample documents are assumed to be equally important.
- Query-biased: $P(D|S) \propto P(D|Q)$, i.e., a document's importance is approximated by its relevance to the original query.
- Inverse query-biased: $P(D|S) \propto 1 - P(D|Q)$, i.e., we reward documents that bring in aspects different from the query.

They observed that maximizing query log likelihood and maximizing average precision perform almost the same. They considered the query likelihood model with smoothing using maximum query log likelihood, and with no query expansion, as the baseline. Their query expansion based methods clearly outperformed the baseline. Uniform query expansion method brought in more "rare" relevant documents, that are not identified by the standard query-biased expansion methods.

6.6 Selective Interactive Reduction and Expansion

Interactive retrieval is a setting where the user is allowed to choose the reduced or expanded forms of the verbose query. As discussed in §3, it was observed that set cover pruning as well as snippet-based pruning worked fine with both Interactive Query Reduction (IQR) and Interactive Query Expansion (IQE). However, an inter-leaving of the top results from IQR and IQE (i.e., Selective Interactive Reduction and Expansion) with set cover pruning performed better than either IQR or IQE while significantly reducing the number of options [Kumaran and Allan, 2008].

6.7 Summary

For short queries, query expansion using pseudo-relevance feedback has been found to be very useful. However, for verbose queries, query expansion should be performed only if the predicted performance of the verbose query is low and when the expected improvement in performance using query expansion is high as indicated by the $Info_{Bo2}$ and the $Info$ measures. Query expansion with the right set of expansion words can be helpful even for verbose queries. We discussed various techniques for adding expansion terms: adding ODP category label to queries, adding weighted latent concepts to the query, and interactive query expansion using pseudo-relevance feedback. In the first and the third technique, words are added as a whole while in the second technique words are added along with a weight capturing their importance. The first technique needs a query log while the others use the traditional method of pseudo-relevance feedback for query expansion. Expansion-based techniques improve performance even for the Markov random field network-based models discussed in the previous chapter. Also, expansion using multiple data sources has been shown to be complementary and hence effective. Finally, we observed that query expansion becomes further effective when the user pro-actively provides guidance in the form of relevant reference documents.

Suggested Further Reading: [Bailey et al., 2010]: Adding a category label from ODP to queries using query logs; [Balog et al., 2008]: Expansion using user-supplied query-specific sample reference documents; [Bendersky et al., 2011b]: Expansion by modeling dependencies between concepts between query words and also latent concepts; [Kumaran and Allan, 2008]: Efficient and effective user interaction for query expansion.

7

Query Reformulation for Verbose Queries

7.1 Introduction

So far we have discussed query reduction, query weighting and query expansion mechanisms to deal with verbose queries. While, as shown in previous chapters, these methods can significantly boost retrieval performance they have one major drawback. These methods almost completely ignore user interaction with these queries, as can be evidenced from search logs, or other sources. Such interactions can be especially powerful in the context of web search and community question answering, which provide abundant sources of information on how users formulate and re-formulate the same information needs in different ways.

These reformulations can have vast differences in retrieval performance, which can be leveraged by various query log mining methods to boost the retrieval performance of the worst performing queries. For instance, QRU-1, a public dataset that contains reformulations of TREC queries as available in query logs[1] demonstrates that query per-

[1] http://ciir.cs.umass.edu/sigir2011/qru/

formance (in terms of NDCG) can be improved by more than 60%, if the best query formulation is used for each query.

A definition of query reformulation is quite general, and includes, among others, abbreviation induction, term expansion, term substitution, term reduction and URL or source suggestion (e.g., adding a relevant URL or source to a query). Query reformulation can also apply several of these transformations. For instance for a query like "Find information about Mitchell College in Connecticut", possible reformulations are (based on the QRU-1 dataset):

- "mitchell college new london" – expansion and reduction applied
- "mitchell college new london ct" – abbreviation induction applied
- "mitchell college mitchell.edu" – relevant URL added

As these examples demonstrate, query reformulation can be viewed as a generalization of the techniques discussed in the previous sections. However, as mentioned above, it often requires rich sources of implicit feedback, and might not be applicable in domains where such sources are unavailable.

While query reformulation can be applied to keyword queries as well, it can be especially helpful for verbose queries that can be rewritten in a more succinct way, or in which the query vocabulary is very different from the corpus vocabulary. Therefore, in this chapter, we discuss various query reformulation techniques using translation-based language models (§7.2), random walks on term-query and query-URL click graphs (§7.3), query logs (§7.4) and anchor text (§7.5).

7.2 Reformulation using Translation-based Language Model

Consider the queries asked on various community question answering platforms. Such questions are usually quite long and verbose in nature. In a Wondir[2] collection with roughly 1 million question-answer pairs, the average length for the question part and the answer part is 27 words and 28 words respectively. Given a new question, an interesting problem

[2]Wondir was a community based Q&A service, popular in 2004.

is to find a matching question from the Question-answer archives. In this case, one can treat the question as a verbose query. While there are a large number of papers dealing with this problem, we discuss the basic framework here as described in detail in [Xue et al., 2008].

The proposed retrieval model combines a word-to-word translation-based language model for the question part with a query likelihood approach for the answer part. The major problem is that there is a word mismatch between the user's question and the question-answer pairs in the archive. For example, "what is francis scott key best known for?" and "who wrote the star spangle banner?" are two very similar questions, but they have no words in common.

Assuming independence between the query words, probability of generating the user query q_u given the question-answer corpus (q, a) can be written as follows.

$$P(q_u|(q,a)) = \prod_{q_i \in q_u} P(q_i|(q,a)) \tag{7.1}$$

where the smoothed probability of observing a query word q_i given a question answer pair (q, a) can be expressed as follows.

$$P(q_i|(q,a)) = \frac{|(q,a)|}{|(q,a)|+\lambda} P_{mx}(q_i|(q,a)) + \frac{\lambda}{|(q,a)|+\lambda} P_{ml}(q_i|C) \tag{7.2}$$

Here $P_{ml}(q_i|C)$ is the ratio of number of times the word q_i appears in the collection C to the total size of C. Note that $P_{mx}(q_i|q,a)$ does not depend only on exact word match but can also incorporate translations as follows.

$$P_{mx}(q_i|(q,a)) = (1-\beta)P_{ml}(q_i|q) + \beta \sum_{t \in q} P(q_i|t)P_{ml}(t|q) \tag{7.3}$$

Here $P(q_i|t)$ is the translation probability. Incorporating the answer part, this can be rewritten as follows.

$$P_{mx}(q_i|(q,a)) = \alpha P_{ml}(q_i|q) + \beta \sum_{t \in q} P(q_i|t)P_{ml}(t|q) + \gamma P_{ml}(q_i|a) \tag{7.4}$$

where $\alpha + \beta + \gamma = 1$.

The next challenge is to learn the word-word translation probability $P(q_i|t)$. Consider a scenario of translation from English word

e to French word f. Given a English-French parallel corpus: $S = \{(\boldsymbol{e_1}, \boldsymbol{f_1}), (\boldsymbol{e_2}, \boldsymbol{f_2}), \ldots, (\boldsymbol{e_n}, \boldsymbol{f_n})\}$, EM algorithm works as follows. In the first step, traslation probability estimates are computed as follows.

$$P(f|e) = \lambda_e^{-1} \sum_{i=1}^{N} c(f|e; \boldsymbol{f_i}, \boldsymbol{e_i}) \tag{7.5}$$

where λ_e is a normalization factor across all f's. In the second step, the counts are updated as follows using the translation probability estimates computed in the previous step.

$$c(f|e; \boldsymbol{f_i}, \boldsymbol{e_i}) = \frac{P(f|e)}{P(f|e_1) + \ldots + P(f|e_l)} \#(f, \boldsymbol{f_i}) \#(e, \boldsymbol{e_i}) \tag{7.6}$$

For Q&A archives, either question or answer could be the source and target leading to $P(A|Q)$ or $P(Q|A)$. The two models could then be combined in the following two ways.

- Linear: This model computes the final translation probability as a linear combination of the two models.

$$P_{lin}(w_i|w_j) = (1 - \delta)P(w_i, Q|w_j, A) + \delta P(w_i, A|w_j, Q) \tag{7.7}$$

- Pooling: In this model, the dataset is expanded to incorporate both (q, a) and (a, q) pairs as $(q, a)_1, \ldots, (q, a)_n, (a, q)_1, \ldots, (a, q)_n$. This expanded dataset is then used with EM to compute $P_{pool}(w_i|w_j)$.

Xue et al. [2008] found that P_{pool} performed better than $P(A|Q)$, $P(Q|A)$ as well as P_{lin} in terms of the word-word translation accuracy. They also found that TransLM+QL is better than both LM-Comb and TransLM. Note that LM-Comb combines the LM for questions with the LM for answers without any translation term. Also, TransLM is the approach that does not consider the answer part when performing translation, while TransLM+QL does.

7.3 Reformulation using Random Walks

In this section, we discuss two methods which use random walks on term-query graph and click graph to compute query reformulations.

7.3.1 Query Log-based Term-Query Graph

Bonchi et al. [2011] present a method for performing query reformulation using random walks on a term-query graph. A term-query graph consists of terms and queries. A term is linked to a query if it appears in the query. A query q_2 is linked from q_1 iff the likelihood of query q_2 appearing after q_1 is not null. Query suggestions for a query $q = \{t_1, \ldots, t_m\}$ are generated as follows.

- Compute m random walks from each term in q.
- Compute Hadamard product of these m vectors to get a resultant score for every other query in the graph.
- Return top ranking queries as query reformulations.

For example, for the query "menu restaurant design", the following query reformulations were generated: "free restaurant design software", "free restaurant kitchen layout and design", "restaurant menu design", "restaurant menu design software", "restaurant menu design samples".

7.3.2 LambdaMerge with Click Graph

LambdaMerge, proposed by Sheldon et al. [2011], is a supervised merging method to merge results from several reformulations generated by random walks on a click graph. It directly optimizes a retrieval metric using features that describe both the reformulations and the documents they return.

For query q, let $q^{(1)}, q^{(2)}, \ldots, q^{(K)}$ be the different reformulations and let $D^{(1)}, D^{(2)}, \ldots, D^{(K)}$ be the corresponding result lists. Let $NormScore(q^{(k)}, d)$ be the relevance score of d for $q^{(k)}$ between 0 and 1. Consider two merging strategies: CombSum and CombRW as follows.

$$CombSum(d) = \sum_{k:d \in D^{(k)}} NormScore(q^{(k)}, d) \tag{7.8}$$

$$CombRW(d) = \sum_{k:d \in D^{(k)}} NormScore(q^{(k)}, d) \times W(D^{(k)}) \tag{7.9}$$

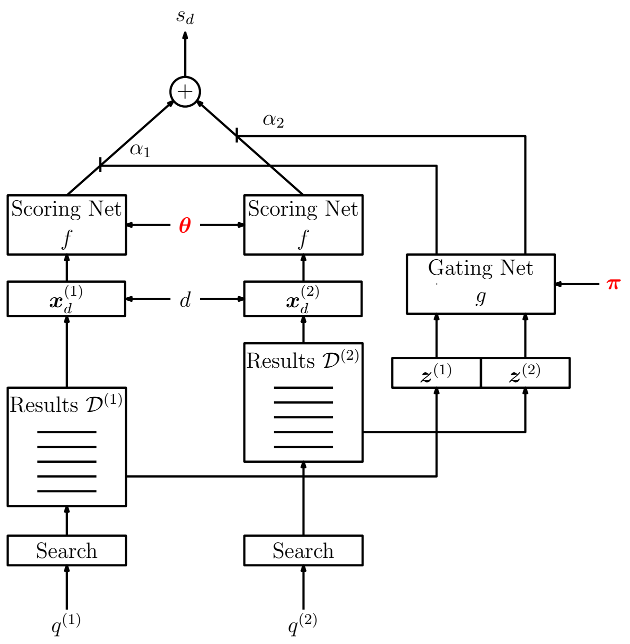


Figure 7.1: LambdaMerge Architecture [Sheldon et al., 2011]

where $W(D^{(k)})$ is the probability assigned to $q^{(k)}$ by a random walk in the query-URL graph starting at q.

LambdaMerge provides a flexible class of gated neural-network score-combination functions that (1) utilize multiple query-document features instead of just the retrieval score, (2) weigh contributions from each reformulation according to multiple features, such as those predicting list quality and query drift, so that contributions vary depending on the query and reformulation, and (3) are trained to optimize a retrieval metric. The following two types of features are used.

- Query-document features
  - Score: The original search engine ranking score.
  - Rank: The position of d in the ranked list $D^{(k)}$.
  - $NormScore_{[0,1]}$: The score normalized using min-max normalization.

  - $NormScore_{\mathcal{N}(0,1)}$: The score normalized to fit a standard Gaussian.
  - IsTopN: A binary feature to indicate if the document is within top 1, 3, 5, 10.

- Gating features that describe the quality of reformulation and its results list.
  - IsRewrite: A binary flag to distinguish between the original query and the reformulation.
  - RewriteScore: Random walk-based query reformulation score.
  - RewriteRank: 0 for the original query, then n for the n-th random walk reformulation as sorted by the random walk probability.
  - ListMean, ListStd, ListSkew: Respectively, mean, standard deviation and skew of raw ranking scores over each result list.
  - RewriteLen: Number of words in the reformulation.
  - Clarity: KL divergence between the query language model (over top 10 snippets) and the collection model.
  - Overlap@N: Overlap in the top 10 documents for the original query versus the reformulated query.
  - RAPP: This is computed as the linear regression model score trained to predict NDCG using the above features.

Let $x_d^{(k)}$ be a vector of query-document features for document d and k^{th} query reformulation $q^{(k)}$. Let $f(x;\theta)$ be a scoring function with parameters θ. Let $z^{(k)}$ be a vector of gating features for reformulation $q^{(k)}$ which describe the qualities of the k^{th} reformulation and its result list $D^{(k)}$. Gating network determines contribution of each reformulation to final score. The weight for the k^{th} result list is computed as follows.

$$\alpha_k = softmax(z^{(1)}, z^{(2)}, \ldots, z^{(k)}; \pi) = \frac{exp(\pi^T z^{(k)})}{\sum\limits_p exp(\pi^T z^{(p)})} \tag{7.10}$$

Mixing weights are normalized to sum to 1. Final score s_d for document d is given by the following equation.

$$s_d = \sum_k \alpha_k . f(x_d^{(k)}; \theta) \tag{7.11}$$

f could be any differentiable function like a linear function, a neural network, or a set of boosted decision trees. They use a two layer neural network with tanh activation functions. Scoring parameters θ and π are trained to optimize NDCG.

LambdaMerge was found to be better than CombSUM, CombRW, RAPP-L (linear regression trained to predict $\Delta NDCG$), and RAPP-Ω (oracle that chooses a single reformulation that gives highest NDCG@5). Performance gains were found to be higher for LambdaMerge with five reformulations compared to a single reformulation. RAPP, RewriteScore, RewriteRank, isRewrite, overlap@* and Overlap@1 were the most important features.

7.4 Reformulation using Query Logs

There are multiple ways in which query logs can be used for query reformulation.

Wang and Zhai [2008] extract term associations based on their context distribution in queries in the query log. For a new query, the method decides whether to substitute a term with one of its "similar" words based on whether this new word matches the context of the query better than the original term. Context sensitive stemming based on query logs is another type of query reformulation [Peng et al., 2007].

Xue et al. [2012] mine 5w1h question reformulation patterns from query logs. 5w1h questions are queries starting with where, why, what, when, who and how. For example, alternative expressions for the original question: "how far is it from Boston to Seattle" are "how many miles is it from Boston to Seattle", "distance from Boston to Seattle", "Boston to Seattle", "how long does it take to drive from Boston to Seattle". Also here are three question reformulation patterns generated for the query pair ("how far is it from Boston to Seattle", "distance from Boston to Seattle").

- S_1=Boston:("how far is it from X_1 to Seattle", "distance from X_1 to Seattle").
- S_2=Seattle:("how far is it from Boston to X_1", "distance from Boston to X_1").
- S_3=Boston, Seattle:("how far is it from X_1 to X_2", "distance from X_1 to X_2").

Xue et al. [2012] propose the following method. They extract query pairs (q, q_r) from the query log such that they are issued by the same user one after the other within a certain time period, and q is a 5w1h query. Patterns $P = \{(p, p_r)\}$ are then extracted from pairs with many common words and that occur with a high frequency. Given a new query q^{new} the algorithm picks the best question pattern p^* according to the number of prefix words and the total number of matching words in pattern. Query reformulations patterns containing p^* are ranked by frequency.

$$P(p_r|p^*) = \frac{f(p^*, p_r)}{\sum\limits_{p'_r} f(p^*, p'_r)} \tag{7.12}$$

Finally these reformulation patterns are used in the retrieval model as follows.

$$\begin{aligned} score(q^{new}, D) &= \lambda \log P(q^{new}|D) \\ +(1-\lambda) &\sum_{i=1}^{k} P(p_{r_i}|p^*) \log P(q^{new}_{r_i}|D) \end{aligned} \tag{7.13}$$

where $\{q^{new}_r\}$ are the k reformulations. They observed that using the question reformulations can significantly improve the retrieval performance of natural language questions.

7.5 Reformulation using Anchor Text

In absence of query logs, anchor text is a good approximation. Dang and Croft [2010] present a method which is very similar to [Wang and Zhai, 2008] but uses anchor text instead of query logs.

Here is a sketch of the approach proposed by Wang and Zhai [2008]. First, they estimate the context distribution for all words in the anchor stream or the query log as follows. Let $count_w(c_i)$ be the number of times word c_i occurs in context of w. Given a term w, the smoothed probability distribution of its context words can be written as follows.

$$\tilde{P_C}(c_i|w) = \frac{count_w(c_i) + \mu P(c_i|\theta)}{\sum\limits_{c_j \in C(w)} count_w(c_j) + \mu} \tag{7.14}$$

where $C(w)$ is the set of context words for w. They learn a translation model to translate from one word to another based on their distributional similarity.

$$t(s|w) = \frac{e^{-D(P_C(.|w)||\tilde{P_C}(.|s))}}{\sum\limits_u e^{-D(P_C(.|w)||\tilde{P_C}(.|u))}} \tag{7.15}$$

where the KL divergence term can be written as follows.

$$D(P_C(.|w)||\tilde{P_C}(.|s)) = \sum_{c \in C(w)} P(c|w) \log \frac{P(c|w)}{\tilde{P_C}(c|s)} \tag{7.16}$$

A substitution model is built on top of the translation model.

$$P(w_i \rightarrow s|q) \propto t(s|w_i) \times P(w_1 \ldots w_{i-1} w_{i+1} \ldots w_n|s) \tag{7.17}$$

where $P(w_1 \ldots w_{i-1} w_{i+1} \ldots w_n|s)$ can be considered as the probability that the new term s fits into the context of the query and is computed as follows.

$$\begin{aligned} &P(w_1 \ldots w_{i-1} w_{i+1} \ldots w_n|s) \\ &= \tilde{P_{L_2}}(w_{i-2}|s) \times \tilde{P_{L_1}}(w_{i-1}|s) \times \tilde{P_{R_1}}(w_{i+1}|s) \times \tilde{P_{R_2}}(w_{i+2}|s) \end{aligned} \tag{7.18}$$

Given a query term and a query context, substitution model decides whether to make a substitution, and if so, which candidates among those suggested by the translation model should be used. For each w_i in the query, try to replace each one of them. Consider only top M translation candidates s_i sorted by $t(s_i|w_i)$. Remove all s_i that have $NMI(s_i, w_i) < \tau$ where $NMI(s, w) = \frac{MI(s,w)}{MI(w,w)}$ and MI is computed over query log sessions. Substitutions are made if $\frac{P(w_i \to s_i|q)}{P(w_i \to w_i|q)} > 1$.

Besides substitution one could also do expansion. Dang and Croft [2010] observed that query substitution for long queries provided favorable results compared to the language modeling (LM) baseline and only a little worse compared to using the MSN query log. On the other hand, for query expansion, using anchors was better than both the LM baseline as well as using the MSN query log.

7.6 Summary

Query reformulation is a useful way of handling verbose queries especially when there is a mismatch between the original verbose query and the corpus vocabulary. There are various ways of deriving high confidence reformulation signals. We discussed four main sources of generating query reformulations: translation probabilities learned from the corpus itself, random walks on term-query and query-URL graphs, query logs and anchor text. Depending on the data source available, all these forms of reformulations could be effective. Since all of them contain complementary signals, a combination of reformulations generated by these techniques would surely be more useful. It has been shown that such reformulations help bridge the vocabulary gap effectively.

Suggested Further Reading: [Xue et al., 2008]: Reformulating questions to perform search on question-answer archives; [Bonchi et al., 2011]: Reformulation using random walks on a term-query graph; [Sheldon et al., 2011]: Reformulation using random walks on a click graph; [Xue et al., 2012]: 5w1h question reformulation patterns from query logs; [Dang and Croft, 2010]: Reformulation using Anchor Text.

8

Query Segmentation for Verbose Queries

8.1 Introduction

A verbose query often contains multiple concepts or pieces of information. Rather than reducing, expanding or reformulating the query as a whole, it might be helpful to first split the query into multiple segments and then process each segment separately.

Query segmentation can be formally defined as follows. Consider a query $Q = q_1, q_2, \ldots, q_n$ consisting of n query tokens. Segmentation is a mapping $S : Q \rightarrow y \in Y_n$, where y is a segmentation from the set Y_n. Since we can either have or not have a segmentation break at each of the $n - 1$ spaces between n tokens, $|Y_n| = 2^{n-1}$.

Query segmentation can be used to increase query recall by breaking the query into meaningful concepts. For instance, Parikh et al. [2013] observed that a verbose product description query, like "new ac adapter and battery charger for hp pavilion notebook" often suffers from zero-recall on eBay. But breaking the query into meaningful phrases, such as {new|ac adapter|and|battery charger|for|hp pavilion notebook} helps in reformulating the query as "new battery charger hp pavilion". This shortened query retrieves more than four thousand products.

In addition to increasing recall, correct query segmentation can also help with ambiguity resolution and query understanding. For instance, consider the query "new york times square". While segmentation like {new york times|square} is theoretically possible, the segmentation {new york|times square} is much more likely to express the true query intent.

This last example demonstrates that simple greedy query segmentation based on term co-occurrence will not always yield the optimal results. Therefore, in this chapter, we cover four kinds of more advanced methods for segmenting long queries: statistical (§8.2), supervised (§8.3), generative (§8.4) and NLP-based (§8.5).

8.2 Statistical Methods

These methods are term frequency-based or based on mutual information between terms. An approach proposed by Jones et al. [2006] uses mutual information (MI) between pairs of tokens as the sole factor in deciding on segmentation breaks. If the MI is above a threshold (optimized on a small training set), the pair of tokens is joined in a segment. Otherwise, a segmentation break is made. Risvik et al. [2003] proposed a method which combines the frequency count of a segment and mutual information between pairs of words in the segment in a heuristic scoring function. Mishra et al. [2011] use the bag of words model as the null model, and use this null model to find the probability of a multi-word expression (MWE) w to be a phrase. Given a set of n queries where each query contains each of the words of w, the method finds the number of queries m in this set that contain the MWE. MWE score is computed as $-\log\ \delta$ where δ is Hoeffding's upper bound on the probability that using the null model $\geq m$ queries may contain w as an MWE.

Parikh et al. [2013] proposed QSegment which focuses on domain-specific segmentation of e-commerce queries using frequency data from query log and product titles. Recent segmentation efforts use Wikipedia titles to find noun phrases and then perform segmentation. But this cannot be used for e-commerce (for new products, or products without a wiki page). Also, different phrases in a query are permuted more often in an e-commerce query versus web query making e-commerce query segmentation difficult.

Parikh et al. [2013] use a query segmentation which is similar to the one proposed by Hagen et al. [2010]. In this model, the score of a segmentation S depends on the length of each segment and also on the frequency of each segment.

$$score(S) = \begin{cases} \sum\limits_{s \in S, |s| \geq 2} |s|^{|s|}.freq(s), & \text{if } freq(s) > 0 \text{ for all } s \in S, |s| \geq 2 \\ -1, & \text{otherwise} \end{cases}$$

They observed that QSegment performed much better compared to the MWE approach [Mishra et al., 2011] and the MI baselines. As mentioned before, the MI baseline computes MI at different segment boundaries and based on that finds the best segmentation.

8.3 Supervised Methods

Given a query $\boldsymbol{x}$ with N terms, whether to partition at position i (where $0 < i < N$) can be modeled as a binary classification problem. Bergsma and Wang [2007] proposed a supervised approach using SVMs. For a position i, features are generated from tokens up to three positions to the left and to the right of the decision location i. That is for a decision between x_{L_0} and x_{R_0}, features are extracted from a window of six tokens in the query: $\{\ldots, x_{L_2}, x_{L_1}, x_{L_0}, x_{R_0}, x_{R_1}, x_{R_2}, \ldots\}$. The following three types of feature sets are considered.

- Decision boundary features
  - Indicator features: This set includes the following features. is-the-L_0 (token x_{L_0}="the"), is-the-R_0 (token x_{R_0}="the"), is-free-L_0 (token x_{L_0}="free"), is-free-R_0 (token x_{R_0}="free"), POS-tags (Part-of-speech tags of pair x_{L_0} and x_{R_0}), fwd-pos (position from beginning, i), rev-pos (position from end $N-i$)

  - Statistical features: This set includes the following features. web-count (count of x on the web), pair-count (web count of "w x''), definite (web count of "the w x"), collapsed (web count of "wx'' (one word)), and-count (web count of "w and x"), genitive (web count of "w's x"), Qcount-1 (count of x in the query database), Qcounts-2 (count of "w x'' in the query database).

- Context features: This set includes features to indicate what parts of the size-6 window are available. This also includes token-level, pair-wise, tri-gram and four-gram counts for all sub-sequences in the size-6 window.
- Dependency features: Pairwise counts of x_{L_0} and x_{R_1}; and of x_{L_1} and x_{R_0}. The intuition is to check, for example if the token x_{L_0} is more likely to modify a later token, such as x_{R_1}.

They found that using all the features provides better segmentation decision accuracy as well as better query segmentation accuracy compared to Jones et al. [2006]'s MI approach, as well as compared to approaches with less number of feature sets.

8.4 Generative Methods

Tan and Peng [2008] presented an unsupervised approach which uses a generative query model to recover a query's underlying concepts that compose its original segmented form. They assume that queries are made up of unigrams where rather than word unigrams they consider 'concept' unigrams. They compute the parameters of the unigram model (i.e., probability of these concepts) as follows. Given a huge web crawl, they compute frequencies of all possible n-grams up to a certain length ($n = 1, 2, \ldots, 5$) that occur at least once in the corpus. Let V be the vocabulary of all concepts. Then the probability of a concept x is computed as follows.

$$P_C(x) = \frac{\#x}{\sum\limits_{x' \in V} \#x'} \tag{8.1}$$

Probability of a segmentation is then computed as follows.

$$P(S^Q) = \prod_{s_i \in S^Q} P_C(s_i) \tag{8.2}$$

where s_i is the i^{th} segment in S^Q. Probability of a query can be written as follows.

$$P(Q) = \sum_{S^Q} P(S^Q) \tag{8.3}$$

where S^Q is one of the 2^{n-1} different segmentations and n is the number of query words. Top k segmentations from among all possible segmentations can be evaluated using a dynamic programming algorithm in $O(nkm\log(km))$ time where m is the maximum allowed segment length.

However, there are two problems with the above computation of concept probabilities. (1) It is unclear how to set V. One way is to limit V to n-grams up to a certain length, e.g., 5. But it is hard to justify why the higher order n-grams should be excluded from the normalization. (2) Concept probability should describe how likely the n-gram is to appear in a piece of text as an independent concept, which is not captured by raw frequencies. For example, P_C("York times") will be $\geq P_C$("new York times") which is incorrect. Hence, they propose another method to estimate the concept probabilities using an expectation-maximization (EM) algorithm, optimizing the minimum description length objective function on a partial corpus that is specific to the query. In the E step, the top K documents matching the query are segmented using the current set of estimated parameter values. In the M step, new set of parameter values (concept probabilities) are calculated to maximize the complete likelihood of the data which is augmented with segmentation information. To ensure that the output segments are well-formed concepts and not just frequent patterns, they incorporate evidence from Wikipedia titles. Tan and Peng [2008] observed that EM+Wiki provides better Segment F1 compared to baselines like MI, LM (Language modeling), EM and LM+Wiki.

8.5 NLP-based Methods

Bendersky et al. [2011a] exploited the dependency between different unsupervised text annotations to improve the accuracy of the entire set of annotations. Specifically, they leveraged the information about estimated parts-of-speech tags and capitalization of query terms to improve the accuracy of query segmentation. They worked with the following set of annotations $Z_Q = \{CAP, TAG, SEG\}$.

For example, here are the annotations for the query "who won the 2004 kentucky derby". Note that the capitalization label could be L: lowercase or C: otherwise; the POS labels could be N: noun, V: verb, or X: otherwise; and the segmentation label could be B/I: beginning of/inside the chunk.

- CAP: who (L), won (L), the (L), 2004 (L), kentucky (C), derby (C)
- TAG: who (X), won (V), the (X), 2004 (X), kentucky (N), derby (N)
- SEQ: who (B), won (I), the (B), 2004 (B), kentucky (B), derby (I)

The algorithm starts with an initial set of independently computed annotations $Z_Q^{*(I)}$. Given $Z_Q^{*(I)}$, the algorithm computes an annotation set $Z_Q^{*(J)}$ which jointly optimizes the probability of all annotations as follows.

$$Z_Q^{*(J)} = \arg\max_{Z_Q} p(Z_Q | Z_Q^{*(I)}) \tag{8.4}$$

This method of joint query annotation is basically a stacked classification in which a second, more effective, classifier is trained using the labels inferred by the first classifier as features.

Let $z_Q = (\zeta_1, \zeta_2, \ldots, \zeta_n)$ be an annotation in the set Z_Q. For the first step of independent query annotations, the following two methods could be used.

- Query-based estimation: Use large n-gram corpus to estimate $p(\zeta_i | q_i)$ for annotating query with capitalization and segmentation markup, and a standard POS tagger for POS tagging. Finally, for each $z_Q \in Z_Q$, the best annotation could be computed as follows.

$$z_Q^{*(QRY)} = \arg\max_{(\zeta_1,\ldots,\zeta_n)} \prod_{i \in (1,\ldots,n)} p(\zeta_i|q_i) \tag{8.5}$$

- PRF-based estimation: z_Q can be computed based on the set of top R sentences matching Q. $p(\zeta_i|q_i)$ is a smoothed estimator that combines the information from the retrieved sentence $r \in R$ with the information about unigrams (for capitalization and POS tagging) and bigrams (for segmentation) from a large n-gram corpus.

  $$p(z_Q|Q) \approx \sum_{r \in R} p(z_Q|r)p(r|Q) \tag{8.6}$$

  Intuitively, this equation models the query as a mixture of top k retrieved sentences, where each sentence is weighted by its relevance to the query. Finally, for each $z_Q \in Z_Q$, the best annotation could be computed as follows.

  $$z_Q^{*(PRF)} = \arg\max_{(\zeta_1,\ldots,\zeta_n)} \sum_{r \in R} \prod_{i \in (1,\ldots,n)} p(\zeta_i|r)p(r|Q) \tag{8.7}$$

Bendersky et al. [2011a] compared this method with two other methods: SEG-1 and SEG-2. SEG-1 is [Hagen et al., 2010] which was also used by Parikh et al. [2013]. SEG-2 is [Bergsma and Wang, 2007] which is based on supervised segmentation using a large number of features. Let i denote the independent approach and j denote a joint approach. They observed that $j{-}PRF$ is better than $i{-}QRY$, $j{-}QRY$ and $i{-}PRF$. Also $j{-}PRF$ was found to be 6.7% and 0.6% better than SEG1 and SEG2 respectively in terms of F1.

8.6 Summary

Segmentation is a critical component for handling verbose queries. Other transformations like reduction or weighting can be further applied on segmented queries. In this chapter, we discussed four main approaches to segmentation of verbose queries: statistical, supervised, generative and NLP-based. Statistical approaches are heavily dependent on corpus statistics, and cannot model dependencies beyond pair of words. While supervised approaches do not depend much on corpus statistics and can also model complex dependencies, they need a large amount of training data and well-designed domain dependent features. Generative and NLP-based methods help avoid such drawbacks and perform better than both statistical and supervised methods.

Suggested Further Reading: [Parikh et al., 2013]: QSegment for statistical segmentation using category-wise n-gram frequencies; [Bergsma and Wang, 2007]: Supervised Segmentation using decision-boundary, context and dependency features; [Tan and Peng, 2008]: Unsupervised segmentation using generative language models and Wikipedia; [Bendersky et al., 2011a]: Performing segmentation jointly with parts-of-speech tagging and capitalization.

9

Sources and Treatment of Verbose Queries

In this chapter, we will discuss a few domains where verbose queries are commonly found. While most of the verbose queries are explicitly asked by the users, some of them are implicit. Users ask verbose queries explicitly in a large number of scenarios. Advanced users searching for exhaustive list of relevant documents in medical literature, patents or legal documents often use verbose comprehensive queries. Naïve users like children or the elderly are not trained to ask short queries to search engines and hence end up using full sentence queries. Sometimes users end up firing long queries implicitly. For example, to find a relevant image for a paragraph in a textbook, one may fire the entire paragraph as a query to the search engine. In this chapter, we discuss these applications which have benefited significantly from verbose query processing techniques discussed in §3 to §8).

9.1 Finding Images for Books

The following problem is a direct application of reducing verbose queries to short sub-queries. Given chapter text from a book, get a

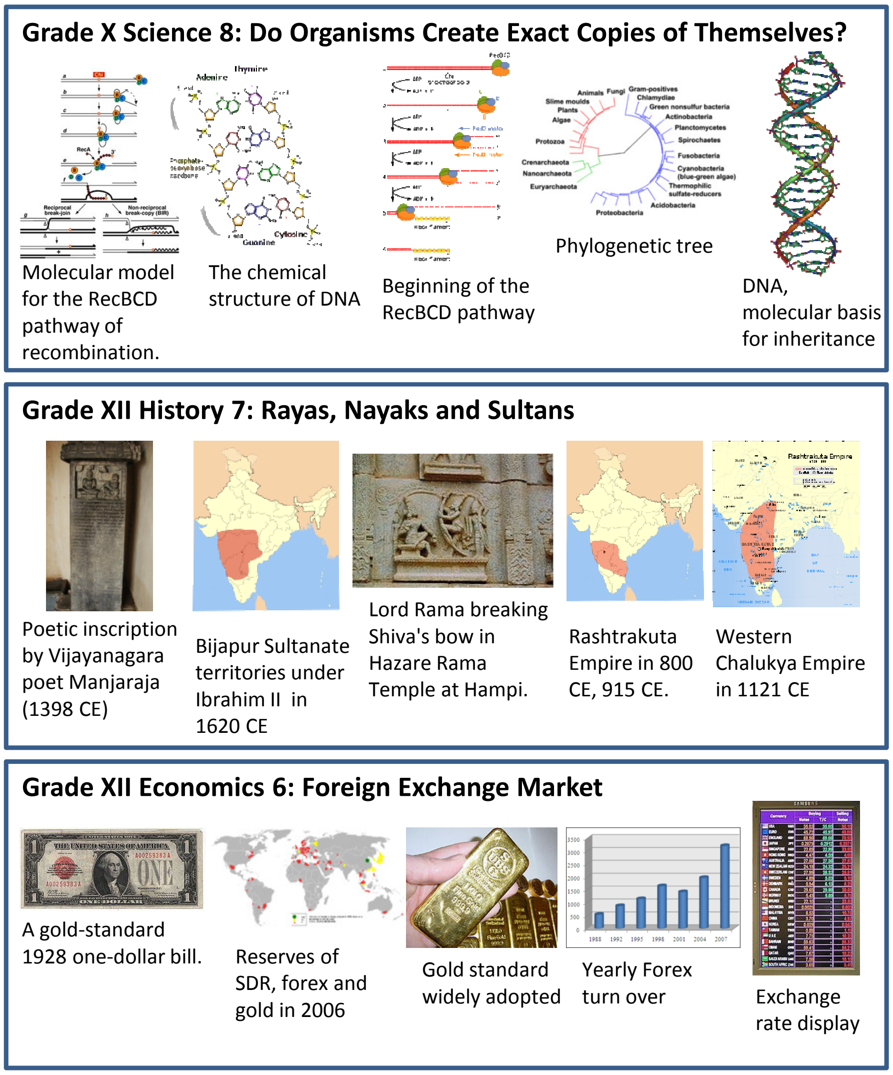

Grade X Science 8: Do Organisms Create Exact Copies of Themselves?
Molecular model for the RecBCD pathway of recombination.
The chemical structure of DNA
Beginning of the RecBCD pathway
Phylogenetic tree
DNA, molecular basis for inheritance
Grade XII History 7: Rayas, Nayaks and Sultans
Poetic inscription by Vijayanagara poet Manjaraja (1398 CE)
Bijapur Sultanate territories under Ibrahim II in 1620 CE
Lord Rama breaking Shiva's bow in Hazare Rama Temple at Hampi.
Rashtrakuta Empire in 800 CE, 915 CE.
Western Chalukya Empire in 1121 CE
Grade XII Economics 6: Foreign Exchange Market
A gold-standard 1928 one-dollar bill.
Reserves of SDR, forex and gold in 2006
Gold standard widely adopted
Yearly Forex turn over
Exchange rate display

Figure 9.1: Queries with Ranked Images [Agrawal et al., 2011]

ranked list of top k images. Agrawal et al. [2011] propose two algorithms to solve this problem: Affinity and Comity.

Affinity works as follows. It obtains key concept phrases from the chapter text using the linguistic pattern A^*N^+ where A denotes an adjective, and N denotes a noun. Next, it gets images from articles with high document similarity to the chapter text. Relevance score for an image is then computed by analyzing the overlap between the concept phrases and the cumulative metadata associated with the various copies of the image present in the narrowed set of articles.

Comity works as follows. It obtains up to top c concept phrases from the chapter text. It then forms queries consisting of two or three concept phrases each ($\binom{c}{2} + \binom{c}{3}$ queries in total). Next, it obtains up to top t image search results for each of the queries from e different search engines. Finally, it aggregates over (potentially $e \times \left[\binom{c}{2} + \binom{c}{3}\right]$) lists of images, to obtain relevance score values for each image and returns the top k images.

Agrawal et al. [2011] observed that for some categories of queries Affinity is better while Comity is better for other query categories. Overall, an ensemble of Comity and Affinity worked better than either of the two. Figure 9.1 shows a few queries along with the discovered images.

9.2 Finding Related Videos

While watching television, people increasingly consume additional content related to what they are watching. Odijk et al. [2015] consider the task of finding video content related to a live television broadcast for which they leverage the textual stream of subtitles associated with the broadcast. They model this task as a Markov decision process and propose a method that uses reinforcement learning to directly optimize the retrieval effectiveness of queries generated from the stream of subtitles. Their retrieval model consists of four parts as follows.

- Incrementally update a list of candidate query words that are obtained from the textual stream of subtitles.

- Compute a score for each word with a weighted sum of static and dynamic word features.
- Generate a query from the scored query word candidates. The generated queries are complex, consisting of many weighted query words, and they are selected from the highest scoring query word candidates.
- Retrieve the results.

The retrieval model defines a set of hyperparameters $w = w_d \cup w_s \cup w_f \cup w_n \cup w_e$ that each alter the retrieval process. The hyperparameters w_d and w_s can be construed as (dynamic and static) feature weights and w_f as field weights, while hyperparameters w_n and w_e alter the number of selected query words and the decay of query word candidate scores respectively. They use a set of word features that are either static for the word or updated dynamically with the stream of text. The static features are computed once for each new query word candidate in the stream. The dynamic features are updated in each new state for all query word candidates. The dynamic features are computed with information from the textual stream. They include the (1) term frequency $TF(w)$, (2) augmented term frequency, intended to prevent a bias towards longer documents, (3) $TF.IDF(w)$ computed using the target collection, and (4) $Capitalized(w)$ that indicates whether a word appeared capitalized in the stream, ignoring the beginning of a sentence. The static features are similar to the statistical features as described in §3.4. Hyperparameter w_e governs the decay rate. Concretely, the weighted sum of features is multiplied with $e^{-w_e \cdot i}$, where i is the relative age of word w in the stream thus far. This relative age ranges from 0 to 1 and is 0 for the current chunk and 1 for the first chunk in the stream. From the ranked query term candidates complex query is generated using the top n terms, where n is based on a hyperparameter w_n. The query is further extended with learned field weights, allowing the model to learn to attribute more importance to specific fields, such as the title of a video. The field weights w_f are also exposed as hyperparameters.

The learning approach considers the retrieval model as a black box that defines a set of hyperparameters, altering the retrieval process in a complex manner. A new state occurs when a new chunk of subtitles presents itself. The action of generating a query is optimized based on the feedback in terms of retrieval effectiveness. This feedback is obtained by generating search results through the retrieval model that is governed by the set of hyperparameters. Offline relevance assessments are used for feedback. Also, the Dueling Bandits Gradient Descent (DBGD) algorithm is used to learn an optimal policy.

They found that carefully weighting terms and decaying these weights based on recency significantly improves effectiveness.

9.3 Question Answering

Question answering platforms often face verbose queries. Huston and Croft [2010] use stop structure identification (as discussed in §3.5.5) and stopword removal for reducing the verbose query to a single subquery. Some examples of query reduction in query answering are as follows.

- "please tell me why the sky is blue" $\rightarrow$ "sky blue".
- "for a year ive been getting some tightening from within my chest" $\rightarrow$ "tightening chest".

An interesting problem in search on question answering platforms is to find the best matching already existing question for a new question. Xue et al. [2008] use translation-based language models for question reformulation. We discussed solution to this problem in detail in §7.

Here are examples of top five retrieved questions for two queries.

Top five retrieved questions for the query: "who is the leader of india?"

- Who is the prime minister of india?
- Who is current vice prime minister of india?
- Who is the army chief of india?

- Who is the finance minister of india?
- Who is the first prime minister of india?

Top five retrieved questions for the query: "who made the first airplane that could fly?"

- What is the oldest aiirline that still fly airplane?
- Who was the first one who fly with plane?
- Who was the first person to fly a plane?
- Who the first one fly to the space?
- Who the first one who fly to sky?

9.4 Searching for Medical Information

Bader and Theofanos [2003] analyzed three months of cancer queries on Ask.com. They found a large number of different types of verbose cancer related queries. The eagerness to know more makes users to write long queries. Also most of such users are novices and hence ask questions using whole sentences and keywords, often mis-spelt words. Types of queries were as follows: Cancer (78.37%), General Research (10.26%), Treatment (5.04%), Diagnosis and Testing (4.36%), Cause/Risk/Link (1.64%), Coping (0.33%). Here are a few examples of such queries.

- "Where can I find information about breast cancer?"
- "Where can I find a Web site with information on using high protein food to fight Breast cancer?"
- "How do antioxidants prevent cancer that may be caused by free radicals?"
- "where can I have an ultra fast CT scan done"
- "When do people who have pancreatic cancer know that they have the disease?"

9.5 Fact Verification

Fact verification is an area where users enter long fact queries like the following to verify the contained fact. "In 1981, Obama transferred to Columbia University in New York City, where he majored in political science with a specialty in international relations". This could be very useful for automatically identifying supporting documents for statements on Wikipedia.

Given a factual statement, the system, proposed by Leong and Cucerzan [2012], first transforms it into a set of semantic terms (likely to occur in the supporting document) by using boosted decision trees with multiple syntactic, encyclopedic and heuristic features (as discussed in §3.4). To further refine the set of words without hurting recall, it employs a quasi-random strategy for selecting subsets of the semantic terms according to topical likelihood. These semantic terms are used to construct queries. Retrieved documents are aggregated and re-ranked by employing additional measures of their suitability to support the factual statement.

Leong and Cucerzan [2012] specifically focused on Wikipedia which has in-sentence and end-of-sentence citations. The sentences (or their parts) are facts and the references serve as fact verification documents. They observed that verbatim query resulted in top 1 precision of 1.36; 95.67% results were not in top 50. After removing stop words, top 1 precision was 3.48; 90.08% were not in top 50. The oracle approach (which identifies matching documents by considering only the words that are shared between the statement and the true referenced document) had top 1 precision of 8.48%; 71.67% were not in top 50.

Let t be a term and S be the verbose query (or the fact). They used the following set of features.

- Offset(t,S): Backward distance (measured in semantic terms) from the end of the statement (the citation marker in Wikipedia).

- TermCount(S): The number of unique semantic terms.

- Keyphraseness(t): $\frac{N_{sd}(t)}{N_s(t)}$, where $N_{sd}(t)$ is the number of times t is an overlapping term, $N_s(t)$ is the number of statements in which t occurs in training.

- WikitextFreq(t): The term's frequency in the associated Wikipedia article (or, generally, the article where the term is found).

- POS(t, S): The part-of-speech tag (Penn Treebank) of the term in the statement.

- TF-IDF(t, S): The product of the term's frequency in factual statement and its IDF derived from two corpora, BNC and NYT.

- Multiword(t): A binary value indicating whether the semantic term is a multi-word compound.

To drop terms further and select only best k, one method is to drop terms randomly. A smarter way is to use Pachinko Allocation model to compute similarity between the term and the fact in topic space. Probability of a term being dropped depends on this topic match score. Multiple such size k queries are fired. Results from multiple searches are combined and re-ranked using the following score for a document d.

$$Score(d) = \frac{1}{minRank(d)} \times \cos(d, S) \times PageRank(d) \tag{9.1}$$

where S is the fact and $minRank(d)$ is the minimum rank for document d across multiple lists.

They observed that their system (drop-k PAM) is better than other baselines: Verbatim without stop words, using classifier but no dropping, drop-k uniform and also the oracle (which searches based on overlapping words only). Drop-k PAM achieved 9.96% top 1 precision, 21.4% top 10 precision and 28.96% top 50 precision. Fact verification URLs generated by Drop-k PAM were found to have 71% quality, 74.7% ease of effort, and were 88.7% trustworthy.

9.6 Natural Language Interface for Databases

XML databases need users to write structured queries in XQuery. Hence, it is non-trivial to handle an arbitrary English language sentence as query input, which can include aggregation, nesting, and value joins, among other things, and convert it to a structured query. NaLIX [Li et al., 2006] is a generic interactive natural language query interface to an XML database. It translates the query, potentially after reformulation, into an XQuery expression that can be evaluated against an XML database. The translation is done through mapping grammatical proximity of natural language parsed tokens to proximity of corresponding elements in the result XML. Query Translation consists of the following three phases.

- Parse tree classification: Identifies words/phrases in parse tree of original sentence that can be mapped into XQuery components or not.
- Parse tree validation: Checks if the parse tree is mappable to an XQuery. Checks if element/attribute names or values contained in user query can be found in the database.
- Parse tree translation: Utilizes the structure of the NL constructions as reflected in the parse tree to generate an appropriate structure in XQuery expression.

This method can be treated as a specific form of verbose query processing which uses query segmentation and query reformulation approaches, along with domain-specific signals.

9.7 E-Commerce

A large number of queries on e-commerce portals are long. Parikh et al. [2013] observed that a long query, like “new ac adapter and battery charger for hp pavilion notebook” often suffers from zero-recall on eBay. But breaking the query into meaningful phrases, such as {new|ac adapter|and|battery charger|for|hp pavilion notebook} helps in reformulating the query as “new battery charger hp pavilion”. This shortened query retrieves more than four thousand products. They proposed an algorithm called QSegment that uses frequency data from past query logs and product titles for domain-specific segmentation of such long queries. We discussed this algorithm in detail in §8.

Besides segmentation, query reduction can also be helpful for e-commerce queries. Yang et al. [2014] experimented with reducing queries by single term deletion and found that accurate prediction of term deletion can potentially help users recover from poor search results and improve shopping experience. For example, "small carry on bag for air plane" results in no search results while the query "carry on bag" leads to many relevant items. They perform single term deletion by learning domain-specific classifiers using multiple term-dependent and query-dependent features as described in §3.

9.8 Search Queries from Children

Query length is an indicator of the complexity of the query and the difficulty of the user to express information needs using keywords. Torres et al. [2010] experimented with the AOL query log and found that the average query length for the kids, teens and mature teens were 3.8, 3.4 and 3.2 words, respectively. The overall average query length was 2.5 words. These studies suggest that query reduction and query segmentation techniques can be particularly beneficial for children content queries. Similarly query reformulation techniques based on morphological and syntactical features of the queries can improve the search process by mapping phrases to concepts and keywords. Torres et al. [2010] found that compared to adults, kids look at a much larger set of results and fire new queries far less often. It should be noted that in a more recent study, Gossen et al. [2011] found contradictory results. Using query logs of three German search engines targeted towards children, they found that children queries are shorter in length compared to adult queries.

9.9 Music Search

Lee [2010] discuss a large number and type of examples of verbose queries in music search. People often first encounter new music from various media (e.g., a movie, TV commercial, radio) without ever knowing the artist name and song title, or they often forget the information over the course of time. This brings challenges to known-item searches for music as those users must attempt to describe the sought music other than using the bibliographic information. People who had no formal music education or people who seek music from different cultures or music in non-native languages can experience difficulties describing the music they seek. Here is an example query: "I heard this song by a female singer in an ARBY's. I believe it is from the 70s or early 80s. The main chorus of the song says, "over and over again." Kind of a sad, slow, easy listening love song."

Types of music related queries include the following.

- Identification: Questions asking for assistance in identifying information objects
- Location: Questions asking about the location of a specific information object and/or source
- Verification: Questions asking for assistance in verifying some information the user has
- Recommendation: Questions asking for a list of music related information objects
- Evaluation: Questions asking for an evaluative discussion of a particular subject (e.g., review)
- Ready reference: Questions asking for simple, factual answers (e.g., birth/death date of an artist, record label)
- Reproduction: Questions asking for text, taken directly from an information source and unchanged (e.g., lyrics)

- Description: Questions asking for a description of something (e.g., description of artist, album)
- Research: Questions asking for involved answers requiring some effort and wide use of information sources to formulate.

Some examples of music queries are as follows.

- It has a female voice similar to Lora Logic or Linder Sterling from Ludus (I always thought this was a song by Ludus but now I have their complete discography and it isn't there).
- Sounds like the Temptations or some band from the 1960s.
- The whole thing sounds a little John Mayersque. In that genre, with Jack Johnson, Jason Mraz, etc.
- They were the sort of hyper skinny 26 year old white guys you think of when you think of Red Hot chili peppers ... sort of a repetitive pop song in the spirit of, say, FatBoy Slims "wonderful night".

Many users also seem to search for music based on information from other people, especially when they are searching for lullabies or folk songs. For example, they heard a song repeatedly from a family member when they were younger and later they try to search for the right song often based on the fuzzy memory they have from their childhood (e.g., "My grandfather, who was born in 1899, used to sing me to sleep with this song and I can't remember the words"). This type of search can turn out to be quite difficult because often the information that was relayed to them in the first place might not be precise or accurate, and secondly these queries are quite long. Such queries clearly need processing using techniques described in this book.

9.10 Queries from User Selected Text

People browsing the web or reading a document may see text passages that describe a topic of interest, and want to know more about it by searching. They could select a piece of such text and may want to get relevant insights. To retrieve insights, the system needs to generate effective queries from the content of an arbitrary user selected text passage.

For example, consider the following text segment: "Knee joint replacement may be recommended for: Severe arthritis (osteoarthritis or rheumatoid arthritis) of the knee that has not gotten better with medicine, injections, and physical therapy after 6 months or more of treatment. Your doctor may recommend knee replacement for these problems: Inability to sleep through the night because of knee pain. Knee pain that has not improved with other treatment. Knee pain that limits or keeps you from being able to do your normal activities, especially your daily activities such as bathing, preparing meals, household chores, and other things. Some tumors that affect the knee."

The following text chunks could be extracted from the above text segment. "household chores, knee replacement, knee pain, Severe arthritis, osteoarthritis, rheumatoid arthritis, injections, tumors, joint replacement, bathing, Inability, physical therapy, normal activities, knee, meals, daily activities, other treatment, medicine, treatment, 6 months".

These text chunks could be combined in various ways to form a query. Lee and Croft [2012] use CRF-perf model (proposed by Xue et al. [2010]) with rich features to identify discriminative phrases or words which could form the query. They also tried various ways of weighting query terms – ones that incorporate the number of results from search API, CRF probability scores, removing stop words to obtain high retrieval performance for such verbose queries.

Another example of queries from user selected text is for the entity linking applications where the mention text (or the user selected text) is large in size. For example, given a user highlighted event mention from a cricket match report, Gupta [2015] solve the problem of linking it to a relevant set of balls from the corresponding match commentary. Here the event mention could often times be an entire sentence, and the problem is to find matching "ball" documents from the ball-by-ball commentaries. The problem is solved by reducing the mention to a short query containing important entities like player name and country name and then using it to query the ball-by-ball corpus.

9.11 Summary

Verbose queries are frequent in multiple applications. In some applications, verbose queries are fired by the users explicitly, while in other applications they are implicit. We discussed examples from various domains like finding images for books, finding related videos, question answering, cancer search portals, fact verification, natural language interfaces for databases, e-commerce, children search engines, music search, queries from user selected text, etc. Implementations of all such systems need to have a module that can convert the verbose query to a short one, and then optionally a reformulated query. The variety of applications where verbose queries are frequent and the rate at which such scenarios are increasing make verbose query handling a critical topic.
Suggested Further Reading: [Agrawal et al., 2011]: Finding images for books; [Odijk et al., 2015]: Finding related videos for a streaming video; [Xue et al., 2008]: Finding related questions given a new question from question-answer archives; [Bader and Theofanos, 2003]: Cancer-related search queries; [Leong and Cucerzan, 2012]: Fact verification on Wikipedia using syntactic, encyclopedic and heuristic features; [Li et al., 2006]: Natural language interface for XML databases; [Parikh et al., 2013]: Long queries on e-commerce portals; [Torres et al., 2010]: Analysis of search queries from children; [Lee, 2010]: Verbose queries in music search; [Lee and Croft, 2012]: Implicit verbose queries from user-selected text; [Gupta, 2015]: Implicit verbose queries from cricket match reports.

10

Summary and Research Directions

With the advent of various new ways of querying, there is a strong need to solve the problem of "answering verbose queries" more effectively. We discussed six main ways of handling verbose queries – query reduction to a single sub-query, query reduction to multiple sub-queries, query weighting, query expansion, query reformulation, and query segmentation. We also discussed various applications where supporting search for verbose queries can make a significant difference.

This survey shows that the work on information retrieval with verbose queries has already produced many valuable insights and significant improvements in the state-of-the-art of the current information retrieval models. However, much remains to be done, and many exciting research directions are still in their infancy or completely unexplored. In the remainder of this chapter, we will examine some of these research directions in depth.

10.1 Towards a Unified Verbose Query Processing Framework

In this survey, we discussed multiple methods for query reduction, weighting, expansion, segmentation and reformulation. Each of these methods was shown to be effective on its own, in certain scenarios. However, the researchers have yet to find a principled way to unify all these transformations into a joint query processing framework. Clearly, simply performing all possible combinations of query transformations leads to an exponential explosion of candidate transformations that could be considered and is infeasible. Therefore, there is a need for a principled model for effective and efficient selection of candidate transformations for a given verbose query.

As an interesting step in this direction, Guo et al. [2008] propose a unified model for query refinement based on Conditional Random Fields (CRF). They consider a number of tasks such as spelling error correction, word splitting, word merging, phrase segmentation, word stemming, and acronym expansion as "refinement operations", and integrate them into a single graphical model, CRF-QR, which encodes the joint dependency of the refined query on both the input query words and the refinement operations. Guo et al. [2008] demonstrate that the CRF-QR model can improve upon existing cascade models that simply apply the query transformations in sequence.

However, this query refinement approach mostly deals with misspelled, or poorly specified queries, and will have little to no effect on well-formed grammatically correct verbose queries, since it applies no weighting to the terms and concepts in the refined queries. To this end, an interesting direction for future work is incorporating query refinement into an existing weighting framework, for instance the query hypergraphs, proposed by Bendersky and Croft [2012] (see §5 for details). Such an integration will make the current work on concept weighting for verbose queries more applicable in realistic search scenarios where the input verbose queries may contain typos and grammatical errors.

Another promising direction towards a unified query processing framework is incorporating query transformations into current state-of-the-art learning-to-rank algorithms [Cao et al., 2007]. These algorithms mostly treat the query-document pairs as numerical features. Specifically linear ranking functions – used in many IR applications due to their effectiveness and efficiency – learn the same ranking function across all queries. Therefore, methods for query-dependent transformation that will encode information about all possible query reductions, weightings, expansions and reformulations as numerical features that can be effectively used by all learning-to-rank methods, including the linear ones, is an important research problem.

Better understanding of how learning-to-rank and query transformations can be combined is a very fruitful direction for future research, as evidenced by some recent work by Iyyer et al. [2014] that employs deep learning for question answering. Thus far, information retrieval approaches to this problem have been relatively simplistic. For example, Dang et al. [2013] propose a two-stage solution, where query concept weighting is done at the retrieval stage, and the resulting documents are re-ranked using a standard learning-to-rank method.

10.2 Multi-modal Verbose Query Processing

As mobile computing becomes ubiquitous, verbose search queries are often spoken, rather than written. However, with the voice-based search, Automatic Speech Recognition can have errors leading to "incorrect" queries. One interesting direction for future research is understanding what is the most effective way to simultaneously auto-correct errors in Automatic Speech Recognition and perform query processing simultaneously. Right now, in most systems these functionalities are decoupled, which leads to an error propagation and amplification between the speech recognition and the information retrieval systems.

In addition to single-modality queries, we can imagine a scenario where queried information has multiple modalities. For example, a user can ask the search engine: "What is the name of the blue flower in this photo?", while taking a photo with her mobile device. In this case, a query consists of both spoken and visual modalities, and the search engine needs to both understand the visual representation of the "blue flower" concept in the taken picture, as well as find occurrences of this visual representation on the web, and finally map this visual representation back into the concept “iris” for providing the right answer. While this scenario seems futuristic at the time of writing, current research suggests that we will see a lot of progress in this area in the few coming years [Socher et al., 2011].

10.3 Search Personalization

Search personalization and user modeling is an important research topic. One of the key observations in search personalization is that an individual’s search history can be effectively utilized for improving search relevance [Bennett et al., 2012]. For instance, one simplistic way to model a user, in the context of search personalization, is a weighted combination of its search queries.

While user modeling is a well-established research field, to the best of our knowledge the techniques described in this survey, such as query expansion, query weighting and query reformulation, have not yet been applied in the context of user search history, even if a new user query was not directly observed in user history. For example, query expansion algorithms can take into account prior user queries in addition to other information sources. Similarly query weighting can be biased towards concepts which are known to be important to a specific user. Data sparsity in such applications (e.g., for users with few queries in their history) can be addressed by either user or query clustering (e.g., dimensionality reduction methods such as topic models or word embeddings).

10.4 Natural Language Query Understanding

The techniques described in this survey apply only basic morphological and syntactic processing to documents and user queries. However, in many cases, natural language processing (NLP) techniques such as part-of-speech tagging, dependency parsing, summarization, sentiment analysis, coreference resolution, entity and relation extraction, can prove highly beneficial if synthesized with more traditional information retrieval techniques.

NLP and IR synthesis has the potential to revolutionize search in many domains, from the web to the enterprise. It will help search engines to tackle complex queries with location and temporal features and enable retrieval based on entities and relations, rather than keyword matching. For example, consider queries such as "companies that specialize in IR or NLP in Cambridge, MA", or "most influential academic papers related to information retrieval published in the last five years" that would be difficult to answer precisely with the current information retrieval systems.

There have been some successful attempts of such synthesis within a restricted domain (for example, IBM's Watson DeepQA system [Ferrucci et al., 2010] winning Jeopardy!). However, a general theoretical framework – one that can be applied to all types of information retrieval tasks and can efficiently handle large-scale corpora on web scale – is yet to emerge. Making progress towards such a framework, based on our current understanding of verbose search queries, will significantly advance both IR and NLP research, and will undoubtedly inspire a large number of commercially viable applications for individual users, enterprises and governments.

Acknowledgements

We are indebted to many people for making this survey possible. We thank Mark Sanderson and Doug Oard for reviewing our first version and helping us produce a significantly improved final version. We thank the anonymous reviewers for their insightful detailed comments. We thank our families for being patient, and for their constant support and encouragement. Finally, we thank our employers Microsoft and Google respectively for their support.

Appendices

A

Basic Information Retrieval Concepts

In this chapter, we will discuss the following basic information retrieval concepts: language modeling, the query likelihood model, the divergence from randomness framework, singular value decomposition, and pseudo-relevance feedback.

A.1 Language Modeling

A statistical language model assigns a probability to a sequence of m words $P(q_1, \ldots, q_m)$ by means of a probability distribution. It reflects how frequently the set of words appear as a sequence in a reference collection of documents C or a single document. Language models are used in information retrieval in the query likelihood model.

In a unigram language model, probability of a sequence is computed as follows.

$$P(q_1, \ldots, q_m) = \prod_{i=1}^{m} P(q_i) \tag{A.1}$$

where $P(q_i)$ is computed as $\frac{\text{count}(q_i)}{|C|}$ where $\text{count}(q_i)$ is the number of times q_i appears in C and $|C|$ is the size of C in number of words.

In a n-gram language model, probability of a sequence is computed as follows.

$$P(q_1, \ldots, q_m) = \prod_{i=1}^{m} P(q_i \mid q_{i-(n-1)}, \ldots, q_{i-1}) \tag{A.2}$$

where $P(q_i \mid q_{i-(n-1)}, \ldots, q_{i-1})$ is computed as $\frac{\text{count}(q_{i-(n-1)}, \ldots, q_{i-1}, q_i)}{\text{count}(q_{i-(n-1)}, \ldots, q_{i-1})}$.

Language models are usually used with some form of smoothing.

A.2 Query Likelihood Model

Query likelihood model [Ponte and Croft, 1998] is a basic method for using language models in IR. Our goal is to rank documents by $P(d|q)$, where the probability of a document is interpreted as the likelihood that it is relevant to the query. Using Bayes' rule, the probability $P(d|q)$ of a document d, given a query q can be written as follows.

$$P(d|q) = \frac{P(q|d)P(d)}{P(q)} \tag{A.3}$$

Since the probability of the query $P(q)$ is the same for all documents, this can be ignored. Further, it is typical to assume that the probability of documents is uniform. Thus, $P(d)$ is also ignored leading to the following.

$$P(d|q) \propto P(q|d) \tag{A.4}$$

Documents are then ranked by the probability that a query is observed as a random sample from the document model.

$$P(q|d) \propto \prod_{i=1}^{n} P(q_i|d) \tag{A.5}$$

A.3 Pseudo-Relevance Feedback

The idea behind relevance feedback in IR systems is to take the results that are initially returned from a given query and to use information about whether or not those results are relevant to perform a new query. Pseudo-relevance feedback automates the process of feedback. The procedure is as follows.

- Take the results returned by initial query as relevant results (only top k with k being between 10 to 50 in most experiments).
- Select top few terms from these documents using for instance the TF-IDF weights.
- Do query expansion, add these terms to query, find relevant documents for this query and finally return the most relevant documents.

Pseudo-relevance feedback is frequently used for query expansion. Relevance feedback is often implemented using the Rocchio Algorithm.

A.4 Divergence from Randomness Framework

Divergence from randomness is one type of probabilistic model. The DFR models are based on this simple idea: "The more the divergence of the within-document term-frequency from its frequency within the collection, the more the information carried by the word t in the document d". In other words the term-weight is inversely related to the probability of term-frequency within the document d obtained by a model M of randomness.

$$\text{weight}(t|d) \propto -\log \mathrm{P}_M(t \in d|\text{Collection}) \tag{A.6}$$

where the subscript M stands for the type of model of randomness employed to compute the probability. In order to choose the appropriate model M of randomness, we can use different models. There are many ways to choose M, each of these provides a basic DFR model. The basic DFR models are Divergence approximation of the binomial, Approximation of the binomial, Bose-Einstein distribution, Geometric approximation of the Bose-Einstein, Inverse Document Frequency model, Inverse Term Frequency model, and the Inverse Expected Document Frequency model.

A.5 Singular Value Decomposition

Singular value decomposition of an $m \times n$ real or complex matrix M is a factorization of the form $M = U\Sigma V^T$, where U is an $m \times m$ real or complex unitary matrix, Σ is an $m \times n$ rectangular diagonal matrix with non-negative real numbers on the diagonal, and V^T (the conjugate transpose of V, or simply the transpose of V if V is real) is an $n \times n$ real or complex unitary matrix. The diagonal entries $\Sigma_{i,i}$ of Σ are known as the singular values of M. The m columns of U and the n columns of V are called the left-singular vectors and right-singular vectors of M, respectively.

A.6 Metrics

In this section, we discuss various metrics that are popularly used to evaluate the performance of information retrieval systems.

A.6.1 Precision@K

Precision is the fraction of retrieved documents that are relevant to the query.

$$\text{precision} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{retrieved documents}\}|} \tag{A.7}$$

When precision is evaluated at a given cut-off rank, considering only the topmost results returned by the system, it is called precision at k or P@k.

A.6.2 Normalized Discounted Cumulative Gain (NDCG)

Cumulative Gain (CG) is the sum of the graded relevance values of all results in a search result list. The CG at a particular rank position p is defined as follows.

$$\mathrm{CG_p} = \sum_{i=1}^{p} rel_i \tag{A.8}$$

where rel_i is the graded relevance of the result at position i. The value computed with the CG function is unaffected by changes in the ordering of search results.

The premise of Discounted Cumulative Gain (DCG) is that highly relevant documents appearing lower in a search result list should be penalized as the graded relevance value is reduced logarithmically proportional to the position of the result. The discounted CG accumulated at a particular rank position p is defined as follows.

$$\mathrm{DCG_p} = \sum_{i=1}^{p} \frac{2^{rel_i} - 1}{\log_2(i+1)} \tag{A.9}$$

Sorting the documents of a result list by relevance produces the maximum possible DCG till position p, also called Ideal DCG ($IDCG$) till that position. For a query, the normalized discounted cumulative gain, or $NDCG$, is computed as follows.

$$\mathrm{NDCG_p} = \frac{DCG_p}{IDCG_p} \tag{A.10}$$

The NDCG values for all queries can be averaged to obtain a measure of the average performance of a search engine's ranking algorithm. Note that in a perfect ranking algorithm, the DCG_p will be the same as the $IDCG_p$ producing an $NDCG$ of 1.0. All $NDCG$ calculations are then relative values on the interval 0.0 to 1.0.

A.6.3 Mean Average Precision (MAP)

Average precision at n is computed as follows.

$$\mathrm{AP} = \frac{\sum_{k=1}^{n}(P(k) \times \mathrm{rel}(k))}{\text{number of relevant documents}} \tag{A.11}$$

where $\mathrm{rel}(k)$ is an indicator function equaling 1 if the item at rank k is a relevant document, zero otherwise. Also, n is the number of retrieved documents.

Note that the average is over all relevant documents and the relevant documents not retrieved get a precision score of zero. Mean average precision for a set of queries is the mean of the average precision scores for each query.

$$\mathrm{MAP} = \frac{\sum_{q=1}^{Q} \mathrm{AP(q)}}{Q} \tag{A.12}$$

where Q is the number of queries.

A.6.4 Mean Reciprocal Rank (MRR)

The reciprocal rank of a query response is the multiplicative inverse of the rank of the first correct answer. The mean reciprocal rank is the average of the reciprocal ranks of results for a sample of queries Q

$$\text{MRR} = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{\text{rank}_i}. \tag{A.13}$$

The reciprocal value of the mean reciprocal rank corresponds to the harmonic mean of the ranks.

B

Graphical Models: MRFs and CRFs

In this chapter, we provide a basic introduction to two graphical models: Markov Random Fields (MRFs) and Conditional Random Fields (CRFs).

B.1 Markov Random Fields (MRFs)

A Markov random field (MRF) is a set of random variables having a Markov property described by an undirected graph. A Markov random field is similar to a Bayesian network in its representation of dependencies; the differences being that Bayesian networks are directed and acyclic, whereas Markov networks are undirected and may be cyclic.

Given an undirected graph $G = (V, E)$, a set of random variables X form a Markov random field with respect to G if they satisfy the local Markov properties: (1) Pairwise Markov property: Any two non-adjacent variables are conditionally independent given all other variables. (2) Local Markov property: A variable is conditionally independent of all other variables given its neighbors. (3) Global Markov property: Any two subsets of variables are conditionally independent given a separating subset.

As the Markov properties of an arbitrary probability distribution can be difficult to establish, a commonly used class of Markov random fields are those that can be factorized according to the cliques of the graph.

Given a set of random variables X, let $P(X = x)$ be the probability of X taking on the particular value x. Because X is a set, the probability of x should be understood to be taken with respect to a joint distribution of the variables in X. If this joint density can be factorized over the cliques of G, $P(X = x) = \prod_{c \in C(G)} \phi_c(x_c)$ then X forms a Markov random field with respect to G. Here, $C(G)$ is the set of cliques of G. The functions ϕ_c are sometimes referred to as factor potentials or clique potentials.

B.2 Conditional Random Fields (CRFs)

One notable variant of a Markov random field is a Conditional random field, in which each random variable may also be conditioned upon a set of global observations x. Conditional random fields are graphical models that can capture such dependencies among the input. A CRF over observations X and hidden labels Y is a conditional distribution $P(y|x)$ where $y \in Y$ and $x \in X$. CRF is an undirected graphical model in which each vertex represents a random variable whose distribution is to be inferred, and each edge represents a dependency between two random variables. The observation x can be dependent on the current hidden label y, previous n hidden labels and on any of the other observations in a n order CRF. Sutton and McCallum [2006] provide an excellent tutorial on CRFs. CRFs have been shown to outperform other probabilistic graphical models like Hidden Markov Models (HMMs) and Maximum Entropy Markov Models (MeMMs).

For a typical CRF, the distribution over labels y is given by the following equation.

$$P(y|x) = \frac{exp\left[\sum_{k=1}^{K} \lambda_k f_k(x, y)\right]}{Z(x)} \tag{B.1}$$

where the partition function $Z(x)$ is computed as follows.

$$Z(x) = \sum_{y \in Y} exp\left[\sum_{k=1}^{K} \lambda_k f_k(x, y)\right] \tag{B.2}$$

Here, f_k are the feature functions, and λ_k is the weight of the k^{th} feature.

For general graphs, the problem of exact inference in CRFs is intractable. The inference problem for a CRF is basically the same as for an MRF. However there exist special cases for which exact inference is feasible. If exact inference is impossible, several algorithms like loopy belief propagation can be used to obtain approximate solutions.

C

Dependency Parsing

Dependency is the notion that linguistic units, e.g. words, are connected to each other by directed links. The (finite) verb is taken to be the structural center of clause structure. All other syntactic units (words) are either directly or indirectly connected to the verb in terms of the directed links, which are called dependencies. Dependency parsers capture such linguistic dependencies between words and output directed dependency trees over words: each word in the tree has exactly one incoming edge, which comes from its 'parent', except the root word which has no incoming edges.

Dependency parse trees help to illustrate the binary dependencies existing between various query words as shown in Figure 3.2.

Synchronous grammars, originally proposed for machine translation, jointly generate trees of a source and target sentence. Words in a source tree (e.g., in a document) are not always translated into a target tree (e.g., a query) with the same syntactic structure. Some source words may be translated into one target word, and others may be match more than one word or a phrase. To solve these disagreements in source and target languages, a quasi-synchronous model allows words in a target sentence, which are aligned with a words in a parent-child dependency relation in a source sentence, to have a different relationship to each other.

Various kinds of relationships between nodes can be considered by the quasi-synchronous grammars. Some of these relations are parent-child, ancestor-descendant, siblings and c-commanding as shown in Figure 3.3. A node A c-commands a node B if and only if (a) neither A nor B dominate the other, and (b) the lowest branching node that dominates A also dominates B. If a node A dominates a node B, A appears above B in the tree. Note that c-command is not necessarily a symmetric relation. In other words, a node A can c-command a node B without B c-commanding A. For example, in Figure 3.3, "chemical" c-commands "weapons" but not vice-versa.

References

Elena Agapie, Gene Golovchinsky, and Pernilla Qvarfordt. Leading People to Longer Queries. In *Proc. of the SIGCHI Conf. on Human Factors in Computing Systems (SIGCHI)*, pages 3019–3022, New York, NY, USA, 2013. ACM.

Rakesh Agrawal, Sreenivas Gollapudi, Anitha Kannan, and Krishnaram Kenthapadi. Enriching Textbooks with Images. In *Proc. of the* 20th *ACM Intl. Conf. on Information and Knowledge Management (CIKM)*, pages 1847–1856, New York, NY, USA, 2011. ACM.

James Allan, Jamie Callan, W Bruce Croft, Lisa Ballesteros, John Broglio, Jinxi Xu, and Hongming Shu. INQUERY at TREC-5. In *Proc. of the* 5th *Text REtrieval Conference (TREC)*, pages 119–132. NIST, 1996.

Giambattista Amati. *Probability Models for Information Retrieval based on Divergence From Randomness.* PhD thesis, University of Glasgow, 2003.

Giambattista Amati, Claudio Carpineto, and Giovanni Romano. Query Difficulty, Robustness, and Selective Application of Query Expansion. In *Proc. of the* 25th *European Conf. on Information Retrieval (ECIR)*, pages 127–137, Berlin, Heidelberg, 2004. Springer-Verlag.

Avi Arampatzis and Jaap Kamps. A Study of Query Length. In *Proc. of the* 31st *Annual Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR)*, pages 811–812, New York, NY, USA, 2008. ACM.

Judith L Bader and Mary Frances Theofanos. Searching for Cancer Information on the Internet: Analyzing Natural Language Search Queries. *Journal of Medical Internet Research*, 5(4):e31, 2003.

Peter Bailey, Ryen W White, Han Liu, and Giridhar Kumaran. Mining Historic Query Trails to Label Long and Rare Search Engine Queries. *ACM Transactions on the Web (TWEB)*, 4(4):15, 2010.

Niranjan Balasubramanian, Giridhar Kumaran, and Vitor R Carvalho. Exploring Reductions for Long Web Queries. In *Proc. of the* 33rd *Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR)*, pages 571–578, New York, NY, USA, 2010. ACM.

Krisztian Balog, Wouter Weerkamp, and Maarten de Rijke. A Few Examples go a Long Way: Constructing Query Models from Elaborate Query Formulations. In *Proc. of the* 31st *Annual Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR)*, pages 371–378. ACM, 2008.

Michael Bendersky. *Information Retrieval with Query Hypegraphs.* Ir, University of Massachusetts Amherst, July, 2012.

Michael Bendersky and W Bruce Croft. Discovering Key Concepts in Verbose Queries. In *Proc. of the* 31st *Annual Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR)*, pages 491–498, New York, NY, USA, 2008. ACM.

Michael Bendersky and W Bruce Croft. Analysis of Long Queries in a Large Scale Search Log. In *Proc. of the 2009 Workshop on Web Search Click Data (WSCD)*, pages 8–14, New York, NY, USA, 2009. ACM.

Michael Bendersky and W Bruce Croft. Modeling Higher-Order Term Dependencies in Information Retrieval using Query Hypergraphs. In *Proc. of the* 35th *Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR)*, pages 941–950, New York, NY, USA, 2012. ACM.

Michael Bendersky, Donald Metzler, and W Bruce Croft. Learning Concept Importance using a Weighted Dependence Model. In *Proc. of the* 3rd *ACM Intl. Conf. on Web Search and Data Mining (WSDM)*, pages 31–40, New York, NY, USA, 2010. ACM.

Michael Bendersky, W Bruce Croft, and David A Smith. Joint Annotation of Search Queries. In *Proc. of the* 49th *Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (HLT)*, pages 102–111, Stroudsburg, PA, USA, 2011a. Association for Computational Linguistics.

Michael Bendersky, Donald Metzler, and W Bruce Croft. Parameterized Concept Weighting in Verbose Queries. In *Proc. of the* 34th *Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR)*, pages 605–614, New York, NY, USA, 2011b. ACM.

Michael Bendersky, Donald Metzler, and W Bruce Croft. Effective Query Formulation with Multiple Information Sources. In *Proc. of the* 5th *ACM Intl. Conf. on Web Search and Data Mining (WSDM)*, pages 443–452, New York, NY, USA, 2012. ACM.

Paul N. Bennett, Ryen W. White, Wei Chu, Susan T. Dumais, Peter Bailey, Fedor Borisyuk, and Xiaoyuan Cui. Modeling the Impact of Short- and Long-term Behavior on Search Personalization. In *Proc. of the* 35th *Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR)*, pages 185–194, New York, NY, USA, 2012.

Shane Bergsma and Qin Iris Wang. Learning Noun Phrase Query Segmentation. In *Proc. of the 2007 Joint Conf. on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, volume 7, pages 819–826, Prague, Czech Republic, 2007. Association for Computational Linguistics.

Francesco Bonchi, Raffaele Perego, Fabrizio Silvestri, Hossein Vahabi, and Rossano Venturini. Recommendations for the Long Tail by Term-Query Graph. In *Proc. of the* 20th *Intl. Conf. Companion on World Wide Web (WWW)*, pages 15–16, New York, NY, USA, 2011. ACM.

Thorsten Brants and Alex Franz. Web 1T 5-gram Version 1. `https://catalog.ldc.upenn.edu/LDC2006T13`, 2006.

Andrei Z Broder, David Carmel, Michael Herscovici, Aya Soffer, and Jason Zien. Efficient query evaluation using a two-level retrieval process. In *Proceedings of the* 12th *International Conference on Information and Knowledge Management*, pages 426–434. ACM, 2003.

Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to Rank: From Pairwise Approach to Listwise Approach. In *Proc. of the* 24th *Intl. Conf. on Machine Learning (ICML)*, pages 129–136, New York, NY, USA, 2007. ACM.

Yan Chen and Yan-Qing Zhang. A Query Substitution-Search Result Refinement Approach for Long Query Web Searches. In *Proc. of the 2009 IEEE/WIC/ACM Intl. Joint Conf. on Web Intelligence and Intelligent Agent Technology-Volume 01 (WI-IAT)*, pages 245–251, Washington, DC, USA, 2009. IEEE Computer Society.

Sungbin Choi, Jinwook Choi, Sooyoung Yoo, Heechun Kim, and Youngho Lee. Semantic concept-enriched dependence model for medical information retrieval. *Journal of Biomedical Informatics*, 47(0):18 – 27, 2014.

Kenneth Church and William Gale. Inverse Document Frequency (idf): A Measure of Deviations from Poisson. In *Natural Language Processing using Very Large Corpora*, pages 283–295. Springer, 1999.

Fabio Crestani and Heather Du. Written versus Spoken Queries: A Qualitative and Quantitative Comparative Analysis. *Journal of the American Society for Information Science and Technology (JASIST)*, 57(7):881–890, 2006.

Ronan Cummins, Mounia Lalmas, Colm O'Riordan, and Joemon M Jose. Navigating the User Query Space. In *Proc. of the* 18*th* *Intl. Symposium on String Processing and Information Retrieval (SPIRE)*, pages 380–385, Berlin, Heidelberg, 2011. Springer-Verlag.

Ronan Cummins, Jiaul H. Paik, and Yuanhua Lv. A Pólya Urn Document Language Model for Improved Information Retrieval. *ACM Transactions on Information Systems (TOIS)*, 33(4):21:1–21:34, May 2015.

Van Dang and Bruce W Croft. Query Reformulation using Anchor Text. In *Proc. of the* 3*rd* *ACM Intl. Conf. on Web Search and Data Mining (WSDM)*, pages 41–50, New York, NY, USA, 2010. ACM.

Van Dang, Michael Bendersky, and W. Bruce Croft. Two-Stage Learning to Rank for Information Retrieval. In *Proc. of the* 35*th* *European Conf. on Advances in Information Retrieval (ECIR)*, pages 423–434, Berlin, Heidelberg, 2013. Springer-Verlag.

Sudip Datta and Vasudeva Varma. Tossing Coins to Trim Long Queries. In *Proc. of the* 34*th* *Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR)*, pages 1255–1256, New York, NY, USA, 2011. ACM.

David A. Ferrucci, Eric W. Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya Kalyanpur, Adam Lally, J. William Murdock, Eric Nyberg, John M. Prager, Nico Schlaefer, and Christopher A. Welty. Building Watson: An Overview of the DeepQA Project. *AI Magazine*, 31:59–79, 2010.

Kristofer Franzen and Jussi Karlgren. Verbosity and Interface Design. Technical Report T2000:04, Swedish Institute of Computer Science, 2000.

Tatiana Gossen, Thomas Low, and Andreas Nürnberger. What are the Real Differences of Children's and Adults' Web Search. In *Proc. of the* 34*th* *Annual Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR)*, pages 1115–1116. ACM, 2011.

Jiafeng Guo, Gu Xu, Hang Li, and Xueqi Cheng. A Unified and Discriminative Model for Query Refinement. In *Proc. of the* 31*st* *Annual Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR)*, pages 379–386, New York, NY, USA, 2008. ACM.

Manish Gupta. CricketLinking: Linking Event Mentions from Cricket Match Reports to Ball Entities in Commentaries. In *Proc. of the* 38th *Annual Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR)*, New York, NY, USA, 2015. ACM.

Matthias Hagen, Martin Potthast, Benno Stein, and Christof Braeutigam. The Power of Naïve Query Segmentation. In *Proc. of the* 33rd *Annual Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR)*, pages 797–798, New York, NY, USA, 2010. ACM.

Samuel Huston and W Bruce Croft. Evaluating Verbose Query Processing Techniques. In *Proc. of the* 33rd *Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR)*, pages 291–298, New York, NY, USA, 2010. ACM.

Mohit Iyyer, Jordan Boyd-Graber, Leonardo Claudino, Richard Socher, and Hal Daumé III. A Neural Network for Factoid Question Answering over Paragraphs. In *Proc. of the 2014 Intl. Conf. on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

Jing Jiang and Chengxiang Zhai. An Empirical Study of Tokenization Strategies for Biomedical Information Retrieval. *Information Retrieval*, 10(4-5): 341–363, 2007.

Rosie Jones and Daniel C Fain. Query Word Deletion Prediction. In *Proc. of the* 26th *Annual Intl. ACM SIGIR Conf. on Research and Development in Informaion Retrieval (SIGIR)*, pages 435–436, New York, NY, USA, 2003. ACM.

Rosie Jones, Benjamin Rey, Omid Madani, and Wiley Greiner. Generating Query Substitutions. In *Proc. of the 15th Intl. Conf. on World Wide Web (WWW)*, pages 387–396, New York, NY, USA, 2006. ACM.

Ron Kohavi and George H John. Wrappers for feature subset selection. *Artificial intelligence*, 97(1):273–324, 1997.

Giridhar Kumaran and James Allan. A Case For Shorter Queries, and Helping Users Create Them. In *Proc. of the Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, pages 220–227, Stroudsburg, PA, USA, 2007. The Association for Computational Linguistics.

Giridhar Kumaran and James Allan. Effective and Efficient User Interaction for Long Queries. In *Proc. of the* 31st *Annual Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR)*, pages 11–18, New York, NY, USA, 2008. ACM.

Giridhar Kumaran and Vitor R Carvalho. Reducing Long Queries using Query Quality Predictors. In *Proc. of the* 32nd *Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR)*, pages 564–571, New York, NY, USA, 2009. ACM.

Tessa Lau and Eric Horvitz. Patterns of Search: Analyzing and Modeling Web Query Refinement. In *Proc. of the* 7th *Intl. Conf. on User Modeling (UM)*, pages 119–128, Secaucus, NJ, USA, 1999. Springer-Verlag New York, Inc.

Victor Lavrenko and W Bruce Croft. Relevance Models in Information Retrieval. In *Language Modeling for Information Retrieval*, pages 11–56. Springer, Netherlands, 2003.

Matthew Lease. An Improved Markov Random Field Model for Supporting Verbose Queries. In *Proc. of the* 32nd *Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR)*, pages 476–483, New York, NY, USA, 2009. ACM.

Matthew Lease, James Allan, and W Bruce Croft. Regression Rank: Learning to Meet the Opportunity of Descriptive Queries. In *Proc. of the* 31th *European Conf. on IR Research on Advances in Information Retrieval (ECIR)*, pages 90–101, Berlin, Heidelberg, 2009. Springer-Verlag.

Chia-Jung Lee and W Bruce Croft. Generating Queries from User-Selected Text. In *Proc. of the* 4th *Symposium on Information Interaction in Context (IIiX)*, pages 100–109, New York, NY, USA, 2012. ACM.

Chia-Jung Lee, Ruey-Cheng Chen, Shao-Hang Kao, and Pu-Jen Cheng. A Term Dependency-based Approach for Query Terms Ranking. In *Proc. of the* 18th *ACM Conf. on Information and Knowledge Management (CIKM)*, pages 1267–1276, New York, NY, USA, 2009a. ACM.

Chia-Jung Lee, Yi-Chun Lin, Ruey-Cheng Chen, and Pu-Jen Cheng. Selecting Effective Terms for Query Formulation. In *Proc. of the* 5th *Asia Information Retrieval Symposium on Information Retrieval Technology (AIRS)*, pages 168–180, Berlin, Heidelberg, 2009b. Springer-Verlag.

Jin Ha Lee. Analysis of User Needs and Information Features in Natural Language Queries seeking Music Information. *Journal of the American Society for Information Science and Technology (JASIST)*, 61(5):1025–1045, 2010.

Chee Wee Leong and Silviu Cucerzan. Supporting Factual Statements with Evidence from the Web. In *Proc. of the* 21st *ACM Intl. Conf. on Information and Knowledge Management (CIKM)*, pages 1153–1162, New York, NY, USA, 2012. ACM.

Yunyao Li, Huahai Yang, and HV Jagadish. Constructing a Generic Natural Language Interface for an XML Database. In *Proc. of the 2006 Conf. on Advances in Database Technology (EDBT)*, pages 737–754, Berlin, Heidelberg, 2006. Springer-Verlag.

Christina Lioma and Iadh Ounis. A Syntactically-based Query Reformulation Technique for Information Retrieval. *Information processing & management (IPM)*, 44(1):143–162, 2008.

K Tamsin Maxwell and W Bruce Croft. Compact Query Term Selection using Topically Related Text. In *Proc. of the* 36th *Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR)*, pages 583–592, New York, NY, USA, 2013. ACM.

Donald Metzler and W. Bruce Croft. A Markov Random Field Model for Term Dependencies. In *Proc. of the* 28th *Annual Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR)*, pages 472–479, New York, NY, USA, 2005. ACM.

Nikita Mishra, Rishiraj Saha Roy, Niloy Ganguly, Srivatsan Laxman, and Monojit Choudhury. Unsupervised Query Segmentation using Only Query Logs. In *Proc. of the* 20th *Intl. Conf. on World Wide Web (WWW)*, pages 91–92, New York, NY, USA, 2011. ACM.

Liqiang Nie, Shuicheng Yan, Meng Wang, Richang Hong, and Tat-Seng Chua. Harvesting visual concepts for image search with complex queries. In *Proceedings of the 20th ACM International Conference on Multimedia*, MM '12, pages 59–68, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1089-5. . URL `http://doi.acm.org/10.1145/2393347.2393363`.

Daan Odijk, Edgar Meij, Isaac Sijaranamual, and Maarten de Rijke. Dynamic Query Modeling for Related Content Finding. In *Proc. of the* 38th *Annual Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR)*. ACM, 2015.

Jiaul H. Paik and Douglas W. Oard. A Fixed-Point Method for Weighting Terms in Verbose Informational Queries. In *Proc. of the* 23rd *ACM Conf. on Information and Knowledge Management (CIKM)*, pages 131–140, New York, NY, USA, 2014. ACM.

Nish Parikh, Prasad Sriram, and Mohammad Al Hasan. On Segmentation of E-Commerce Queries. In *Proc. of the* 22nd *ACM Intl. Conf. on Information and Knowledge Management (CIKM)*, pages 1137–1146, New York, NY, USA, 2013. ACM.

Jae Hyun Park and W Bruce Croft. Query Term Ranking based on Dependency Parsing of Verbose Queries. In *Proc. of the* 33rd *Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR)*, pages 829–830, New York, NY, USA, 2010. ACM.

Jae Hyun Park, W Bruce Croft, and David A Smith. A Quasi-Synchronous Dependence Model for Information Retrieval. In *Proc. of the* 20th *ACM Intl. Conf. on Information and Knowledge Management (CIKM)*, pages 17–26, New York, NY, USA, 2011. ACM.

Fuchun Peng, Nawaaz Ahmed, Xin Li, and Yumao Lu. Context Sensitive Stemming for Web Search. In *Proc. of the* 30th *Annual Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR)*, pages 639–646, New York, NY, USA, 2007. ACM.

Nina Phan, Peter Bailey, and Ross Wilkinson. Understanding the Relationship of Information Need Specificity to Search Query Length. In *Proc. of the* 30th *Annual Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR)*, pages 709–710, New York, NY, USA, 2007. ACM.

Jay M Ponte and W Bruce Croft. A Language Modeling Approach to Information Retrieval. In *Proc. of the* 21st *Annual Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR)*, pages 275–281, New York, NY, USA, 1998. ACM.

Knut Magne Risvik, Tomasz Mikolajewski, and Peter Boros. Query Segmentation for Web Search. In *Proc. of the* 12th *Intl. Conf. on World Wide Web (WWW)*, New York, NY, USA, 2003. ACM.

Daniel Sheldon, Milad Shokouhi, Martin Szummer, and Nick Craswell. LambdaMerge: Merging the Results of Query Reformulations. In *Proc. of the* 4th *ACM Intl. Conf. on Web Search and Data Mining (WSDM)*, pages 795–804, New York, NY, USA, 2011. ACM.

Gyanit Singh, Nish Parikh, and Neel Sundaresan. Rewriting Null E-Commerce Queries to Recommend Products. In *Proc. of the* 21st *Intl. Conf. Companion on World Wide Web (WWW)*, pages 73–82, New York, NY, USA, 2012. ACM.

Mark D Smucker and James Allan. Lightening the Load of Document Smoothing for Better Language Modeling Retrieval. In *Proc. of the* 29th *Annual Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR)*, pages 699–700. ACM, 2006.

Richard Socher, Cliff C. Lin, Andrew Y. Ng, and Christopher D. Manning. Parsing Natural Scenes and Natural Language with Recursive Neural Networks. In *Proc. of the* 26th *Intl. Conf. on Machine Learning (ICML)*, 2011.

Charles Sutton and Andrew McCallum. Introduction to Conditional Random Fields for Relational Learning. In Lise Getoor and Ben Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press, 2006.

Krysta M. Svore, Pallika H. Kanani, and Nazan Khan. How Good is a Span of Terms?: Exploiting Proximity to Improve Web Retrieval. In *Proc. of the* 33*rd* *Annual Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR)*, pages 154–161, New York, NY, USA, 2010. ACM.

Bin Tan and Fuchun Peng. Unsupervised Query Segmentation using Generative Language Models and Wikipedia. In *Proc. of the* 17*th* *Intl. Conf. on World Wide Web (WWW)*, pages 347–356, New York, NY, USA, 2008. ACM.

Sergio D. Torres, Djoerd Hiemstra, and Pavel Serdyukov. An Analysis of Queries Intended to Search Information for Children. In *Proc. of the* 3*rd* *Symposium on Information Interaction in Context (IIiX)*, pages 235–244, New York, NY, USA, 2010. ACM.

Manos Tsagkias, Maarten De Rijke, and Wouter Weerkamp. Hypergeometric Language Models for Republished Article Finding. In *Proc. of the* 34*th* *Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR)*, pages 485–494. ACM, 2011.

Xuanhui Wang and ChengXiang Zhai. Mining Term Association Patterns from Search Logs for Effective Query Reformulation. In *Proc. of the* 17*th* *ACM Conf. on Information and Knowledge Management (CIKM)*, pages 479–488, New York, NY, USA, 2008. ACM.

Xiaobing Xue and W Bruce Croft. Modeling Subset Distributions for Verbose Queries. In *Proc. of the* 34*th* *Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR)*, pages 1133–1134, New York, NY, USA, 2011. ACM.

Xiaobing Xue and W Bruce Croft. Generating Reformulation Trees for Complex Queries. In *Proc. of the* 35*th* *Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR)*, pages 525–534, New York, NY, USA, 2012. ACM.

Xiaobing Xue, Jiwoon Jeon, and W. Bruce Croft. Retrieval Models for Question and Answer Archives. In *Proc. of the* 31*st* *Annual Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR)*, pages 475–482, New York, NY, USA, 2008. ACM.

Xiaobing Xue, Samuel Huston, and W Bruce Croft. Improving Verbose Queries using Subset Distribution. In *Proc. of the* 19th *ACM Intl. Conf. on Information and Knowledge Management (CIKM)*, pages 1059–1068, New York, NY, USA, 2010. ACM.

Xiaobing Xue, Yu Tao, Daxin Jiang, and Hang Li. Automatically Mining Question Reformulation Patterns from Search Log Data. In *Proc. of the* 50th *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 187–192, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

Bishan Yang, Nish Parikh, Gyanit Singh, and Neel Sundaresan. A Study of Query Term Deletion Using Large-Scale E-commerce Search Logs. In *Proc. of the* 36th *European Conf. on Information Retrieval (ECIR)*, pages 235–246, Berlin, Heidelberg, 2014. Springer-Verlag.

Yin Yang, Nilesh Bansal, Wisam Dakka, Panagiotis Ipeirotis, Nick Koudas, and Dimitris Papadias. Query by Document. In *Proc. of the* 2nd *ACM Intl. Conf. on Web Search and Data Mining (WSDM)*, pages 34–43, New York, NY, USA, 2009. ACM.

Jeonghe Yi and Farzin Maghoul. Mobile Search Pattern Evolution: The Trend and the Impact of Voice Queries. In *Proc. of the* 20th *Intl. Conf. Companion on World Wide Web (WWW)*, pages 165–166, New York, NY, USA, 2011. ACM.

Wen-tau Yih, Joshua Goodman, and Vitor R Carvalho. Finding Advertising Keywords on Web Pages. In *Proc. of the* 15th *Intl. Conf. on World Wide Web (WWW)*, pages 213–222, New York, NY, USA, 2006. ACM.

Chengxiang Zhai and John Lafferty. A Study of Smoothing Methods for Language Models applied to Ad hoc Information Retrieval. In *Proc. of the* 24th *Annual Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR)*, pages 334–342, New York, NY, USA, 2001. ACM.

Chengxiang Zhai and John Lafferty. A Study of Smoothing Methods for Language Models applied to Information Retrieval. *ACM Transactions on Information Systems (TOIS)*, 22(2):179–214, 2004.

Le Zhao and Jamie Callan. Term Necessity Prediction. In *Proc. of the* 19th *ACM Intl. Conf. on Information and Knowledge Management (CIKM)*, pages 259–268, New York, NY, USA, 2010. ACM.

Index