Improving Search Quality of the Google Search Appliance

by

Huy Nguyen

Submitted to the Department of Electrical Engineering and Computer Science

in partial fulfillment of the requirements for the degree of

Master of Engineering in Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

[June]

May 2009

Author . . . [illegible] . . . .
Department of Electrical Engineering and Computer Science
May 22, 2009

Certified by . . . .
David Elworthy
~~VI-A Company Thesis Supervisor~~
Thesis Supervisor

Certified by . . . .
Regina Barzilay
Associate Professor
Thesis Supervisor

Accepted by . . . .
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Improving Search Quality of the Google Search Appliance

by

Huy Nguyen

Submitted to the Department of Electrical Engineering and Computer Science
on May 22, 2009, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Computer Science

Abstract

In this thesis, we describe various experiments on the ranking function of the Google Search Appliance to improve search quality. An evolutionary computation framework is implemented and applied to optimize various parameter settings of the ranking function. We evaluate the importance of IDF in the ranking function and achieve small improvements in performance. We also examine many ways to combining the query-independent and query-dependent scores. Lastly, we perform various experiments with signals based on the positions of the query terms in the document.

Thesis Supervisor: David Elworthy
Title: VI-A Company Thesis Supervisor

Thesis Supervisor: Regina Barzilay
Title: Associate Professor

Acknowledgments

I am deeply grateful to David Elworthy and Professor Regina Barzilay for providing me with directions and ideas throughout my work in information retrieval. I also want to thank Liviu Panait, Michael Parker, Sibabrata Ray, Emily Rocke, Nikola Jevtic and the rest of the enterprise search team at Google, who have been very helpful and supportive during my entire time at Google.

Contents

List of Figures

Chapter 1

Introduction

1.1 Ranking in web and enterprise search

In the past few decades, there has been an explosion in the amount of content available on the World Wide Web (WWW). Navigating on the web and finding necessary information have become extremely important tasks as the web grows and the number of web pages a person needs to use quickly grows out of the manageable range of a person's memory. Starting from directory services provided by companies such as Yahoo, the task was handed over to automated search engines by Google, Yahoo, Microsoft, etc. As people get used to search tools available on the web, they also look for similar ways to find information in private repositories such as companies' private documents. In some sense, these repositories are similar to the public web as companies put their documents on internal websites with a link structure among documents just like on the web. In some other ways, however, private repositories are much more heterogeneous than the web with a lot of documents stored in databases and in local files without any link between them.

The link structure of private corpora is different from that of the web in many respects. Unlike in the web, there is no artificial effort to boost the ranking of certain documents so there are much fewer spammy links in these corpora. On the other hand, in web search, links from a different website usually give a better indication of popularity than links from the same website. In private corpora, such distinction usually does not exist as documents are spread out on many different machines in some arbitrary manner and sometimes on-site links are actually equally important as off-site links. These and other structural differences make searching in an enterprise setting an interesting problem, which requires re-evaluation of the contributions and relative importance of link-based signals and document-based signals and the ways to combine them into a final score for the ranking function. Most importantly the way the signals are combined should work well across many different types of corpora with different link structure characteristics and sizes.

As companies put more and more documents into searchable formats, an interesting challenge arises. The private corpora have grown quickly beyond the scope one single machine can handle and require being divided into multiple parts to be handled by multiple machines. This is known as the federated search problem, where multiple machines serve multiple parts of the same corpus with the ability to combine the search results quickly and accurately. To reduce communication cost, improve latency and ease maintenance and extensibility, it is desirable for each machine to be able to answer queries while not knowing about the rest of the corpus. Therefore, signals based on the whole corpus becomes less preferable to alternatives with similar performance but easier to maintain. One such signal is the inverse document frequency (IDF), which is based on the number of document in the corpus containing each search term. When documents are spread out on many machines, IDF can be skewed toward certain parts of the corpus, so the system incorrectly favors results from those particular parts of the corpus.

Generally, a search ranking function computes the scores based on the web structure of the whole corpus and the matches between the content of the documents and the query terms. For the matches between the content of the documents and the query terms, existing systems overwhelmingly use the "bag-of-words" model with the frequency of occurrences as the sole measure of relevancy. Recently, Troy and Zhang [37] came up with a new signal called Chronological Term Rank (CTR) based on the position of the occurrences in the document and achieved some improvements on a simple system tested on several TREC tasks. This is a promising approach as it is based on information orthogonal to the traditional term frequencies and it is worth investigating further on more sophisticated systems on many different query types.

1.2 Our contribution

In this thesis, we attempt to address the three problems described in the previous section for the ranking function of the Google Search Appliance by re-evaluating the contributions of various signals and the ways to combine them in the ranking function. Intuitively almost all signals by themselves have some discriminatory information on the relevancy of a document against a query. However, it is non-trivial to combine them fairly with respect to their contributions and prevent noisy signals from hampering the accurate ones. For this purpose, we have implemented an evolutionary strategies framework that can evaluate many parameter settings in parallel and given a particular ranking function, optimize the parameter settings for that function.

Effect of IDF on the ranking function We evaluate the impact of the IDF on the ranking function by comparing the performance of the system with and without IDF on various TREC corpora and also several side-by-side experiments. While the IDF is a good way to compare the relative importance of the query terms, it can be noisy, especially when the corpus is split into multiple separate parts e.g. in federated search. Additionally, the contribution from the IDF is reduced for the AND queries where all query terms must be present for a document to be considered: the system does not have to choose between documents with only occurrences of the one query term and documents with only occurrences of another query term. The contribution from the IDF can also be achieved partially with a list of stop words i.e. words providing little discriminatory information such as "an", "and", "in", etc. It turns out that the contribution from IDF is marginal at best on medium size corpora of a million documents and somewhat harmful on small corpora. We achieve a modest improvement on a small task while maintaining the same level of performance on bigger corpora.

Methods for combining different signals We look at the relative importance of the web-based score and the content-based score and the ways to combine these scores into a final score. We explore the use of impact transformation [2] in combining them as well as various variations with cutoffs and varying behavior on different parameter ranges. Interestingly, simple linear combination functions work almost as well as more complicated quadratic ones with varying behavior on different parameter ranges even though there are a small number of queries where quadratic functions work slightly better.

Impact of position-based signals We study the impact of position-based signals including the CTR and the length of the shortest snippet containing all the search terms. Previously, the impacts of these signals were studied on a simple system where the way the occurrences are formatted is not taken into account i.e an occurrence in a big font is considered the same as an occurrence in a small font. In this thesis, we explore the contributions of these signals in the context of AND queries on the Google Search Appliance, which has a sophisticated weighting scheme taking into account the formatting of the occurrences. Our experiment results are mixed with improved performance on some tasks but worse performance on other tasks. We also compute upper bounds on the contributions of these signals. These bounds indicate that while they might not help in general, it is possible that they can provide some indication of relevance for home page queries.

1.3 Structure of the thesis

In chapter 2, we review the structure of a web search system and provide a broad overview of the major approaches to the web search problem. Our discussion is mostly around the ranking component as it is arguably the heart of the system and also the focus of our research. We then describe the various signals going into the ranking function and the three major approaches of combining those signals into a final score. Finally, we discuss our corpora, the evaluation metrics, and the relative importance of the metrics on different corpora.

Chapter 3 presents our system for running experiments and selecting weights using evolutionary strategies. We describe our setup for parallelized evaluation and our use of evolutionary strategies to optimize the parameter settings.

Chapter 4 investigates the impact of IDF in the topicality score. In this chapter, we compare the performance of two settings: one optimized for combining with IDF, and one optimized for not using IDF, on our evaluation corpora. Besides the overall metrics, we also look at individual queries where one setting performs better than the other and vice versa and the reason why it happens.

Chapter 5 looks at the use of impact transformation and various combining functions for combining link structure measures and content-based scores. We start by describing our experiment and evaluation setup. We then describe our findings by comparing the behavior of the scores and the optimal combining function in different parameter ranges.

Chapter 6 studies the effect of position-based signals including the first hit position and the length of the shortest snippet containing all the query terms. We present our experiment results using these signals with and without using IDF. Then we discuss the optimal reranking using these signals, which is an upper bound for the improvement gain of any method combining the existing score and the position-based signals.

In chapter 7, we discuss the main ideas and contributions of the thesis and directions for future research.

Chapter 2

Related Work

In this chapter, we describe the organization of a generic web search system and the related works in search ranking. Section 2.1 describes the components of a search system: the crawler, the indexer, and the query processor. Then we give an overview of search ranking, and the various signals for ranking in section 2.2. In section 2.3, we go over several ranking signals based on the content of the documents. In section 2.4, we describe the signals based on the links between documents. Then in section 2.5, we go over three main models in search ranking: the vector space model, the probabilistic ranking model, and the language mixture model. Section 2.6 describes the ways to combine different signals into a single final score. Finally in section 2.7, we describe our evaluation metrics, and our evaluation corpora and query sets.

2.1 A web search system

A web search system is a system for organizing and retrieving information using text based queries. The user provides the system with a query consisting of several words, and the system has to return a list of documents ordered by how relevant they are with respect to the query. In big corpora of documents, there could be thousands (or even hundred thousands) of documents matching a particular query. Therefore, the order in which the documents are presented to the user is crucial to the usefulness of the system. The part producing the ordering is the main focus of this thesis.

Firstly, we look at the general organization of a web search system, and in particular, the Google Search Appliance. Generally, a search system is a complex system with many components divided into three parts: a crawler, an indexer, and a query processor. The crawler is used to retrieve all documents in the corpora and prepare them to be indexed. Usually the crawler is given several starting points and then follows the links on the web pages to find the rest of the documents. In an enterprise setting, the crawler also needs to be able to extract documents from databases and files on hard drives. In the Google Search Appliance, there is a connector framework allowing the crawler to read documents from databases and files. The indexer reorganizes information extracted from the documents to produce search indices, which allows finding relevant documents quickly. The query processor takes queries from users and finds matching documents using the indices produced by the indexer. It also computes the relevancy of the matching documents and presents to the users the documents sorted in decreasing relevancy order. In the subsequent sections, we will explore these components in more details.

2.1.1 The crawler

The crawler explores the documents on the web through web links. It is given a set of seed pages and gather the rest of the documents by following the hyperlinks in the web graph. It usually maintains a priority queue of pages, which determines the order they are to be downloaded [8]. Page priorities are determined by some function based on the importance of the document in the web graph and also parameters provided by the user. For example, on the Google Search Appliance, the user can set how often they want particular documents to be re-crawled. Once a page is downloaded, the crawler has to send its content to the indexer to extract the links and insert the links into the priority queue if needed. To maintain the cached content of the documents up to date, downloaded document are periodically reinserted into the priority queue to be re-downloaded. The rate at which a document changes in the past can also be factored into its priority. In an enterprise setting, a crawler also need special connectors to extract documents from databases and files.

2.1.2 The indexer

The indexer parses the information contained in each web page and reorganizes it in a format amenable to searching. Typically, the page is broken down to individual tokens. Each token can be a term and sometimes a phrase. The indexer creates an index for each token, called a *posting list*, containing all the IDs (*postings*) of documents with that token. The IDs are stored in increasing order. In the record of a document in the posting list, all occurrences of the token are also listed in increasing order. There are two ways to organize the posting lists. The first way is to store information about each document all in the same place. This organization has an advantage of allowing scoring a document after reading the posting list exactly once. The second way is to store all the IDs together and the secondary fields later. This organization allows reading the IDs quickly, which makes several operations on posting lists faster.

Beside retrieving all documents containing a particular query term, posting lists must also support more sophisticated operations. In many cases, the user looks for documents not containing only one query term but many of them. These queries require computing the intersection of many posting lists, which can be computed quickly when the lists are sorted. Storing the IDs of the documents separately from the secondary fields also makes this operation faster because the list reader only has to read the IDs instead of everything. In the case of phrase queries (e.g. a user looks for the phrase "computer science" but not individual words "computer" and "science"), we also need to compute the "intersection" of posting lists at the single document level.

Beside the posting lists, there is a dictionary of all the distinct tokens pointing to the corresponding posting lists. Statistics about each term (e.g. number of occurrences, inverse document frequency, etc) are also maintained to facilitate scoring. Additionally, the indexer also takes the links it extracted from the pages and feeds back to the crawler so that new pages can be explored.

2.1.3 The query processor

The query processor parses the users' queries and matches them against the documents. In this thesis, we are only concerned with the non-iterative model, where there is no additional feedback or refinement from the user after the query is answered. This is the situation with web search system without cookies or mechanisms to remember user's past actions.

A query is usually parsed as follows. In languages with no apparent word boundary like Chinese, the query needs to be broken down to terms. Then, typically, the query is expanded by replacing each term with a disjunction of several synonyms. Any web page matching the query (or the expanded query) is then scored and the list of relevant documents are returned to the user in the order from most relevant to least relevant.

There are many possible ways to answer a particular query, each with different advantages and disadvantages. Typically in the literature (e.g. see [2]), all documents containing at least one query term are considered. This approach has an advantage of getting high recall rate because even if all but one of the query term does not occur in the relevant document, the system can still find that document. On the other hand, since it considers a lot of candidate documents, the precision is clearly affected. The other way is to only consider documents containing all the query terms (or their synnonyms). In this setting, the trade-off between recall and precision is controlled by the aggressiveness of the query expansion component. If the queries are expanded a lot with many synonyms, the system might still find the relevant documents in many cases but it might run into a problem with precision just like in the other approach. If the queries are expanded cautiously, sometimes the system might miss out on the relevant documents but the precision is much improved in many other queries. In this thesis, we are only concerned with the second strategy i.e. only documents containing all the query terms are retrieved and ranked. This is the default setting on the Google Search Appliance and many web search engines e.g. Google, Yahoo, Live, etc but they all support the other setting as well.

2.2 Document ranking

A ranking function determining the relevancy of a document to a query is the heart of the query processor. As explained in the previous section, it determines the usefulness of the whole system in big corpora where there are many documents matching the given query. Typically a ranking function computes the score of a document based on both signals computed from the document's content (*page-dependent signals*) and signals computed from the content of other documents or the structure of the web graph(*page-independent signals*). Examples of page-dependent signals are how often a query term occurs in the document, where the first occurrence of a query term is in the document, etc. Examples of page-independent signals are the pagerank of a page, the number of hyperlinks to a page, etc. We will describe these signals in more details in the subsequent sections.

2.3 Page-dependent signals

In this section, we describe the signals computed from the content of the document in consideration. The score of a document is usually computed based on the occurrences of the query terms in the document as well as in the whole corpus. An occurrence of a query term is not strictly the exact same word as appeared in the query but it can possibly be synnonyms or other related words. For example, the word "computers" can be counted as an occurrence for "computer". One way for expanding a query term to related words is to reduce words to their linguistic stems using e.g. the Porter stemmer [28] and compare the stems of the words in the document with the stems of the words in the query. For example, the words "cut" and "cutting" share the same stem of "cut" and can be treated as the same word from the point of view of the ranking function.

A popular way for the ranking functions to handle documents is to model them as "bags of words" i.e. the order of the words in a document is ignored and every word occurs independently in the document according to some hidden distribution. For example, there is no distinction between the documents “test system” and “system test”. The advantage of the “bag of words” model is the simplicity of the model, allowing the system to build compact indices and answer queries quickly. In this section, we describe the traditional “bag of words” model with improvements taking into account short phrases, document formatting, and term positions.

2.3.1 Weighted hit counts

A *hit* is an occurrence of a word in a document. Intuitively, the way the word is formatted (e.g. bold, big font, etc) provides hints about how relevant it is to the content of the page. Therefore, hits are generally weighted depending on their formatting. There are several types of text in a typical web page with different levels of importance.

- Title/Heading text: Hits in the title of the document are generally the most important ones since they are supposed to capture the meaning of the whole document. The heading text are also important as they usually capture the meaning of a significant part of the document.

- Alternative text for visual objects: This text is supposed to convey the same information as the visual objects on machine without the required graphics capabilities.

- Formatted text: Hits in bigger fonts are also more important than hits in smaller fonts. There are many ways to determine what constitute bigger fonts and what constitute smaller fonts. They could be constants set throughout the corpora or they could be normalized for each document individually (e.g. only 10% of the largest text in a document can be considered big). Bold or italic text also deserves higher weights as it usually contains important information.

- URL: URL is an important source of information. Intuitively, if a word occurs in the URL, it usually contains important information about the content of the whole page. In addition to hits, the URL also contains information about the importance of the document. Top level documents (and hence, most important) usually have short URL so the length of the URL is also a signal about the importance of the page regardless of the query.

- Link text: In some sense, link text is not quite the content of the document containing it because it is supposed to describe the document it links to instead. Therefore, link text usually gets a lower weight than the rest of the text.

2.3.2 Phrases

One weakness of the "bag of words" model is that it does not take into account the relative location of the occurrences of the search terms in the documents. Counting occurrences of phrases is a way to partially rectify this problem. There are many different types of phrases with possibly different scoring scheme. The most common type is bi-grams of words occurring in both the document and the query. Another kind of phrases is the bi-gram of two query terms swapped in the document. More generally, phrases can also consist of pairs of query terms occurring near each other but not immediately next to each other.

There is another advantage of counting phrases, especially when the inverse document frequency (IDF) is not available. Sometimes in the query, there are terms that are not as discriminative as other terms because they appear in a lot of documents. However, phrases are weighted much more than single word, so documents containing the non-discriminating terms in the proximity of discriminating terms will get higher scores than documents with these terms in separate places.

Like single word occurrences, a phrase gets higher weights if it appears in more important text such as the title, the section headings, etc.

2.3.3 Term position signals

There has been a lot of work on incorporating term positions, especially the proximity information of the terms, into the "bag of words" model to improve the retrieval effectiveness [13, 20, 29, 6]. Recently, Troy and Zhang [37] propose a signal called chronological term rank (CTR). The CTR of a term is defined as the earliest position the term occurs in the document. In the rest of the thesis, we will use the phrases chronological term rank and the first hit position interchangeably.

In addition to the CTR, we also look at another signal based on the relative position of the query terms in the document. Define the shortest snippet length to be the shortest contiguous sub-sequence of the document containing all the query terms. Intuitively, when the query terms occur in proximity of each other, it is more likely that the document containing them is relevant.

These signals both provide evidents on the relevancy of a document with respect to a query. However, it is not clear if they can provide enough orthogonal information compared to existing signals without adding much noise to the ranking function. In this thesis, we attempt to shed some light on this issue.

2.4 Page-independent signals

The relevancy of a document can be determined not only by information within that page but also by information in other pages. This source of information is extremely important as it is orthogonal to the information presented on the document. It is controlled by a different entity from the owner of the document and therefore, is less susceptible to spamming.

There are two major sources of information. One of them is the hits in the anchor text. If a document is pointed to from another document using the query term as the anchor text, it can be considered as a recommendation of that other document for the document we are considering. The other source of information is the structure of the web graph independent of the query. In this section, we briefly describe some of the measurements based on the link structure.

2.4.1 In-degree

The number of in-links to a page gives some indication about the page's importance. This measure has the advantage of being simple and easy to compute and update.

On the Internet, there are a lot of spammy links, which might reduce the usefulness of this measure. However, in an enterprise setting, it could be helpful as the number of spammy links is much smaller.

2.4.2 HITS

HITS is a link analysis algorithm by Jon Kleinberg [21] for finding authoritative sources of a given topic. The algorithm is designed to be built on top of a crude retrieval system. It relies on the retrieval system to find a small subset of documents containing most of the relevant pages for the given topic. Then all the web pages linked from the subset and some web pages linking to the subset are added to the subset to form a collection of web pages directly or indirectly related to the topic. The algorithm then uses the links between these web pages to determine the most relevant pages.

A web page can be relevant to a topic in two ways: either it contains a lot of information about the topic or it links to a lot of informative pages. The first kind of page is called an authoritative page. The second kind of page is called a hub page. Intuitively, an authoritative page is linked to from many hub pages, and each hub page links to many authoritative pages. The HITS algorithm exploits these self-enforcing relationships to determine two measures for each web page: a hub score determining whether the page is a good hub, and a authority score determining whether the page is a good authoritative page. These two measures have a mutual recursion relationship and can be computed using an iterative algorithm. Let x_p be the authority score of the page p and y_p be the hub score of the page p. Let $G = (V, E)$ be the web graph of the pages in consideration. The relationship between the two scores can be expressed by the following equations.

$$x_p \leftarrow \sum_{q:(q,p)\in E} y_q$$

$$y_p \leftarrow \sum_{q:(p,q)\in E} x_q$$

Approximations for x_p and y_p can be computed in an iterative way: firstly x and y are initialized to arbitrary vectors, and then new approximations are computed based on previous approximations repeatedly until no significant change occurs. The details are shown in algorithm 1.

```
$n \leftarrow$number of linked pages
$x_0 \leftarrow (1, \ldots, 1)$
$y_0 \leftarrow (1, \ldots, 1)$
$k \leftarrow 0$
**repeat**
    $k \leftarrow k + 1$
    **for** $p \in \{1, \ldots, n\}$ **do**
        $x_{k,p} \leftarrow \sum_{q:(q,p) \in E} y_{k-1,q}$
        $y_{k,p} \leftarrow \sum_{q:(p,q) \in E} x_{k,q}$
    **end**
    Normalize $x_k$ and $y_k$ so that $||x_k||_2 = ||y_k||_2 = 1$
**until** $||x_k - x_{k-1}||_1 < \epsilon$ *and* $||y_k - y_{k-1}||_1 < \epsilon$
**return** $x_i, y_i$
```

Algorithm 1: Algorithm for approximating authority and hub scores

HITS is, unfortunately, too expensive to be computed at query time, and hence, cannot be used directly in answering queries.

Bharat and Henzinger [3] pointed out several problems with the original HITS algorithm and suggested several improvements. Lempel and Moran [22] proposed a stochastic algorithm using similar ideas to HITS but is less vulnerable to the Tightly Knit Community (TKC) Effect.

2.4.3 Pagerank

Perhaps the most well-known link-based signal, pagerank is a measure of page importance by Brin and Page [4, 26] based on the global structure of the web graph. It models the behavior of a random web surfer, who starts from a random web page and keeps following links randomly on the web graph. At any point in time, the random web surfer can choose to restart from a random web page with probability p and follow links randomly again from there. Brin and Page found out that $p = 0.85$ gives the optimal performance on their system. The sum of Pagerank over all web pages is normalized to 1 so Pagerank models a probability distribution. In fact, Pagerank of a page is the probability of the random surfer visiting it at a given time.

Let PR_i be the Pagerank of page i, E be the adjacency matrix of the web graph, and b be the bookmark vector where b_i represents the probability of starting/restarting at page i in the surfing process. If all web page are equally likely to be the starting place, $b_i = \frac{1}{n}\ \forall i$ where n is the number of web pages. In a web search system, there is usually a list of starting pages for crawling, from which the rest of the web pages are discovered by following links. These pages can also be used as the set of starting pages for the random surfer. In this case, if there are k starting pages, $b_i = 1/k$ if page i is one of those, and 0 if it is not. Pagerank is the stationary distribution of the Markov chain modeling a random walk on the web graph with random jump and it satisfies the following equation.

$$PR_i = (1-p)b_i + \sum_j \frac{pPR_jE_{ji}}{\sum_k E_{jk}}$$

There are common cases where Pagerank does not converge, however. For example, when some document does not have any link, there is no stationary distribution. Intuitively, the Pageranks of the documents with no links are not circulated to any other page and therefore, are lost. This problem can be fixed by forcing a jump to a random bookmark after visiting a document with no links.

When Pagerank exists, it can be approximated using an iterative algorithm.

```
Initialize $P_0$
$k \leftarrow 0$
repeat
    $k \leftarrow k+1$
    for $i \in \{1,\dots,n\}$ do
        $P_{k,i} \leftarrow (1-p)b_i + \sum_j \frac{pP_{k-1,j}E_{ji}}{\sum_k E_{jk}}$
    end
    Update $P_k$ with jumps from pages with no links
until $||P_k - P_{k-1}||_1 < \epsilon$
```

Algorithm 2: Algorithm for approximating Pagerank

Intuitively, the above algorithm iteratively computes approximations of Pagerank based on previous approximations and stops when the approximations does not change by a significant amount.

There are also variants of Pagerank with different bookmark vectors. Haveliwala [12] proposes a version of Pagerank biased towards a given topic (*personalized Pagerank*) using a pre-computed bookmark vector tailored to the topic (which can be obtained from directory services e.g. DMOZ). Kamvar et al. [19], and Jeh and Widom [15] propose several algorithms for computing personalized Pagerank quickly. There is also a lot of work on improving the computation of Pagerank [1, 11] and generalizing Pagerank [7, 36].

In this thesis, we use a variant of Pagerank to compute the importance of a page regardless of the query.

2.5 Ranking models

In this section, we describe the three major models for combining various signals to produce a similarity score between a query and a document. On the Google Search Appliance, we only work in the vector space model but for completeness, we will briefly describe the other models as well.

2.5.1 Vector space model

In this model, a document d and a query q are viewed as high dimension vectors where each coordinate corresponds to a term t in the document and the query. The assumption here is that the relevance of a document to a query is correlated with the dot product, or the angle, of these two vectors. More precisely, each coordinate of d or q usually consists of three components: *term frequency* $TF_{d,t}$ based on the number of occurrences of the term t (or a weighted sum of the occurrences from various parts of the document), *inverted document frequency* IDF_t [18] based on the number of documents containing the term t, and *document length normalization* DLN_d based on the length of the document d. Intuitively, $TF_{d,t}$ represents the fact that the relevance score is higher if t occurs many times in the document d. IDF_t

measures how important a term is compared to the other query term. For example, a common word like "the" occurs in most (if not all) documents in the corpus and therefore, provides less discriminative information than the words occurring in only a handful of them. DLN_d is used to normalize the relevant score based on the length of the document. Intuitively, longer documents tend to contain the query terms many more times than shorter documents. Originally, DLN_d is set to be the length $||d||_2$ of the vector d. However, Singhal et al. [35] observe that this formulation unfairly discriminates against long documents. They introduce the pivoted DLN_d to account for this fact and significantly improve the retrieval accuracy. Specifically, let W_d be the length of the vector d and $W^{(a)}$ be the average length of all document vectors. The pivoted DLN_d is computed as follows.

$$DLN_d = \frac{1}{1 - s + s \cdot W_d / W^{(a)}}$$

s is a constant with typical value of 0.7 [35].

The relevancy of a document to a query can be viewed as a dot product of the two vectors d and q.

$$Score = \sum_{t \in d \cap q} TF_{d,t} \cdot IDF_t \cdot DLN_d$$

There are many different formulas for the $TF_{d,t}$ and IDF_t components in the above formula. The simplest formula for $TF_{d,t}$ is the number of times the term t occurs in the document d. The problem with this version of $TF_{d,t}$ is that the discriminatory information decreases as the number of occurrences gets larger. For example, if a query term occurs 5 times in the first document and once in the second document, the first document is obviously much more likely to be relevant. However, if a query term occurs 100 times in the first document and 96 times in the second document, it is not clear which document is more relevant. Therefore, a desirable feature of the formula for $TF_{d,t}$ is that it varies a lot when the frequency is small and changes very little when the frequency is large. For example, one popular formula for the TF is $1 + \ln f_{d,t}$ [2] where $f_{d,t}$ is the number of occurrences of the term t in the document d.

Let f_t be the number of documents containing t and f^m be the greatest f_t over all t. A formula for IDF_t is $\ln(1 + f^m/f_t)$ [2]. Combining all these components, a possible formula for the vector space score is as follows.

$$Score = \sum_{t \in d \cap q} \frac{(1 + \ln f_{d,t}) \cdot \ln(1 + f^m/f_t)}{1 - s + s \cdot W_d/W^{(a)}}$$

Even though this model is the simplest of all three, it performs extremely well in practice and achieves good query time. In this thesis, we will only work in the vector space model.

2.5.2 Probabilistic ranking model

This model attempts to solve the retrieval problem based on theoretical framework from information theory and probability theory. This model was proposed by Maron and Kuhns [23], and improved by Robertson and Sparck Jones [32]. The retrieval systems following this model determine whether a document is relevant or not based on estimating the probability it is relevant given the content of the document and the query. Let $P[rel = 1|d, q]$ be the probability that document d is relevant for query q given their content and $P[rel = 0|d, q]$ be the probability that d is not relevant for q given their content. A document is considered relevant if $P[rel = 1|d, q] > P[rel = 0|d, q]$. When a ranking is needed, the system can just sort the documents in the decreasing order of their probabilities of being relevant.

In the early work in this model, the following Binary Independence Model is assumed.

- The relevance of a document is independent of the relevance of other documents. This assumption is, unfortunately, sometimes incorrect e.g. when there are nearly identical documents in the corpus.

- The terms occur independently in a document. Whether a term occurs in a document or not does not affect the probability of another occurring in the same document.

- The order the terms occurring in the document does not matter. In other words, a document can be viewed as a vector where each coordinate is the number of times a term occurs in it.

With these assumptions, the relevant probabilities can be estimated easily.

The state of the art retrieval systems using this model are mostly based on Okapi BM25 by Robertson et al. [33]. Robertson and Walker [30] use approximations for the 2-Poisson model to come up with formulas for $TF_{d,t}$, IDF_t, and $TF_{q,t}$. Jones et al. [16, 17] suggest several ranking functions for different query lengths. For short queries, which is our primary concern in this thesis, the score of a document is computed as follows.

$$Score = \sum_{t \in q} \ln(N/f_t) \frac{(k_1 + 1) TF_{d,t}}{k_1((1-b) + b \cdot dl_d/avdl) + TF_{d,t}}$$

where N is the number of documents in the corpus, dl_d is the length of the document d and $avdl$ is the average length of documents in the corpus.

When the query is extremely long, they suggest the following formula.

$$Score = \sum_{t \in q} \ln(N/f_t) \frac{(k_1 + 1) TF_{d,t}}{k_1((1-b) + b \cdot dl_d/avdl) + TF_{d,t}} \cdot \frac{(k_3 + 1) TF_{q,t}}{k_3 + TF_{q,t}}$$

where $TF_{q,t}$ is the term frequency of the term t in the query q.

When multiple sources of information are available (e.g. anchor text, document text, and Pagerank), Robertson et al. [31] proposed a simple extension to BM25 to make it field-weighted.

2.5.3 Language mixture model

In this model, the similarity between the query and the document is computed by the probability $P(d|q)$ of generating the document d given the query q. By Bayes law, this probability can be computed as follows.

$$P(d|q) = P(q|d)P(d)/P(q)$$

For the same query, $P(q)$ is the same for all documents. If we assume that the prior probability of generating each document is the same, the only component determining the ranking is, therefore, $P(q|d) = P(q|M_d)$, the probability of generating the query q given the language model M_d of the document d. Thus, a ranking of the documents can be determined by sorting the documents in the decreasing order of $P(q|d)$. There are many possible choices for the language model of a given document d. Hiemstra [14], and Miller et al. [24] use the mixture model of two multinomial distributions: one for the document and one for the whole corpus. Ponte and Croft [27] instead use the multivariate Bernoulli model.

To simplify the task, it is usually assumed that the terms are generated independently. The maximum likelihood estimate of the probability of generating t given document d is

$$p_{ml}(t|M_d) = \frac{f_{d,t}}{dl_d}$$

Using this as an estimation for $p(t|M_d)$ is not robust as terms appear very sparsely in the documents. For example, if a query term does not appear in a document, that effectively rules out the document as $\hat{p}(q|M_d)$ would be 0. The estimate, however, can be smoothed in various way to give some weight to unseen words. One possible smoothing rule is to compute the estimate based on not only the occurrences of the term in the document but also the occurrences in all other documents in the corpus.

$$p_{ml}(t|M_c) = (\sum_d f_{d,t})/(\sum_d dl_d)$$

$$\hat{p}(t|M_d) = (1-\lambda)p_{ml}(t|M_d) + \lambda p_{ml}(t|M_c)$$

where M_c is the language model of the whole corpus, and λ is a constant between 0 and 1.

Once these estimates are obtained, Ponte and Croft [27] use the following formula for estimating the probability of generating the query q given the model of the document d.

$$\hat{p}(q|M_d) = \Pi_{t \in q} \hat{p}(t|M_d) \Pi_{t \notin q} (1 - \hat{p}(t|M_d))$$

The above formula comes directly from the assumption that the terms are generated independently.

2.6 Combination of different signals

In the previous sections, we have described various signals based on the content of the documents as well as the links between them. These signals need to be combined to produce a single final score. To prevent the domination of a single signal over all other signals, each signals is usually *soft-limited* i.e. the contribution is linear when the value is small but when the value gets large, the signal's contribution gets cut off slowly. One example is the formula for $TF_{d,t} = 1 + \ln f_{d,t}$ [2]. The contributions of different signals can then be combined in many ways. The simplest combining function is the linear function. Let w_i be the weight of the ith signal, and $f_i(d)$ be the score of the document d with respect to the ith signal. The linear combining function is as follows.

$$Score = \sum_i w_i f_i(d)$$

Soft-limiting and weighting help bring the contributions of different signals to the same scale and similar distributions. When different signals are not on the same scale or exhibiting similar distributions, there are also other methods for combining them. One such method is re-ranking: first the documents are sorted according to one signal. Then the top documents are re-ranked based on the second signal.

In this thesis, we will explore many different combining functions, including many piecewise linear and quadratic functions.

2.7 Evaluation

There are many metrics over which the performance of a search system can be measured. In this thesis, we are only concerned with the relevancy of the search results. In order to measure relevancy, a search system is usually tested against a set of queries on some standard corpora. For each of these queries, human evaluators evaluate the relevancy of all documents in the corpora and rate them as irrelevant or relevant (there might be a finer grain evaluation of how relevant a document is). The performance of a search system is then evaluated by several metrics based on the position of the relevant documents in the search results and how many there are among the top results.

Broder [5] classifies web queries into three categories: navigational, informational, and transactional. We will only consider the following two types:

- **Navigational queries** are queries looking for a specific document given its name, title, or a keyword. For example, the query "CNN" is looking for the homepage of CNN.

- **Informational queries** are queries looking for information about a general topic, which can be found in many web pages. For example, the query "latex format table" looks for information about how to format a table in Latex.

2.7.1 Navigational queries

Among navigational queries, there are two sub-categories: home page finding, and named page finding. Home page finding queries look for specific home page of an organization given its name. For example, the query "USDA" looks for the home page of the United States Department of Agriculture. Named page finding queries can be considered as a more general version of home page finding queries. The objective here is to find a document given its name, or its topic. For example, the query "FBI's most wanted list" looks for the document listing FBI's most wanted list.

2.7.2 Informational queries

Informational queries look for information about a general topic. There are multiple levels of difficulty to these queries. In the ad-hoc tasks, the goal is to find as many relevant documents as possible. In the topic distillation tasks, the goal is to find all the home pages of the important resources for a given topic. For example, the topic distillation query "HIV/AIDS" on the .GOV corpus asks for all the home pages of the government websites about "HIV/AIDS". Note that these pages are not necessarily the home pages of the government organization but are possibly home pages of specific parts of the websites devoted to the topic "HIV/AIDS". In this thesis, we will only consider the ad-hoc tasks, which is the easier category of the two.

2.7.3 Evaluation metrics

A search system can be evaluated in many different ways: the cost of performing each search query, the number of queries handled per second, etc. In this thesis, we are primarily concerned with the retrieval accuracy.

To evaluate a search system, standard test sets are usually used. Each test set consists of a document corpus, a set of queries, and the relevance judgment of the document in the corpus against each of the queries. Heavily trained system might become over-fitted to the test sets so it is important to verify the system on test sets it is not trained on.

There are many metrics for evaluating the accuracy of a search system. Sometimes comparing the metric values is not the best way to compare the performance of two different system as one could outperform the other in one metric but not the other. In this case, a better method for comparing is to put the two search rankings side-by-side and let human evaluators decide which one is better. A drawback of these side-by-side experiments is that they involve human evaluators and cannot be automated. In contrast, performance metrics on standard test corpora can be computed automatically without human intervention. In this thesis, we are concerned with two metrics, each suitable for a different class of queries.

The *Precision@N* of a query is computed by the number of relevant documents among the first N documents returned by the system. In this thesis, we use a positionally weighted variant of this metric as follows. The document at position i gets the weight of $i^{-0.8}$. Weighting different positions differently make the metric taking the positions of the relevant documents into account while still keeping track of the number of relevant documents. The value of the metric is

$$\frac{\sum_{i=1}^{n} i^{-0.8} \cdot 1_{i\text{th document is relevant}}}{\sum_{i=1}^{n} i^{-0.8}}$$

The Precision@N and its variants are suitable for the informational queries where there are a lot of relevant documents.

The *Reciprocal Rank* of a query is computed as one over the rank of the first relevant document in the search result. Intuitively, it measures how many irrelevant results the user has to skip to get to the relevant one. The *Mean Reciprocal Rank (MRR)* of a set of queries is the mean of the Reciprocal Rank of all queries in the set. This metric is suitable for the navigational queries where there is usually only one relevant document. Even when there are a few relevant documents, this metric is still an important measure of user satisfaction because the users usually look at only the top few results.

2.7.4 TREC corpora and web track evaluation

In this thesis, we consider retrieval tasks involving corpora with query sets from the web track and the ad-hoc track of the Text REtrieval Conference (TREC). TREC is a conference sponsored by DARPA allowing information retrieval research groups to compare the effectiveness and accuracy of their system on common test corpora.

We use three different corpora with very different link structures and size: the TREC/TIPSTER corpus disk 4 and 5, the WT2g corpus, and the .GOV corpus.

- *TREC/TIPSTER disk 4 and 5* is a 2GB corpus consisting of approximately 555,000 documents from the Financial Times Limited, the Congressional Record, and the Federal Register (disk 4), and the Foreign Broadcast Information Service, and the Los Angeles Times (disk 5). This is an ad-hoc corpus with no links between documents. Searching on this corpus resembles the problem of searching for a company storing its documents in a database or in hard drives.

- *TREC WT2g* is a 2GB corpus consisting of approximately 250,000 documents, which is a subset of the VLC2 corpus. The average number of links per page is 4.7. This corpus is a relatively small collection of web pages with an interesting quantity of links.
- *TREC .GOV* is a 18.1GB corpus consisting of approximately 1.25 million documents from a web crawl of the .GOV domain in 2001 [10].

We consider four different retrieval tasks on different corpora and with different query types. The same parameter setting is used for all different tasks with no information about the query types available to the system.

- The ad-hoc track of TREC 8 [39]: this task consist of 50 queries (topic 401 to 450) on the TREC/TIPSTER corpus disk 4 and 5.
- The small web track of TREC 8 [39]: the queries for this task are also topics 401 to 450 but on the WT2g corpus.
- The named page queries of the web track at TREC 12 [10]: This task consists of 150 named page queries on the .GOV corpus.
- The named page and home page queries of the web track at TREC 13 [9]: This task consists of 75 named page queries and 75 home page queries also on the .GOV corpus.

With these different tasks, we try to cover many different scenarios with different query types and different link structures. The queries for the first two tasks are informational queries whereas the queries the the last two tasks are navigational queries. The first task resembles the problem of searching in a database without any link structure. The second task represents the problem of searching in a web graph with small number of links. The last two tasks represent the problem of searching in a web graph with a reasonable number of links. Because of the different query types, we pay attention to different metrics for different tasks. For the first two tasks, the metric Precision@5 is the more important one whereas for the last two tasks, the metric MRR is the more important one.

Chapter 3

Methodology

In this chapter, we present a system for finding good sets of parameter values for the ranking function using evolutionary strategies. The ranking function is a complex function combining many signals in a highly non-linear (and sometimes discontinuous) way so we opted for black-box optimization techniques, which are generally applicable to most functions. This also facilitate our experimentation with various forms of the function without recalculating the approximation function (as needed for approaches tailored to specific functions).

At a high level, the system consists of the following parts: a feature extraction program for extracting information from raw documents, a system for evaluating many parameter settings on the extracted features in parallel, and an implementation of an evolution algorithm to generate new sets of parameters based on the performance of the existing settings.

We start by describing how we extract features from documents and distribute to many machines. Then we describe our implementation of two algorithms for selecting new parameter settings based on the existing ones: the Nelder-Mead algorithm, and the evolutionary strategies algorithm.

3.1 Document feature extraction

Documents are crawled and indexed on a GSA. Then the documents are matched against the queries to compute the necessary features for scoring. All features of the documents (pagerank, word counts, number of anchor text hits, etc) are extracted to a log file. This file is then compressed and distributed to many computers to evaluate many settings in parallel.

3.2 Nelder-Mead algorithm

The Nelder-Mead algorithm [25] is a classical algorithm for function maximization. The algorithm is very general and easy to implement as it only requires evaluating the function and no information about the derivatives is needed. Consider a function f of n variables. The algorithm works as follows. At every step, the algorithm keeps track of the function value at $n+1$ vertices $x_1, \ldots, x_{n+1}$ of a simplex in the n dimensional space. Assume that $f(x_1) \geq f(x_2) \geq \ldots \geq f(x_{n+1})$. The algorithm attempts to replace x_{n+1} with a better point by performing the following steps.

1. *Reflection step* Compute $x_0 \leftarrow \frac{1}{n}\sum_{i=1}^{n} x_i$. Set $x_r = 2x_0 - x_{n+1}$. If $f(x_1) \geq f(x_r) > f(x_n)$ then set $x_{n+1} \leftarrow x_r$. Otherwise, continue to the next step.

2. *Expansion step* Only perform this step if $f(x_r) > f(x_1)$. Otherwise, continue to the next step. First, compute $x_e = 3x_0 - 2x_{n+1}$. If $f(x_e) > f(x_r)$, set $x_{n+1} \leftarrow x_e$. Otherwise, set $x_{n+1} \leftarrow x_r$.

3. *Contraction step* If $f(x_r) > f(x_{n+1})$, compute $x_c = (x_0+x_r)/2$. Otherwise, compute $x_c = (x_{n+1} + x_0)/2$. If $f(x_c) > max(f_r, f_{n+1})$, set $x_{n+1} \leftarrow x_c$. Otherwise, continue to the next step.

4. *Shrink step* For $i = 2, 3, \ldots, n+1$, set $x_i \leftarrow (x_i + x_1)/2$.

The above routine is repeated until the simplex converges to a point.

3.3 Evolutionary strategies

Evolutionary strategies [34] is a highly parallelizable optimization method for maximizing a general multivariate function. It uses probabilistic transition rules to simulate a natural evolutionary process. The algorithm repeatedly transits from a population to the next population until the whole population converges on a single point or a sufficient number of iterations has been passed. There are many variants of this general framework. In the rest of this section, we will describe our simple implementation of $(\mu \overset{+}{,} \lambda)$ evolutionary strategies.

Consider a function f over n variables. At every time step, the algorithm maintains a population of $\lambda(+\mu)$ individuals, each corresponds to a point in the n dimensional search space. The algorithm works as follows.

```
Initialize the initial population to $x_1, \ldots, x_\mu$
**repeat**
    Evaluate the fitness of the population
    Select the best $\mu$ parents from the population
    Add random noise to these parents to create $\lambda$ offspring
    Change the noise variances based on the performance of the offspring
    **if** $(\mu, \lambda)$ *evolutionary strategy* **then**
        Use the $\lambda$ offspring as the new population
    **else**
        Use the $\lambda$ offspring and the $\mu$ selected parents as the new population
    **end**
**until** *enough iterations are done*
**return** *the best individual in the last population*
```

Algorithm 3: Evolutionary Strategies

The offspring are generated by adding random Gaussian noise to the parents. Because each parameter varies over a different range, the amount of noise added to each is controlled by its range. The noise variances are adjusted in every iteration to balance exploration and exploitation as we do not want the algorithm to get stuck in a local maximum or jump far away from the optimum while being near it. We use a heuristic so-called the "one-fifth" rule to control the noise variance. If at least 20% of the new population are better than their parents, we are exploiting too much and the noise variance is increased. On the other hand, if at most 20% of the new population are better than their parents, we are exploring too much and the noise variance is decreased.

3.4 Implementation

We implemented the parallel evaluation system in C++ using the map-reduce framework. Parameter settings are distributed and evaluated on many machines in parallel. At the end of each round, the evaluation results are gathered and an algorithm is run to compute the next set of parameters to be evaluated on. We designed the system so that it is easy to switch from one algorithm to another and with variable number of parameters. In all of our experiments, we use evolutionary strategies for computing the new sets of parameter settings based on the performance of the existing ones. For testing purposes, we have also implemented the Nelder-Mead algorithm for comparison. Our experiment results suggests that evolutionary strategies is better with a reasonable population size.

To avoid over-fitting, after obtaining a setting from the system, we perform a sensitivity test by varying each parameter by 10% and verify if the performance varies too much. We also perform side-by-side experiments to compare the search results of the new setting and the original setting on a set of queries on a proprietary corpus.

3.5 Summary

In this chapter, we have described our method of computing parameter settings for the ranking function of the Google Search Appliance using evolutionary strategies. The flexibility and high parallelizability of this approach allow us to explore the parameter space efficiently.

Chapter 4

Importance of IDF

In this section, we investigate the importance of the inverse document frequency(IDF) in the ranking function. In the previous work, IDF has been established as a very helpful source of information in determining the relative importance of query words and hence, improves ranking accuracy a lot. However, IDF requires information about the whole corpus of documents, which is difficult to maintain in many contexts. In federation search, where search results are combined from many computers, the IDF on different machines managing different parts of the corpus can be vastly different, resulting in incongruous scores. IDF also make it difficult to combine search results from heterogeneous corpora such as binary files and web pages. In this thesis, we look at the effect of IDF on search results, comparing its usefulness and the noise it introduce to the ranking function. We hope that in light of our work, the impact of IDF on the ranking function can be reevaluated and there is a hope to avoid its costly computation and maintenance.

We compare two variants of the otherwise exactly identical ranking function, one with the IDF component and one without. The variant with IDF is hand-tuned by human and the variant without IDF is optimized by our evolutionary strategies framework. We then compare these two optimized settings on four retrieval tasks, with and without query expansion.

It is evident from the comparisons in figure 4-1 and 4-2 that without IDF, the system performs generally equally well and sometimes better than with IDF.

Task	change by removing IDF
TREC 8 ad-hoc	
P@5	+0.01
MRR	-0.01
WT2g	
P@5	+0.04
MRR	+0.07
TREC 12 named page	
MRR	0.00
TREC 13 home page	
MRR	-0.01
TREC 13 named page	
MRR	-0.01

Figure 4-1: Change in performance of system using query expansion

Task	change by removing IDF
TREC 8 ad-hoc	
P@5	-0.02
MRR	-0.02
WT2g	
P@5	+0.04
MRR	+0.05
TREC 12 named page	
MRR	-0.01
TREC 13 home page	
MRR	+0.01
TREC 13 named page	
MRR	+0.01

Figure 4-2: Change in performance of system not using query expansion

On navigational queries, the performance of the two variants are basically identical. On some queries such as the query "map, mapping, cartography" in TREC 13, IDF overly emphasizes the importance of "cartography" over the other two terms and pushes the relevant document from the top position to the second. However, on the query "HEV program homepage", the use of IDF helps pulling the relevant document from the second position to the top position.

Interestingly, on informational queries on the WT2g corpus, the variant without IDF has a significant improvement over the variant with IDF. This phenomenon actually makes sense because the WT2g is the smallest corpus (half the size of the ad-hoc corpus and one-fifth the size of the GOV corpus) among the four tasks and it is reasonable to expect the IDF measures are more noisy than the other much larger corpora.

Beside these corpora, we also perform side by side experiments comparing these two variants on a much larger proprietary corpus. In the side by side experiment, the search results returned by the two systems are displayed side by side. Human evaluators compare them and determine which one is better. The results of our experiments also give indication that the two variants are very competitive and the one without IDF performs slightly better.

There are reasons that could explain this phenomenon. The first reason is the use of the stop words. Extremely common words such as "and", "a", "the" are not used in scoring, thus eliminating the need for IDF in these extreme cases. The second reason is the AND semantics of the queries, where we only consider documents containing all the query terms. This avoids the need to compare documents containing only some of the query terms.

4.1 Summary

In this chapter, we investigated the impact of IDF on the ranking performance. Experiment results suggest that even though IDF can be helpful in many situations, it can be noisy and detrimental in other cases. It's impact is limited given the filtering of stop words such as and, a, the, etc and the AND semantics of the search queries.

Chapter 5

Combination of query-independent and query-dependent measures

As described in chapter 2, the score of a document is determined by many measures: a page-dependent score, a score from hits in anchor texts, and a query independent score based on the link graph. Since these scores are computed in vastly different ways with very different theoretical motivations, it is conceivable that they follow different distributions and contribute differently to the final relevance score. In this chapter, we investigate their relative importance and find a good way to combine them.

5.1 Method

We implemented the evaluation function so that it can dynamically take any combination functions with constants and cutoffs and evaluate document using these functions. The evaluation system is then tested against many piece-wise linear and quadratic functions with variable cutoffs. The main criterion these functions have to satisfy is monotonicity. The values for the cutoffs are optimized using our evolutionary strategies framework.

Impact transformation [2] is a technique for transforming the contributions from different query terms to make the score not overly skewed by a single term. Even though this technique is invented for the OR semantic of queries, where any document containing at least one query term is considered, we find this technique also helpful in this context. Here we use impact transformation as a way to balance the contribution from the query-dependent score (combination of the page-dependent score and the contribution from the anchor text) and the query-independent score based on the number of query terms. Intuitively, as the number of query terms increases, the query-dependent score becomes more reliable and thus, should get a higher weight. Impact transformation boosts the low scores of every query term, thus increasing the whole score proportionally to the number of query terms, which is exactly what we want.

5.2 Results

From our experiments, there are several interesting results. These results can be explained when looking at the scatter plots of the documents matching the given queries. Firstly, the functions with varying behaviors for different parameter ranges perform no better than the simpler one behaving the same over all parameter ranges as long as each score is soft-limited individually. Consider the scatter plots for the queries "Art white house" and "Children's butterfly site life cycle page" in figure 5-1. Even though the relevant documents are in very different parameter ranges in these two queries, it is evident that the slope of the combining function should be the same in both of the ranges.

Secondly, in most queries, the linear function works nearly as well as more complicated quadratic functions or even piecewise quadratic functions. In figure 5-2, it is clear that the linear function given for the range of parameters containing the relevant documents of two queries "Club drugs" and "MedWatch" is basically optimal and any piecewise quadratic functions cannot do much better.

However, for a small number of queries in a particular parameter range, quadratic function might perform better as it can vary the relative contribution from the two scores. For the two queries "national fire plan usda" and "Florida Keys Marine sanctuary" in figure 5-3, it is conceivable that a quadratic function would perform

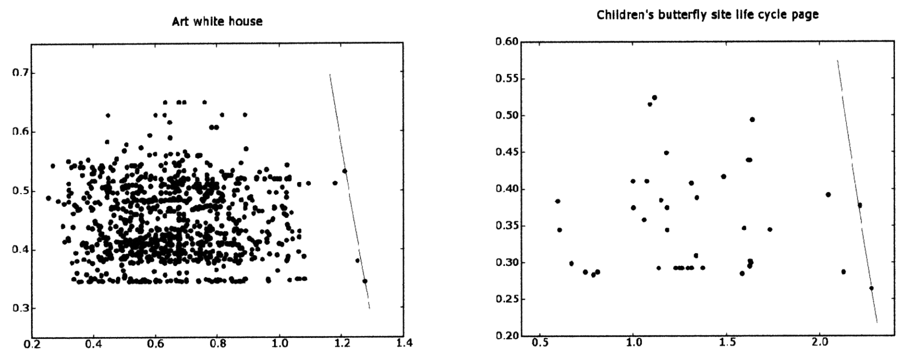


Figure 5-1: Two queries whose relevant documents are in different parameter ranges. Blue dots are irrelevant documents, red dots are relevant documents, the vertical axis is the query-independent score, the horizontal axis is the query-dependent score, the blue curve is the optimal function combining the two scores.

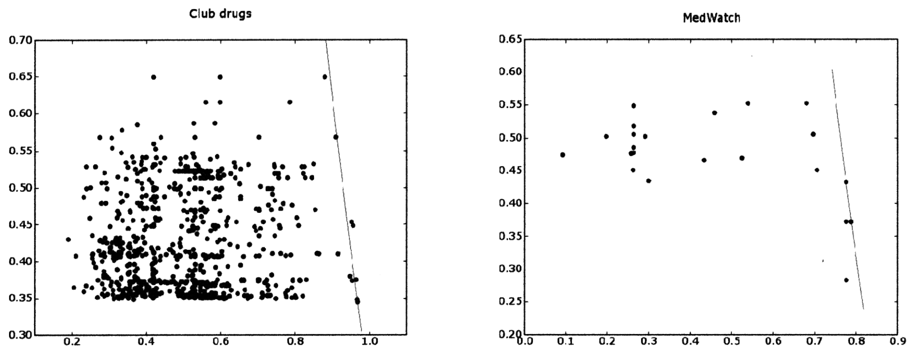
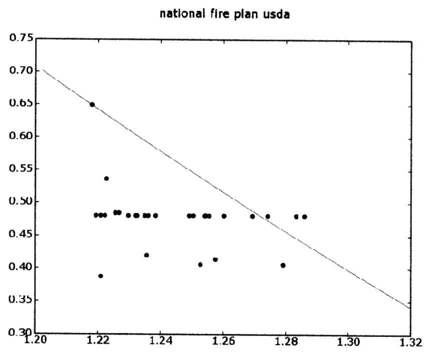


Figure 5-2: Two queries whose relevant documents are in the same parameter range

better than the current near-linear function. These particular queries suggest that the query-independent score is most important when it is large and its importance drops quickly as it gets closer to the average value.

The evaluation results also suggest that for small corpora, query-dependent scores are much more important than query-independent scores. For example, for the query "Food illness reporting" in figure 5-4, it is clear that the page-independent score is largely noisy and does not give much information about the relevancy of the document.

In general, the query-independent score is only helpful in finding home pages. The home page query "Grand Canyon monitoring research center" in figure 5-5 shows that



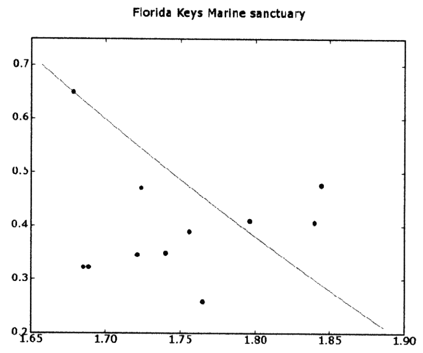


Figure 5-3: Two queries whose relevant documents have high query-independent score.

the query-independent score is a much better indication of relevancy than the query-dependent score.

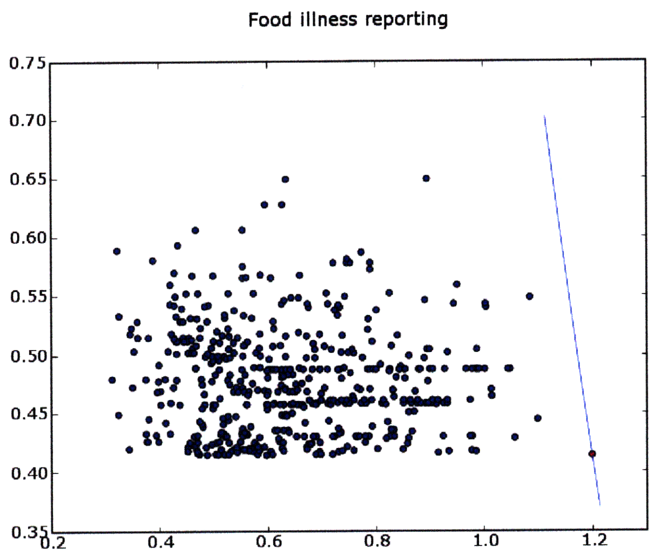


Figure 5-4: A typical query.

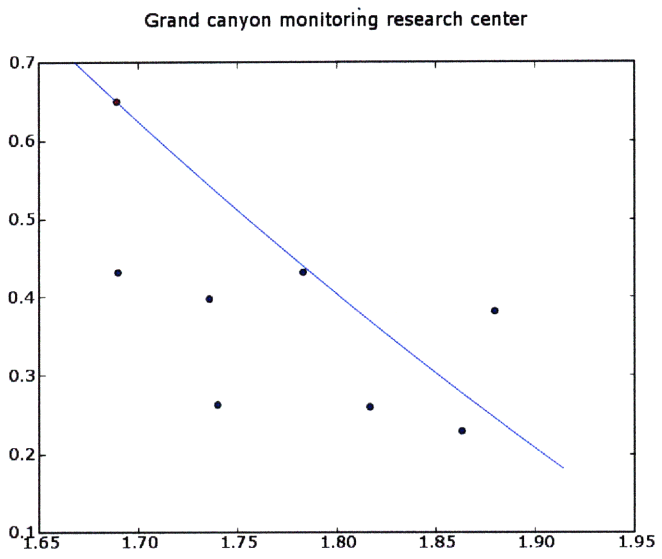


Figure 5-5: A home page finding query where the query-independent score is much more important than the query-dependent score.

Chapter 6

Signals based on term positions

In this chapter, we explore the impacts of two signals based on the positions of the query terms in the document: the position of the first hit in the document (also called Chronological Term Rank (CTR) [37]), and the length of the shortest snippet containing all the query words. Intuitively, these signals provide some information about the relevancy of the document. If a word occurs near the beginning of a document, there is a good chance that the word is relevant to the main content of the document. Similarly, if all queried words occur close to each other, there is a better chance of them being used together as opposed to scattering randomly in the document. However, it is also conceivable that these signals are prone to noise. In the subsequent sections, we attempt to study the use of these signals and evaluate whether they are worth pursuing further.

6.1 Chronological term rank

The chronological term rank (CTR) of a term is the first position it appears in the body text of a document. Because it is part of the content of the body text, it is sensible to apply the document length normalization to it. We investigate 6 ways of combining the first hit position with the other signals, taking different combinations of linear and logarithmic functions, and normalized and unnormalized by document length. Let tr be the CTR of a term. The contribution of CTR to the score is

Function	Description
$\frac{c}{tr+1}$	inverse rank
$\frac{c}{\log(tr+2)}$	inverse log rank
$c \cdot (1 - \frac{(tr+1)}{DLN})$	linear, normalized
$c \cdot (1 - \frac{log(tr+2)}{\log DLN})$	logarithmic, normalized
$\frac{c \cdot DLN}{tr+1}$	inverse rank, normalized
$\frac{c \cdot \log DLN}{log(tr+2)}$	inverse log rank, normalized

Figure 6-1: Formulas for computing the contribution of CTR to the relevance score

Function	Description
$\frac{c}{snippet+2}$	inverse
$\frac{c}{\log(snippet+2)+2}$	inverse log
$c \cdot (1 - \frac{(snippet+1)}{DLN})$	linear, normalized
$c \cdot (1 - \frac{log(snippet+2)}{\log DLN})$	logarithmic, normalized
$\frac{c \cdot DLN}{snippet+1}$	inverse rank, normalized
$\frac{c \cdot \log DLN}{log(snippet+2)}$	inverse log rank, normalized

Figure 6-2: Formulas for computing the contribution of the snippet length to the relevance score

computed as described in figure 6-1. For each of the functions, we optimize the parameters using our evolutionary strategies framework.

Unfortunately, for all the functions we considered, the results are mixed. There is some small improvement for some tasks and there is some small decline in some other tasks. In figure 6-3, we show the performance of the system on all the tasks we consider. While it is possible to get a small improvement in the TREC 8 ad-hoc task and the home page queries of TREC 13, the performance in the WT2g task and the named page queries of TREC 12 decline. The performance of the rest of the functions are shown in the appendix.

6.2 Length of the shortest snippet

A snippet is a sub-sequence of the body text of the document containing all the query terms. In this section, we are only concerned with the length of the shortest snippet. There are many possible ways to incorporate this information into the relevance score. In figure 6-2, we describe 6 possible ways to compute the contribution of the snippet length to the relevance score. For each of the functions, we optimize the parameters using our evolutionary strategies framework.

Similar to the case of CTR, for all the functions we considered, the results are mixed. There is some small improvement for some tasks and there is some small decline in some other tasks. In figure 6-4, we show the performance of the system using one of the formula we described on all the tasks. While it is possible to get a small improvement in the TREC 8 ad-hoc task and the home page queries of TREC 13, the performance in the WT2g task and the named page queries of TREC 12 decline. The performance of the rest of the functions are shown in the appendix.

6.3 Optimal re-ranking

While the performance of the various functions tested in the previous section was not particularly attractive, it is possible that there are other ways to incorporate the positional based signals to the relevant score we have not considered. Thus, we cannot conclude that these signals do not provide any improvement. In this section, we consider the optimal re-rankings [38] based on the CTR and the length of the shortest snippet, which provide strict upper bound of the contribution of these signals.

A re-ranking scheme works as follows. After all the documents are scored and sorted in decreasing order, the top n documents are re-ranked using some additional information available. These top n documents are sorted based solely on the additional information, completely ignoring the original relevance scores. The *optimal re-ranking* of a signal is the re-ranking scheme that for each query, dynamically chooses n

equal to the position of the first relevant document in the list ranked by the relevance score. It is easy to see that this re-ranking scheme works better than any ranking function combining the original relevance score and the additional information. In some sense, it maximally improves the queries where the additional information is helpful and ignores the queries where the additional information is harmful.

In figures 6-5 and 6-6, we present the gap between the performance of the current system and the optimal re-ranking scheme using CTR. In figures 6-7 and 6-8, we present the gap between the performance of the current system and the optimal re-ranking scheme using the shortest snippet length. The wide gap between current performance and the optimal re-ranker's suggests that even though these signals may not be universally useful, there could be a class of queries where they could provide good information. CTR seems to be particularly helpful for home page queries, as the optimal re-ranking using CTR gets a 0.1 improvement in MRR over the current system.

6.4 Summary

In this section, we considered two positional based signals. Even though the experiment results is not promising, it is worth revisiting these signals again given the big difference between the optimal re-ranking and the experimental results. It could be the case that they are only helpful in a specific class of queries like home page queries.

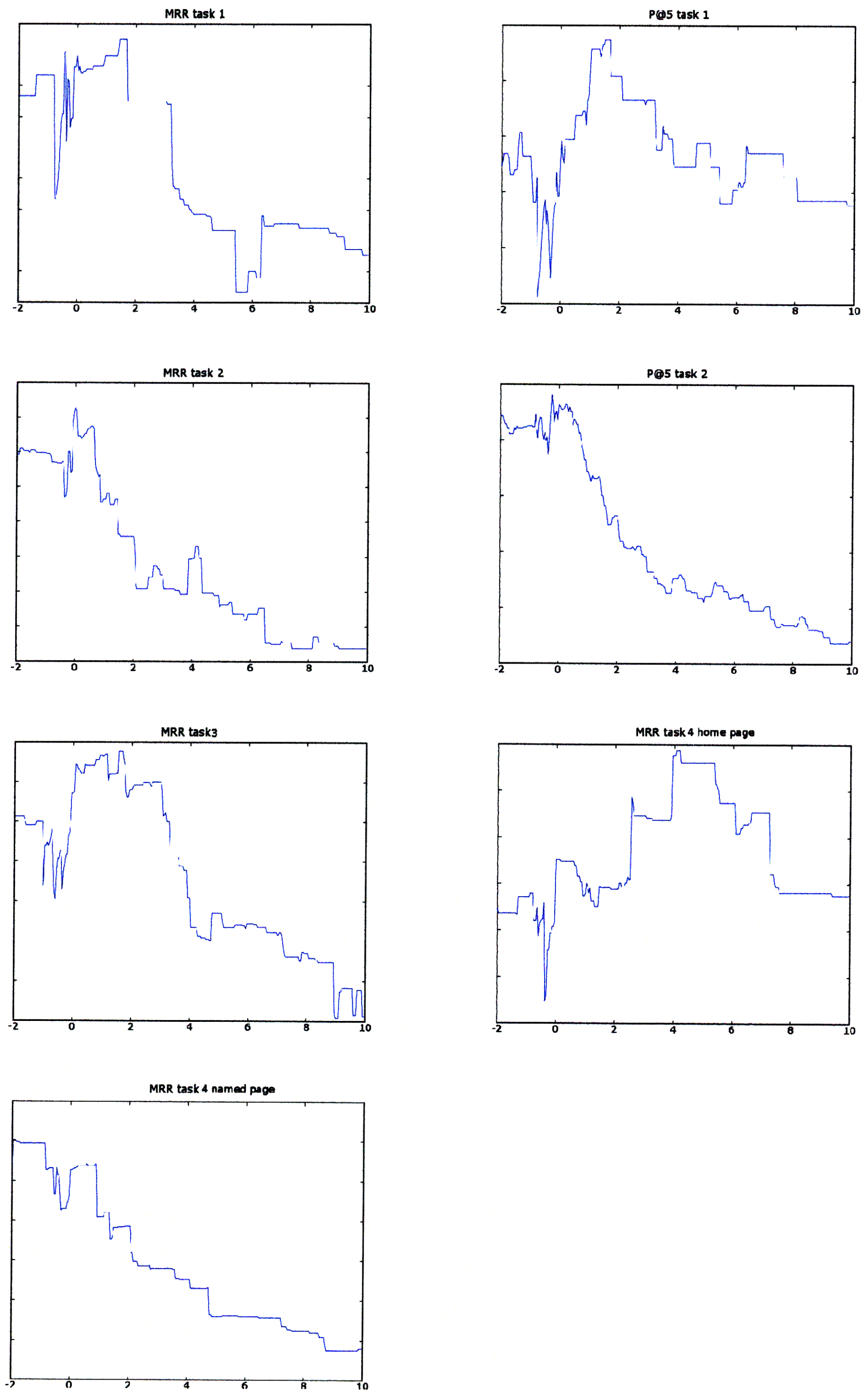


Figure 6-3: Performance of the system with contribution from CTR using the function $c \cdot (1 - \frac{(tr+1)}{DLN})$. The horizontal axis is c and the vertical axis is the metric score

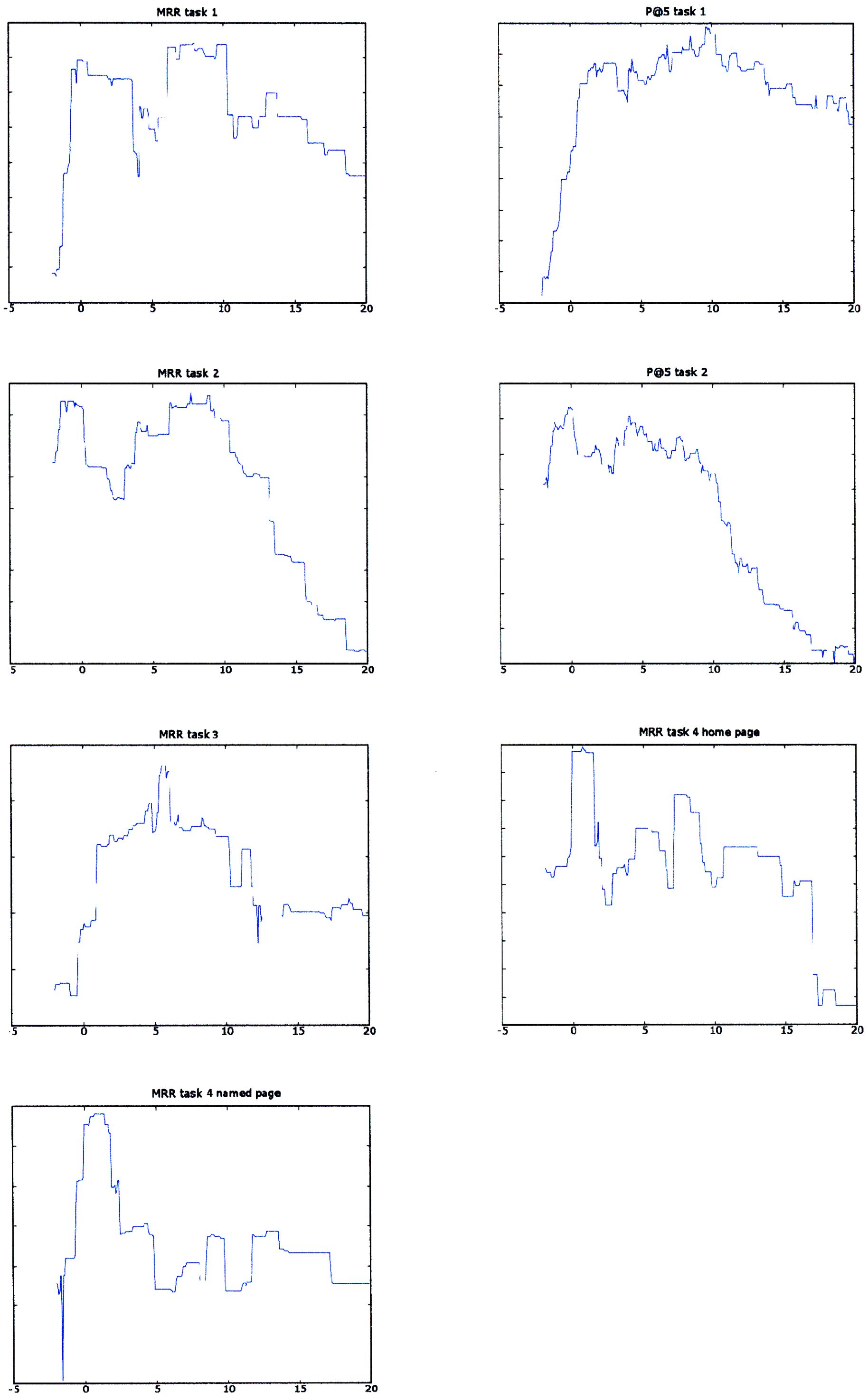


Figure 6-4: Performance of the system with contribution from the snippet length using the function $\frac{c}{snippet+2}$

Task	advantage of optimal re-ranking with CTR
TREC 8 ad-hoc	
P@5	+0.03
MRR	+0.10
WT2g	
P@5	+0.04
MRR	+0.11
TREC 12 named page	
MRR	+0.06
TREC 13 home page	
MRR	+0.11
TREC 13 named page	
MRR	+0.06

Figure 6-5: Re-ranking with CTR on system without IDF

Task	advantage of optimal re-ranking with CTR
TREC 8 ad-hoc	
P@5	+0.02
MRR	+0.07
WT2g	
P@5	+0.04
MRR	+0.10
TREC 12 named page	
MRR	+0.06
TREC 13 home page	
MRR	+0.10
TREC 13 named page	
MRR	+0.05

Figure 6-6: Re-ranking with CTR on system with IDF

Task	advantage of optimal re-ranking with snippet length
TREC 8 ad-hoc	
P@5	+0.03
MRR	+0.08
WT2g	
P@5	+0.05
MRR	+0.14
TREC 12 named page	
MRR	+0.07
TREC 13 home page	
MRR	+0.05
TREC 13 named page	
MRR	+0.05

Figure 6-7: Re-ranking with snippet length on system without IDF

Task	advantage of optimal re-ranking with snippet length
TREC 8 ad-hoc	
P@5	+0.02
MRR	+0.06
WT2g	
P@5	+0.05
MRR	+0.13
TREC 12 named page	
MRR	+0.08
TREC 13 home page	
MRR	+0.06
TREC 13 named page	
MRR	+0.07

Figure 6-8: Re-ranking with snippet length on system with IDF

Chapter 7

Conclusion and future work

In this thesis, we studied the contributions of various signals to the ranking function of the Google Search Appliance. We looked at both well-established signals such as the inverse document frequency (IDF), and newly suggested signals such as the chronological term rank (CTR) and the length of the shortest snippet containing all the search terms. While helpful in many cases, IDF can be noisy when the size of the corpora is small. For the bigger corpora, the contribution of IDF is also marginal. We were able to improve the performance of the ranking function on the Google Search Appliance on small corpora such as the WT2g while keeping the relevancy relatively stable on bigger corpora such as the GOV corpus. Our experiment results on CTR and the snippet length are mixed with both improvement and degradation. It is worth noting, however, that these two signals might be helpful for home page queries and are worth investigating further. Additionally, we also looked at various ways for combining signals, in particular, the query-independent score and the query-dependent score. In most queries, a simple linear combination function works almost as well as more sophisticated functions.

There are many future directions from this work. While our initial investigation on the CTR and the length of the shortest snippet seems to indicate these signals are too noisy to use, it is possible that they are helpful in some specific classes of queries. For example, the optimal re-rankings using these signals improve significantly over the performance of the current system without them on homepage finding queries. It is also interesting to explore the classes of queries where the query-independent score is important. Even though in our experiments, the query-dependent score is much more important, there are specific queries (especially some home page finding queries) where the query-independent score is much more helpful. Another interesting issue is that we did not have enough queries to evaluate the system on specific classes of queries and different query lengths. It would be interesting to study the contribution of various signals for different query lengths.

A shortcoming of our method is that the evolutionary strategies framework lacks the understanding of the implicit constraints imposed by other parts of the search system and the meanings of various parameters being used. Therefore it is more prone to over-fitting to training data than the parameter settings hand-tuned by human engineers. It is possible to extend our framework with additional constraints to restrict the solution space and reduce the risk of over-fitting.

Appendix A

Results of chronological term rank and snippet length experiments

In this section, we describe the results of the experiments involving the CTR and the snippet length.

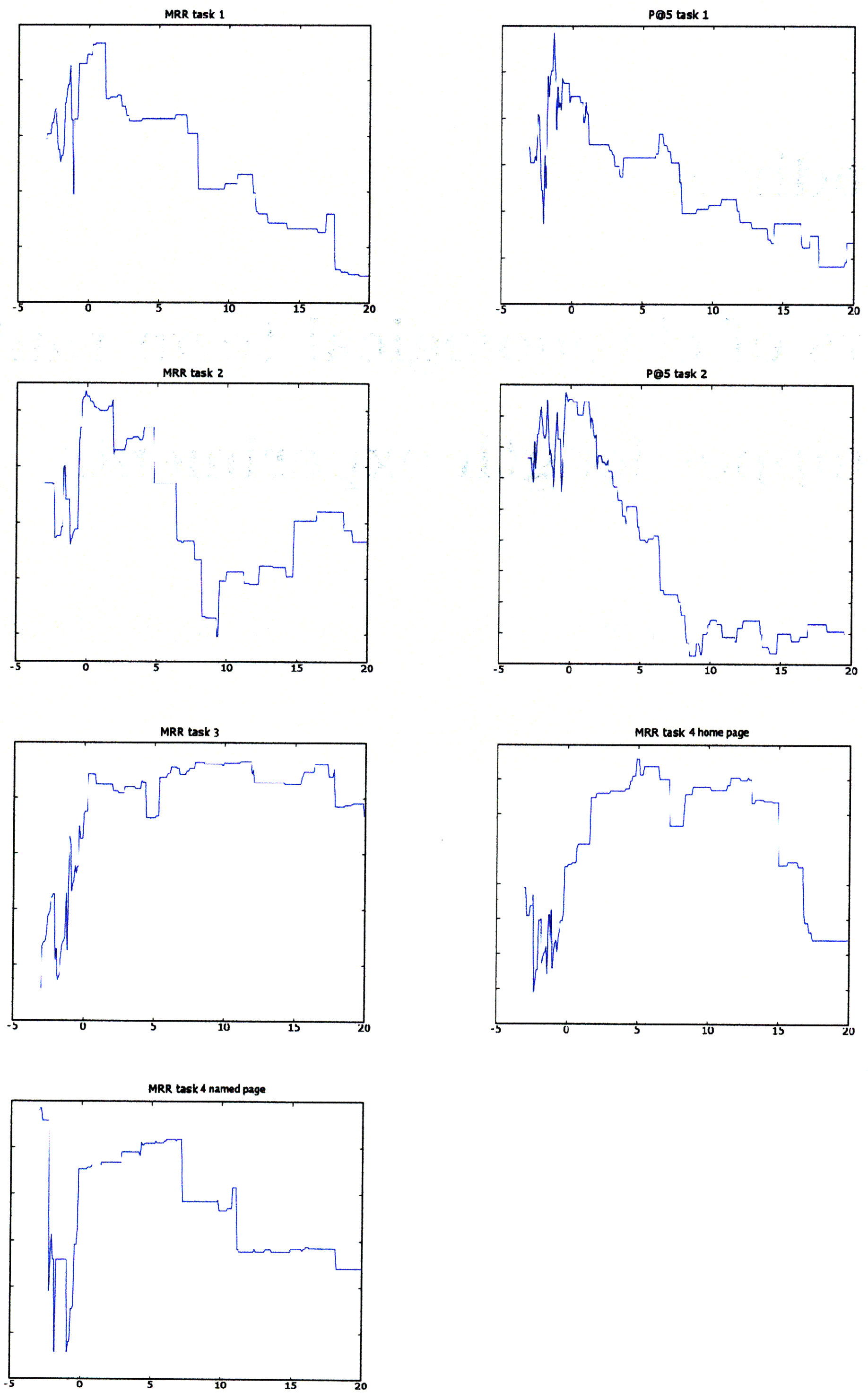


Figure A-1: Performance of the system with contribution from CTR using the function $\frac{c}{\log(tr+2)}$.

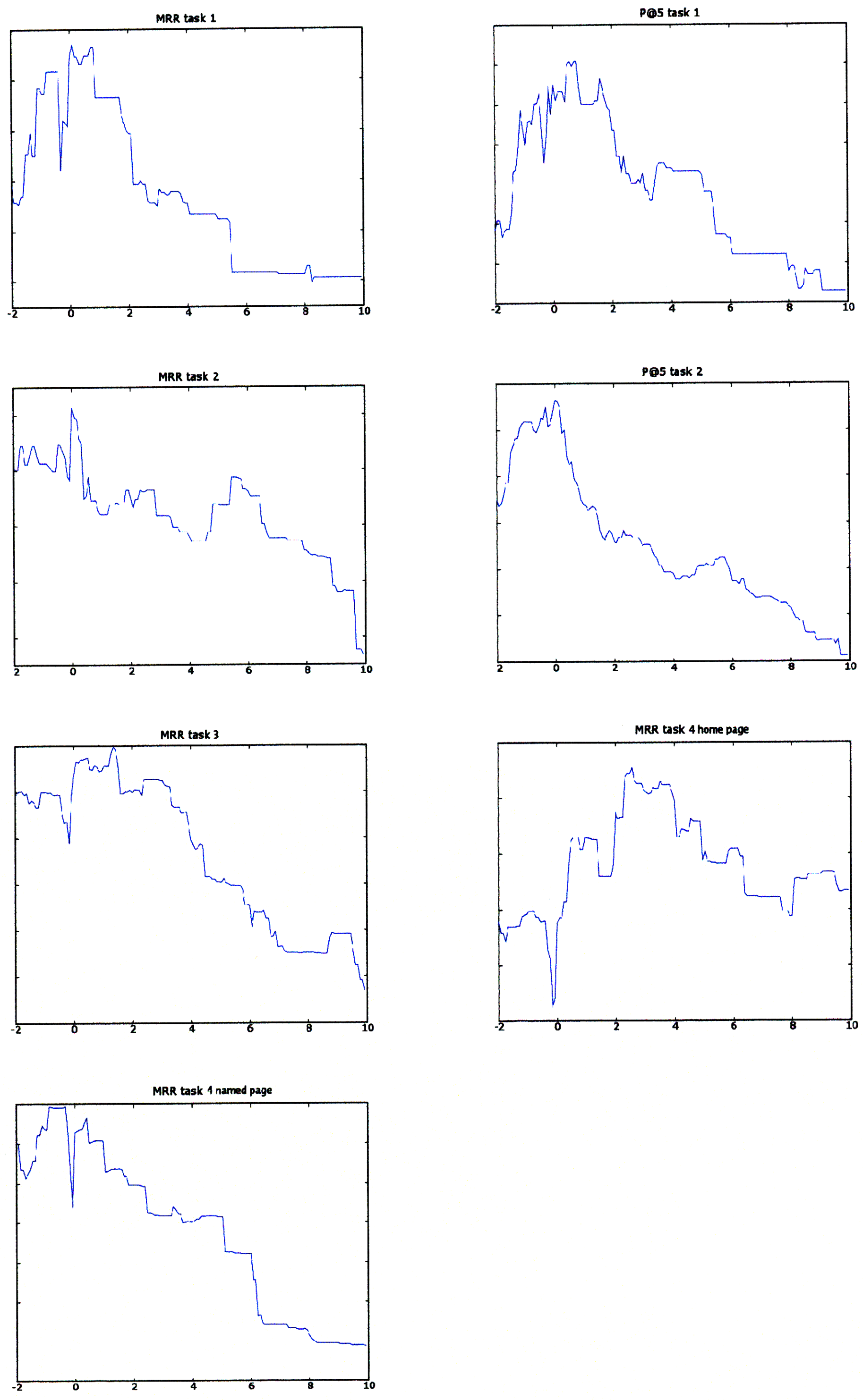


Figure A-2: Performance of the system with contribution from CTR using the function $c \cdot (1 - \frac{log(tr+2)}{\log DLN})$

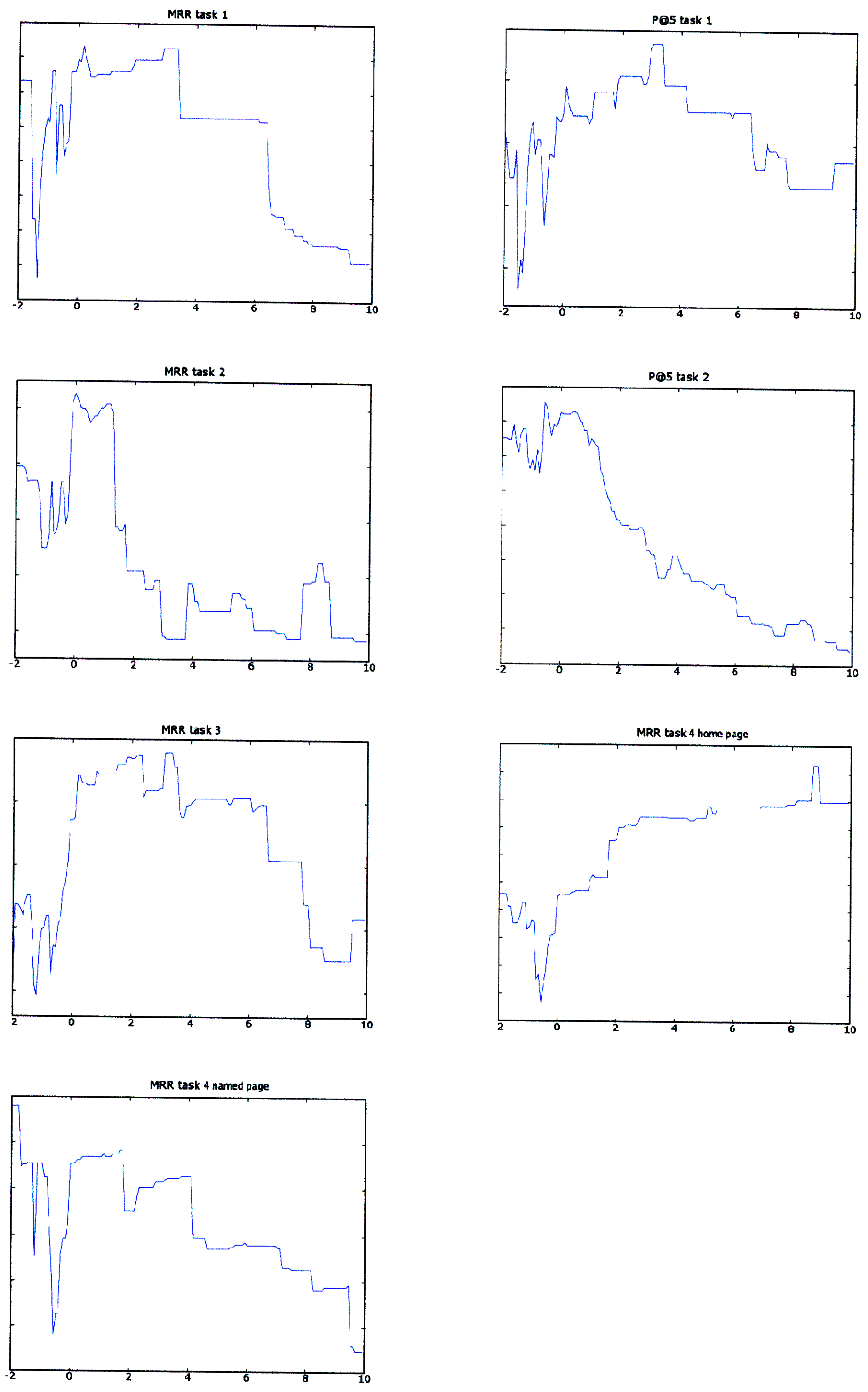


Figure A-3: Performance of the system with contribution from CTR using the function $\frac{c \cdot DLN}{tr+1}$

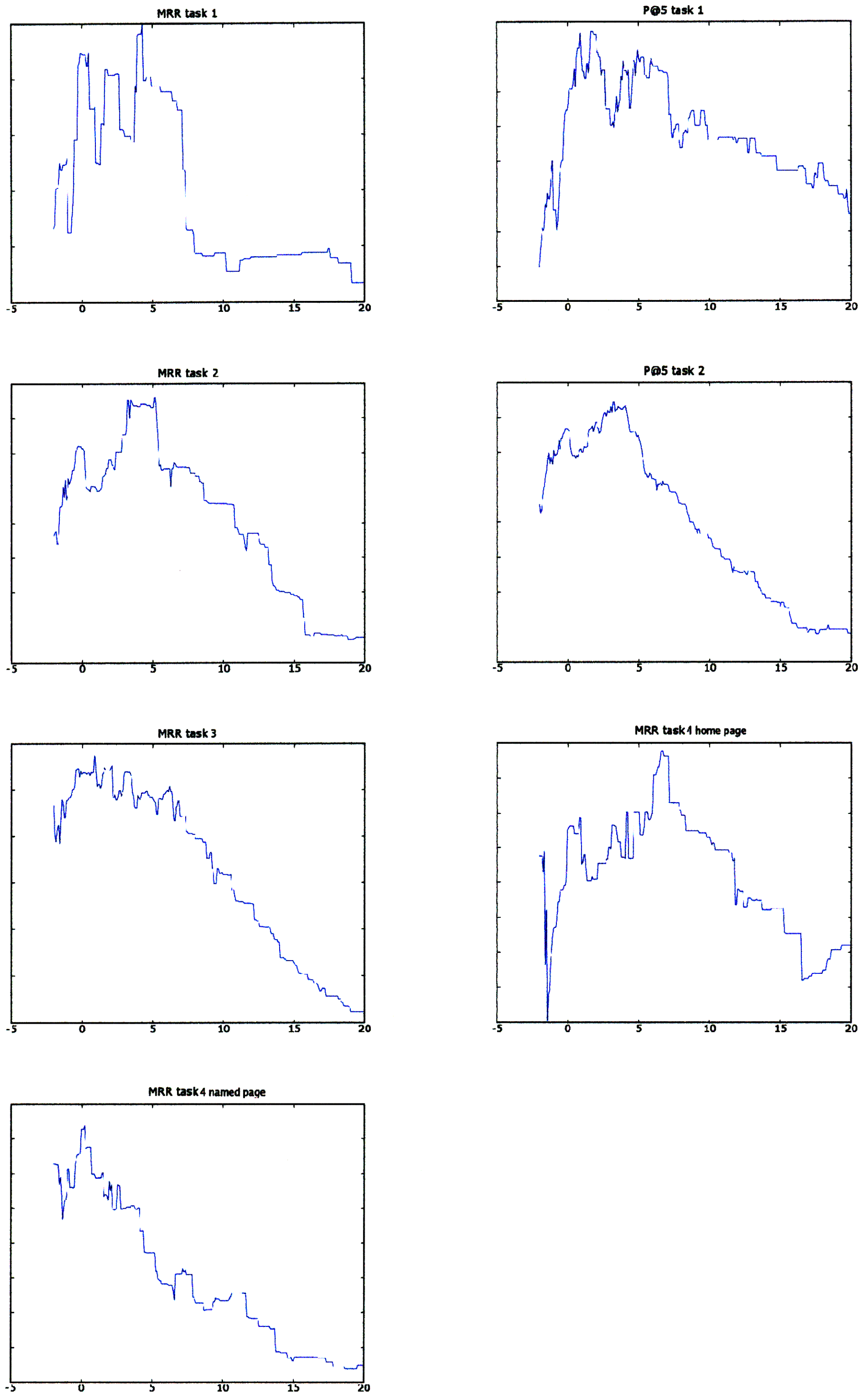


Figure A-4: Performance of the system with contribution from the snippet length using the function $c/(log(snippet + 2) + 2)$

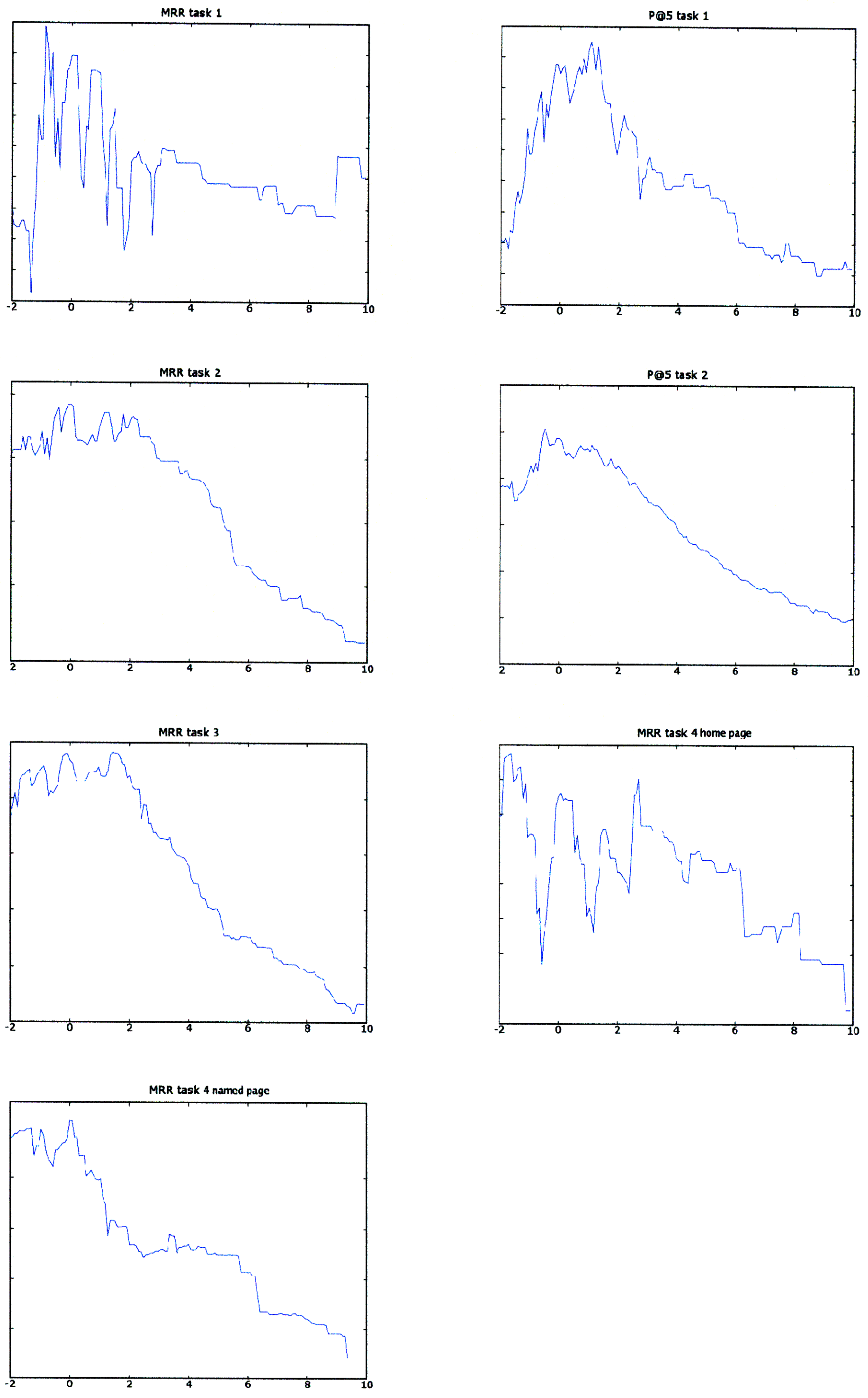


Figure A-5: Performance of the system with contribution from the snippet length using the function $\frac{c \cdot DLN}{snippet+1}$

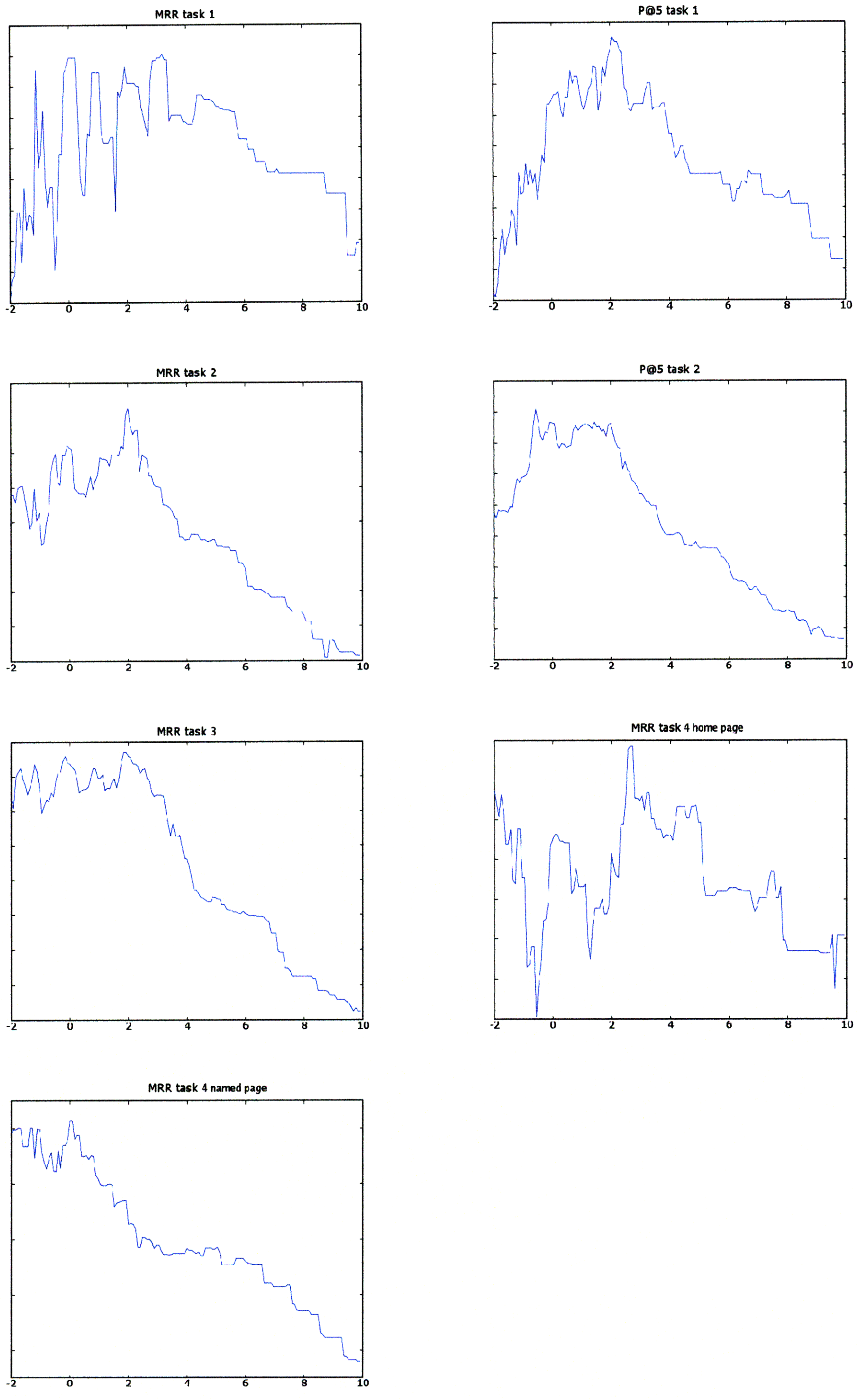


Figure A-6: Performance of the system with contribution from the snippet length using the function $\frac{c \cdot \log DLN}{log(snippet+2)}$

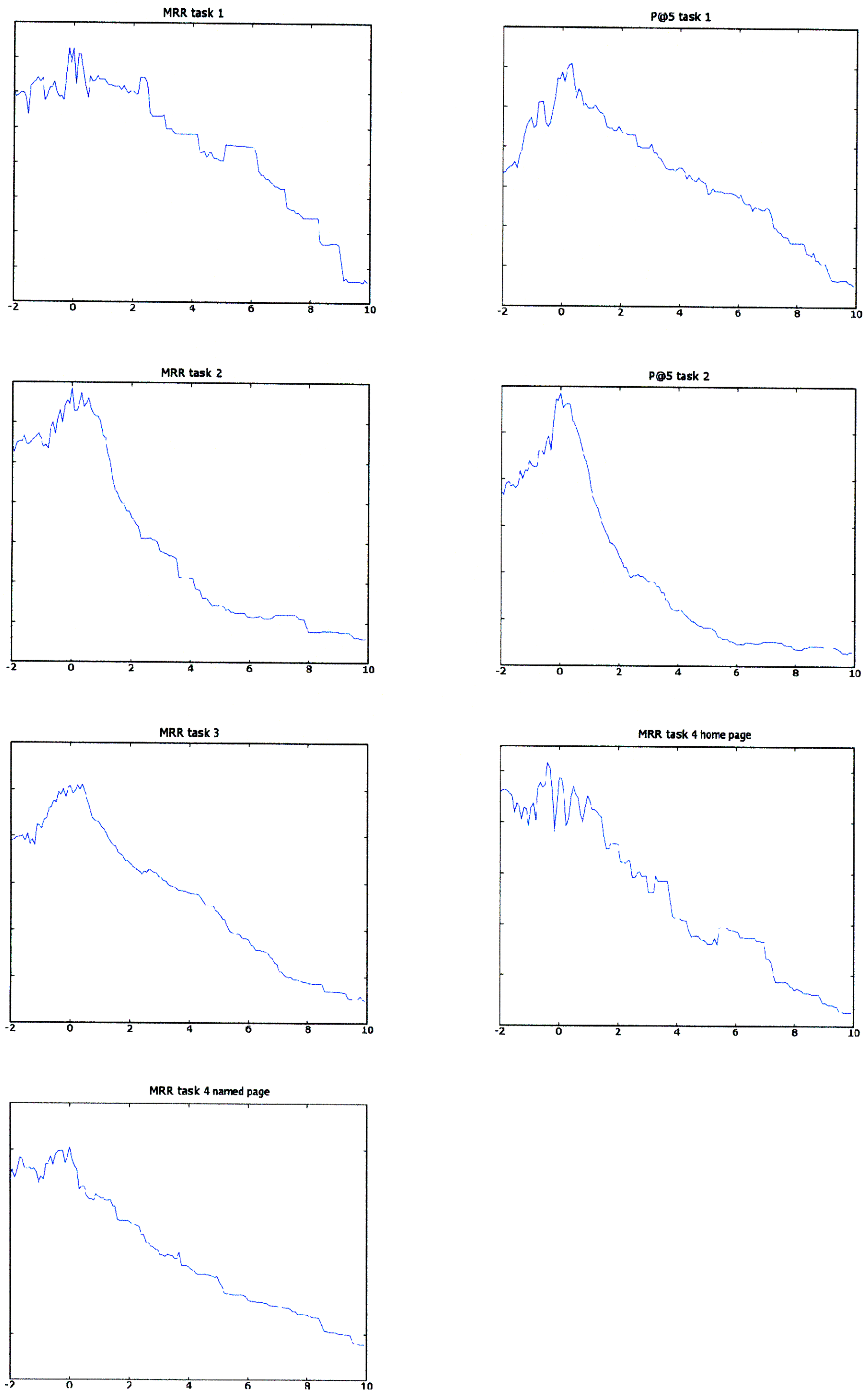


Figure A-7: Performance of the system with contribution from the snippet length using the function $c \cdot (1 - \frac{(snippet+1)}{DLN})$

Bibliography

[1] Serge Abiteboul, Mihai Preda, and Gregory Cobena. Adaptive on-line page importance computation. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 280–290, New York, NY, USA, 2003. ACM.

[2] Vo Ngoc Anh and Alistair Moffat. Impact transformation: effective and efficient web retrieval. In *SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 3–10, New York, NY, USA, 2002. ACM.

[3] Krishna Bharat and Monika R. Henzinger. Improved algorithms for topic distillation in a hyperlinked environment. In *SIGIR '98: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 104–111, New York, NY, USA, 1998. ACM.

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

[5] Andrei Broder. A taxonomy of web search. *SIGIR Forum*, 36(2):3–10, 2002.

[6] Stefan Büttcher, Charles L. A. Clarke, and Brad Lushman. Term proximity scoring for ad-hoc retrieval on very large text collections. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 621–622, New York, NY, USA, 2006. ACM.

[7] Deng Cai, Xiaofei He, Ji-Rong Wen, and Wei-Ying Ma. Block-level link analysis. In *SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 440–447, New York, NY, USA, 2004. ACM.

[8] Junghoo Cho, Hector Garcia-molina, and Lawrence Page. Efficient crawling through url ordering. In *Computer Networks and ISDN Systems*, pages 161–172, 1998.

[9] Nick Craswell and David Hawking. Overview of the trec-2004 web track. In *TREC '04: Proceedings of TREC-2004*, 2004.

[10] Nick Craswell, David Hawking, Ross Wilkinson, and Mingfang Wu. Overview of the trec 2003 web track. In *TREC*, pages 78–92, 2003.

[11] Taher Haveliwala and Sepandar Kamvar. The second eigenvalue of the google matrix. Technical Report 20, Stanford University, 2003.

[12] Taher H. Haveliwala. Topic-sensitive pagerank: A context-sensitive ranking algorithm for web search. *IEEE Trans. on Knowl. and Data Eng.*, 15(4):784–796, 2003.

[13] David Hawking and Paul Thistlewaite. Relevance weighting using distance between term occurrences. Technical Report TR-CS-96-08, Department of Computer Science, The Australian National University, 1996.

[14] D Hiemstra. A probabilistic justification for using tf.idf term weighting in information retrieval. *International Journal on Digital Libraries*, 3(2):131–139, 2000.

[15] Glen Jeh and Jennifer Widom. Scaling personalized web search. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 271–279, New York, NY, USA, 2003. ACM.

[16] K. Sparck Jones, S. Walker, and S. E. Robertson. A probabilistic model of information retrieval: development and comparative experiments. *Inf. Process. Manage.*, 36(6):779–808, 2000.

[17] K. Sparck Jones, S. Walker, and S. E. Robertson. A probabilistic model of information retrieval: development and comparative experiments part 2. *Inf. Process. Manage.*, 36(6):809–840, 2000.

[18] Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28:11–21, 1972.

[19] Sepandar Kamvar, Taher Haveliwala, Christopher Manning, and Gene Golub. Exploiting the block structure of the web for computing pagerank. Technical Report 2003-17, Stanford InfoLab, 2003.

[20] E. Michael Keen. Term position ranking: some new test results. In *SIGIR '92: Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 66–76, New York, NY, USA, 1992. ACM.

[21] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632, 1999.

[22] R. Lempel and S. Moran. Salsa: the stochastic approach for link-structure analysis. *ACM Trans. Inf. Syst.*, 19(2):131–160, 2001.

[23] M. E. Maron and J. L. Kuhns. On relevance, probabilistic indexing and information retrieval. *J. ACM*, 7(3):216–244, 1960.

[24] David R. H. Miller, Tim Leek, and Richard M. Schwartz. A hidden markov model information retrieval system. In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 214–221, New York, NY, USA, 1999. ACM.

[25] J. A. Nelder and R. Mead. A Simplex Method for Function Minimization. *The Computer Journal*, 7(4):308–313, 1965.

[26] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999. Previous number = SIDL-WP-1999-0120.

[27] Jay M. Ponte and W. Bruce Croft. A language modeling approach to information retrieval. In *SIGIR '98: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 275–281, New York, NY, USA, 1998. ACM.

[28] M. F. Porter. *An algorithm for suffix stripping*, pages 313–316. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997.

[29] Yves Rasolofo and Jacques Savoy. Term proximity scoring for keyword-based retrieval systems. In *Advances in Information Retrieval: 25th European Conference on IR Research, ECIR 2003, Pisa, Italy, April 14-16, 2003. Proceedings*, page 79, 2003.

[30] S. E. Robertson and S. Walker. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *SIGIR '94: Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 232–241, New York, NY, USA, 1994. Springer-Verlag New York, Inc.

[31] Stephen Robertson, Hugo Zaragoza, and Michael Taylor. Simple bm25 extension to multiple weighted fields. In *CIKM '04: Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 42–49, New York, NY, USA, 2004. ACM.

[32] Stephen E. Robertson and Karen Sparck Jones. *Relevance weighting of search terms*, pages 143–160. Taylor Graham Publishing, London, UK, UK, 1988.

[33] Stephen E. Robertson, Steve Walker, and Micheline Hancock-Beaulieu. Okapi at trec-7: Automatic ad hoc, filtering, vlc and interactive. In *TREC*, pages 199–210, 1998.

[34] Hans-Paul Paul Schwefel. *Evolution and Optimum Seeking: The Sixth Generation.* John Wiley & Sons, Inc., New York, NY, USA, 1993.

[35] Amit Singhal, Chris Buckley, and Mandar Mitra. Pivoted document length normalization. In *SIGIR '96: Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 21–29, New York, NY, USA, 1996. ACM.

[36] John A. Tomlin. A new paradigm for ranking pages on the world wide web. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 350–355, New York, NY, USA, 2003. ACM.

[37] Adam D. Troy and Guo-Qiang Zhang. Enhancing relevance scoring with chronological term rank. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 599–606, New York, NY, USA, 2007. ACM.

[38] Trystan Upstill. *Document ranking using web evidence.* PhD thesis, Australian National University, 2005.

[39] Ellen M. Voorhees and Donna Harman. Overview of the eighth text retrieval conference (trec-8). In *TREC*, 1999.