Improving Search Engines via Classification

Zheng Zhu

May 2011

A Dissertation Submitted to
Birkbeck College, University of London
in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy

Department of Computer Science & Information Systems
Birkbeck College
University of London

Declaration

This thesis is the result of my own work, except where explicitly acknowledge in the text.

Zheng Zhu

Abstract

In this dissertation, we study the problem of how search engines can be improved by making use of classification. Given a user query, traditional search engines output a list of results that are ranked according to their relevance to the query. However, the ranking is independent of the topic of the document. So the results of different topics are not grouped together within the result output from a search engine. This can be problematic as the user must scroll though many irrelevant results until his/her desired information need is found. This might arise when the user is a novice or has superficial knowledge about the domain of interest, but more typically it is due to the query being short and ambiguous.

One solution is to organise search results via categorization, in particular, the classification. We designed a target testing experiment on a controlled data set, which showed that classification-based search could improve the user's search experience in terms of the numbers of results the user would have to inspect before satisfying his/her query. In our investigation of classification to organise search results, we not only consider the classification of search results, but also query classification. In particular, we investigate the case where the enrichment of the training and test queries is asymmetrical. We also make use of a large search engine log to provide a comprehensive topic specific analysis of search engine queries. Finally we study the problem of ranking the classes using some new features derived from the class.

The contribution of this thesis is the investigation of classification-based search in terms of ranking the search results. This allows us to analyze the effect of a classifier's performance on a classification-based search engines along with different user interaction models. Moreover, it allows the evaluation of class-based ranking methods.

Publications

Publications Relating to the thesis

1. Zheng Zhu, Ingemar J Cox, Mark Levene: **"Ranked-Listed or Categorized Results in IR: 2 Is Better Than 1"**. The 13th international conference on Natural Language and Information Systems, 2008, pages 111-123, London, United Kingdom - Chapter 3

2. Judit Bar-Ilan, Zheng Zhu and Mark Levene: **"Topic-specific analysis of search queries"**. The 2009 workshop on Web Search Click Data (WSCD '09), pages 35-42, Barcelona, Spain Extended version appeared as Judit Bar-Ilan, Zheng Zhu and Mark Levene: **"Topical Analysis of Search Queries "**. in the 10th Bar-Ilan Symposium on the Foundations of Artificial Intelligence - Chapter 4

3. Zheng Zhu, Mark Levene, Ingemar J Cox: **"Query classification using asymmetric learning "**. The 2nd International Conference on the Applications of Digital Information and Web Technologies, 2009, pages 518-524, London, Unite Kingdom - Chapter 4

4. Zheng Zhu, Mark Levene, Ingemar J Cox: **"Ranking Classes of Search Engine Results"**. The International Conference on Knowledge Discovery and Information Retrieval, 2010, Valencia, Spain - Chapter 5

Acknowledgements

I am deeply grateful to my supervisors, Prof. Mark Levene and Prof. Ingemar J Cox, for their continued support, their patient guidance and their faith in my work throughout these years. Whenever I needed any help, their doors were always open.

Many thanks are due to my colleagues at Birkbeck, University of London. Without their collaboration and discussions, I can not have finished our research work. I would particularly like to thank Dell Zhang, Rajesh Pampapathi, Phil Gregg, Aisling Traynor and Simon Upton for their help and support.

Dr. Iadh Ounis and Dr. Tassos Tombros gave me critical comments during the defense, which will make me to think a lot more deeply about my work from here on in.

Special thanks are due to Dr. Kai Yu and Sean Zhang. Their encouragement and suggestions guided me towards starting my Ph.D in the very beginning.

I would also thank Martyn Harris for the final proofreading.

Last but not least,I would like to declare my gratitude and love to my wife Lei Zhu and my parents YongGang Zhu, XiangYa Zheng. Their enthusiasm and constant encouragement with their patience accompany me through this period and every other period. I would never have finished this work without their support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Current State of the Art in Search

Searching is an inherent instinct of human behavior. Every search activity is accompanied by a goal. We all have such experiences, for example, you may be looking for a restaurant to have lunch or searching for the best deal to buy a new mobile phone. A good search process can facilitate the attainment of such goals. In this thesis, we will discuss the search process over information stored in digital media.

A related scenario of our topic is searching for books in a library. Librarians have a long history of organising books so that the user can easily identify his/her book of interest using a library catalogue, facilitated by a library card system. The library card can be regarded as metadata describing the title, the author and the category the book belongs to. This is a traditional model of searching for information.

Since the World Wide Web was invented at the end of the last century, information has experienced exponential growth. Twenty years after the inception of the Web, information seeking on the Web has become one of the most important parts in our daily lives. In contrast to traditional information search in a library, the Web is not a physical "entity" that exists in a specific "place".

It is a virtual "space" in which information can exist. Furthermore, the World Wide Web provides a digitalised format for documents, which can easily be accessed and analyzed by its users. With the popularity of the World Wide Web, searching via the Web, or the so called "information retrieval on the Web" has become an important subarea of Computer Science.

One of the core characteristics of the Web is that it is a distributed, public information space that is freely accessible; there is no central authority to control the distribution of content, which results in some major challenges for information seeking, such as maintaining the quality of information. Therefore, to perform a search over the Web requires mature computational tools. Amongst these, two approaches have been widely adopted.

1. Web directory services, and
2. Search engine services.

A web directory organises web content in a hierarchical structure, which is analogous to the traditional library system, where each document belongs to one or more topics and subtopics. If the user wants to search for certain information related to a single topic, he/she only requires to seek the documents within the category they are interested in rather than searching through all available information. However, the main limitation of web directories is obvious: they are expensive to build and have low coverage. They generally require humans to compile the list, and thus can only cover a small portion of the whole Web. For example, the Open Directory Project [1] contained 4,529,334 web sites as of 2010. However, according to [60], the size of the searchable Web indexes acquired at least 11.5 billion pages by the end of January 2005. Google even claimed that their index reached the size of 1 trillion in July 2008 [2]. Searching through browsing is also limited from a usability perspective, as the user may not be familiar with his/her target category.

Search engines are web portals for finding information on the Web. They index a large portion of the Web and store the information in databases. Unlike a

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

[2]`http://googleblog.blogspot.com/2008/07/we-knew-web-was-big.html`

web directory, the operation of a search engine is carried out automatically. The underlying technology of search engines is information retrieval. The obvious question in information retrieval is how to find the relevant documents for each user query. This is the main issue we try to address (we will discuss retrieval models in Chapter 2). In general, search engines seem to successfully attract a significant amount of search activities. According to comScore, 13.8 billion searches were performed in September 2009 in America, and approximately 460 million web queries were submitted daily, which implies that on average, 5,324 queries were carried out on search engines every second.

Given a user query, search engines typically return a ranked list of web documents. The rank is measured by relevance or the probability of relevance, which is attained by a search engine retrieval model. The first conventional search engines were keyword-based and relevance was determined by the frequency the search terms occurred in the page. Since there is no central authority to control the distribution of the documents, this caused a serious bias for ranking a web page, for example, due to search engine spam. As a result, Google adopted the analysis of human-generated links with the use of PageRank [94] to overcome this problem. It assumes that web pages linked from many important pages are also likely to be important, and each link contributes a vote to the linked page regarding its importance. PageRank is thought to correlate well with human concepts of importance. In addition to PageRank, Google also uses over 200 features to calculate the relevance score of a web page. A rich model together with many relevant features contribute to Google's success in web search market.

Apart from the above approach, there is another machine learning technique called learning to rank [86], which has been gaining much attention recently. The idea is to make use of the user search log, and analyse the correlation between the query and clickthrough data, in order to learn the connection between them, and produce a new ranking model matching the user's information need. The underlying assumption in this case is that the clickthrough data should be ranked higher than the non-clickthrough data. However, a clickthrough is a coarse indicator of relevance, and other refined features can also be integrated into the learning algorithm, such as the duration time the user stays on a page, and the position of the clickthrough [2]. The learning to rank approach improves the performance of ranking algorithms and, as far as we know, Microsoft has adopted this method within Bing, their flagship search engine.

1.2 What is Lacking in Search

Search is not all about discovering information, it is a means to an end. In order to provide a better service to its customers, it is necessary to conduct exploratory conversations to better understand the customers' requirement and the reason behind that requirement. The customer who looks for a local map may be on the way to a tube station, which may be closed for maintenance. In this case, finding the information need does not help for his/her final goal. To achieve the final goal via searching is a challenging issue, in this thesis, we decompose this problem into small solvable subproblems and we will investigate one approach to a single subproblem.

On the web, a user will typically express his/her information need with only three terms or less, and thus due to a gap in knowledge, the searcher is often not able to generate the correct query. Therefore, several search engines have allowed the user to specify his/her domain of interest or to describe his/her interest in a profile [56] to assist the search. Such approaches are domain limited, non-universal and not every user will be able or willing to provide such information.

One way to address the above issues is federated search. Commercial search engines achieve a broad coverage, but nevertheless there is an absence of 'deep web' information, which refers to the vast number of online databases and other web resources that are not directly accessible by traditional search engines. Moreover vertical search engines have the capability of understanding specific domains much better than general search engines and returning more relevant results for domain related queries. By combining different databases with deep knowledge, federated search may overcome the limitation of commercial search engines.

Yet another approach to (partially) address the above issue is to provide advanced visualisation [65], or presentation of search results. Commercial search engines return a ranked list of results, and this has been proven to be one of the most effective methods; however, it is not the only way to present search results. Grouping similar results into categories is another way to present the results. The intuition is that the user can identify the category he/she is interested in. With this method, many irrelevant results can be filtered out, and to support this assumption, researchers had demonstrated the benefits of such a strategy [66, 31].

To summarise, in order to achieve an enhanced user search experience, users prefer to go through smaller result set [93]. Firstly, for informational queries, rather than search through many results within a variety of topics, it is desirable to return a small number of focused and relevant results; this can be achieved by focusing on one domain-specific source of information related to the query rather than the whole web [8]. However, it is challenging to find the right domain to perform the search over without having background knowledge of the user's specific information need. In such cases where users may be satisfied with one relevant result, it is preferable to present top results covering as many topics as possible so that it can meet the requirements of the user population. This leads to result diversification, which we will briefly discuss in Chapter 2. Secondly, there is the approach of grouping similar results together and assigning the class labels to them so that the user can ignore the irrelevant categories and search in a focused subset of results. In this thesis, we will focus on the second approach of grouping, and more specifically on the automatic classification of search results.

1.3 Classification as a Way to Organise Search Results

Organising search results into groups can facilitate the user's search behavior. This is related to the cluster hypothesis. The cluster hypothesis proposed by van Rijsbergen [117] can be stated as follows:

Closely associated documents tend to be relevant to the same request.

There are several criteria to facilitate grouping, depending on the definition of "closeness". Different definitions have led researchers to take different directions within the information retrieval community. When "closeness" refers to data point distribution, i.e., common term co-occurrences in search results, it is called *clustering*.

Although the cluster hypothesis does not explicitly mention clusters, "closely associated" documents tend to exist in the same cluster.

In this thesis, we will address search result grouping from another perspective, i.e., from that of classification. The classification process is a technology with a long history and is applied widely in practice. More details about classification algorithms will be given in Chapter 2. Librarians applied this methodology in library systems successfully. Supermarkets also organise goods according to a classification structure to facilitate their customers in item finding. According to the cluster hypothesis, undoubtingly the clusters and classes can be beneficial to web users. The terms "closely associated" in the cluster hypothesis can be understood as belonging to the same topic of the results assigned by the classifier [117]. In this thesis, we investigate the advantage of the classification quantitatively in both synthetic and realistic cases.

We believe that classification and clustering are two different approaches for tackling the same problem. There is no simple answer to the question which is better. The discrepancy not only lies in the algorithmic aspect, but also in the problem setting. In general, clustering is a light-weight approach as it usually does not require training data. However, in order to achieve improved clustering results, in particularly the problem of assigning reasonable cluster labels, additional information may be necessary [120]. Conversely, classification requires training data associated with an explicit class ontology. In order to produce a reasonable classifier, sufficient training data is needed. However it avoids the problem of label generation. Another issue is how deep should the class ontology be. This is particularly important for specific queries whose results are contained in a small number of classes. In such cases, one layer of class structure may not be sufficient. Two possible approaches exist to address this issue. First, we can adopt some hierarchical structure as a Web directory and train a multi-level classifier [122]. This approach has been applied in query classification [23]. Secondly, we can adopt a hybrid system, i.e., in the second layer, we could use clustering technology. This thesis will mainly focus on ambiguous queries and adapt a flat class structure.

1.4 Outline of the Thesis

This thesis is organised as follows. Chapter 2 introduces the background knowledge about information retrieval and machine learning needed for the thesis. Foundational concepts and the tasks of information retrieval are described. The main issues faced in information retrieval are discussed, followed by retrieval models and evaluation criteria. In the machine learning section, we describe three components: the input space, the output space and the hypothesis space. Here we mainly focus on the input space and the hypothesis space. Although in recent years the output structure has become popular, it is beyond the scope of the thesis to deal with this aspect in detail. Supervised learning and unsupervised learning are also studied. For the former, we mainly consider the k-nearest neighbour classifier and Support Vector Machines (SVM). For the latter, we simply explain k-means algorithm and agglomerative clustering. We conclude Chapter 2 with applications of supervised and unsupervised learning in information retrieval.

Chapter 3 introduces a model for a classification-based search engine. We first explain the problems encountered in traditional search engines, then present a framework for a classification-based search engine, and finally define some related concepts. Following that we describe our experimental methodology and experiment design. The experimental results are presented, which compare classification-based systems to the traditional ranked-list systems.

Query classification is studied in Chapter 4, which is related to one of the key questions faced by every search engine, i.e., understanding the user query. We review some approaches to tackle the different aspects of this problem. The main problem we study is the impact of the classifier, when the training and test data are asymmetric. This scenario occurs often, as offline training data can be rich, while online test data is usually limited. Experimental results are then reported. Thereafter, we apply the trained classifier to a large MSN search engine query log. Topic specific analysis of search queries is also performed and reported on.

When we apply the classification to search results, we derive the new features for each class. In Chapter 5, we investigate the effect of these new features on ranking classes. We then review the related methods from the literature, and present a model for ranking classes. We suggest six ranking methods derived from this model. Experiments are conducted and the class-based ranks are compared with the traditional list ranks.

Finally, in Chapter 5, a Proof of Concept for a classification-based search engine is presented next. Some practical implementation issues are discussed and a prototype search engine is also described.

In Chapter 6, we summarise the contributions and discuss future work arising from the thesis.

In this thesis, we address the problems arose from ambiguous queries. Classification is presented as one approach to handle this problem. Ranking classes is also discussed. The main novelties of our work are as follows:

- We propose several class-based rank methods in Chapter 3. Previous user studies confirmed that a category based interface can improve user search in terms of search time. In this case users may be satisfied with one relevant result, such ranking criteria are correlated with users' search time.

- We propose six user interaction models to describe the user search behavior when they are using a category based interface in Chapter 3.

- In Chapter 3 we analyze the correlation between the performance of the classifier and the performance of a classification-based IR system. To our knowledge, this research has not been carried out before.

- In Chapter 4, we investigate the query classification problems by making use of various enrichment strategies, which leads to topic specific analysis of search queries.

- We propose a class ranking model based on Bayesian theory in Chapter 5, and investigate it by using a query log, therefore our results were drawn from an analysis of medium scale data.

Our work has its limitation, as we mainly focus on a known-item search task, i.e., the queries seek a single relevant web page. This is one type of task on the web search scenario. In practise, search task can be more complex than the known-item search. For example, some queries may have many relevant results. But our work does not cover this aspect. Our experiments did not involve any user testing. However, it would be useful to conduct a user test and to compare it with the known-item test carried out in this thesis.

Chapter 2

Background

2.1 Web Information Retrieval

Searching over the Web is becoming a daily activity in current society for many people. This activity is typically carried out via search engines. The underlying science of search engines is based on *Information Retrieval* (IR). One definition of information retrieval might be:

Information retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers) [36].

From this definition, we can see that a basic information retrieval problem is as follows. There exists a document collection stored on a computer, which is the set of documents searching will be performed over. A *document* consists of a sequence of words, for example an email message, a Microsoft Word document or a single web page. A user expresses his/her specific information need in a few terms, which is called a *query*, to communicate with the computer that retrieves the documents. A *term* is the smallest unit used in information retrieval and it is normally a word. An information need is the topic the user desires to know more about, and the retrieved documents constitute the result set. The relevant documents are what the user is looking for, which contains valuable information to the user with respect to his/her personal information need.

The task of information retrieval is to retrieve search results in response to the user query and present them in order of their relevance to the user's request, therefore helping users to find things faster and satisfying their information needs as quickly as possible. When relevance is interpreted as a probability, the ranking of results is called the probability ranking principle from a probabilistic perspective [103]. Robertson [103] and Van Rijsbergen [117] have demonstrated that such a principle leads to minimal decision-theoretic loss associated with retrieving a set of n top-ranked documents.

Research in information retrieval can be dated back to 1950's. Since then, the application of information retrieval has moved from library systems to general purpose search engines. Commercial search engines have existed since the inception of the Web and have had a huge impact on our society. However, there are still some unresolved issues. One important issue is that of the interpretation of relevance. The concept of relevance serves as the foundation for the field of information retrieval. Generally speaking, relevance is a representation of the user's informational requirement, a reflection of what they are seeking. However, as a scientific concept it is required to have a strict formalism. Researchers have studied relevance from a variety of perspectives, for example, when the relevance is exclusively determined by the relation between a set of keywords of query terms and the keywords set from documents, this is called *system-oriented relevance*. Such an approach includes BM25 [102]. In contrast to *system-oriented relevance*, *user-oriented relevance* introduce the user's state of knowledge into the definition. It is claimed that the nature of relevance to a large degree depends on the person who is making the judgement [77]. In most definitions, relevance is regarded as a binary decision, i.e., either relevant or not. Cooper treated relevance as a real number and applied a measure of utility to the particular task [40]. Zhai [127] extended this view by modeling the user preferences as loss functions in a probabilistic framework, thus the whole retrieval process was re-casted as a risk minimisation problem.

Let us review the user's search process according to [90]; the user is in a "problematic situation", that needs information for being solved. The Real Information Need (RIN) defines the information that will truly help the user solve the problem. However, the user might not be fully aware of what constitutes his/her real information need in the first stage, instead he perceives the RIN and builds the Perceived Information Need (PIN). Then the user will express the PIN in a request, usually in natural language, and finally they will formalise the request in a query, a representation of the request in a 'system' language, which can be recognised by a search engine. This procedure is represented in Figure 2.1. The above process is usually required to perform iterations as the

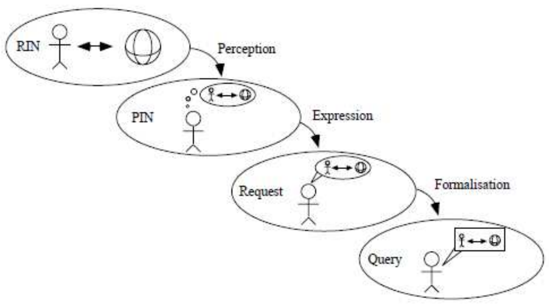


Figure 2.1: Real information need, perceived information need, request, and query [90]

user refines his/her query based on the current state of knowledge. During the search process, the user typically expresses his/her request in two to three terms, known as the user's request or query; however one concept can be expressed in a variety of distinct ways. So simply comparing query terms to the representation of documents will result in many relevant documents being missed, due to the absence of the query terms in that document, this is called the term mismatch problem. We will discuss some possible solutions to this problem shortly. Another problem is related to *polysemy*, particularly in the absence of a context. This is due to the fact that a query typically comprises of only two to three terms and this will result in vague concepts or ambiguity in terms of the user information need. One example is the query "jaguar team"; we cannot distinguish whether the information the user requires is regarding the Jaguar car team or the Jacksonville Jaguar football team. This kind of relevance is called "*topical relevance*". This thesis mainly focuses on improving topical relevance.

In contrast to "*topical relevance*", "*user relevance*" refers to the user's subjective judgement. For example, given a query "Samsung CLP-310 Colour Laser Printer", one user may be interested in the price of the printer, while others may look for the drivers of the printer. Not only may the users convey different information needs using the same or similar queries, but the users may interpret relevance differently based on their individual knowledge and the state of this knowledge.

To achieve the goal of information retrieval, researchers have proposed a variety of models in the last sixty years. A good retrieval model should find the relevant documents meeting the user's information need. We will discuss retrieval models in Section 2.1.1.

Another important issue is the evaluation of an information retrieval system. The user expresses his/her information need as the user's query to communicate with the system. However, we know there is a gap between the user's query and the user's information need. The user's query is a representation of the underlying information need in the form of keyword list and is observable and readily available to the system and other users. Moreover, the information need is an abstract concept which is only fully understood by the user who issued the query. The retrieved documents may perfectly match the user's query on the subject level, but they may not satisfy the user's underlying information need. Therefore, relevance is user-dependent in practice; one more straightforward way is to conduct user testing when comparing the retrieval models. To evaluate relevance, user testing is typically carried out on a small scale as it is expensive and time consuming. Search engine log data can be an alternative source for evaluation by assuming clickthroughs are more relevant than non-clickthroughs. The problem of search log data is that it is biased towards highly ranked documents. Nevertheless, it is a valuable and large resource. Another method to evaluate an IR system is to make use of a standard test collection comprising of selected documents, together with relevant judgements for the known information needs. In addition to the design setting, the evaluation metrics are essential for the evaluation process. The two basic measures are precision and recall. We will discuss them in Section 2.1.2.

When the user specifies his/her information need through a query which initiates a search (executed by the information system) for documents which are likely to be relevant to the user, this retrieval task is called ad-hoc retrieval [10]. A search engine is a kind of ad-hoc retrieval machine. If the corpus is dynamic, e.g. news web sites, and the user query is static, this is called filtering. The goal of filtering is to detect and provide information based on the user's preference; one example is that of topic-based RSS feeds. Other applications related to IR include categorisation and query answering systems. Categorisation assigns a set of pre-defined labels to documents. The Open Directory Project is an example of categorisation. Query answering is similar to ad-hoc retrieval, but it provides more specific results in response to the user query, than a list of ranked documents.

Although we will mainly focus on text retrieval, and particularly on web document retrieval, there are other aspects of information retrieval, such as image retrieval and video retrieval.

2.1.1 Retrieval Models

Before we discuss retrieval models, we will first introduce some notation. We assume there is a document collection $D = \{D_1, D_2, ...D_N\}$, where D_i is a text document; and we let $T = \{t_1, t_2, ...t_M\}$ be the vocabulary set and t_i is a term. We denote a document as $D_i = (d_{i1}, d_{i2}, ...d_{in})$ and a query as $Q = (q_1, q_2, ...q_m)$, where d_{ij} and q_j represent the terms occurring in the document and query respectively, that are the subsets of T. Conventional retrieval models seek a function $R(D_i, Q)$, where the result of this function implies the relevance level of the document in response to the query. The models can be roughly categorised as query-dependent models and query-independent models.

2.1.1.1 Query-Dependent Models

These models are based on the presence of query terms in the documents. One of the earliest such models is the boolean retrieval model, also known as the exact-match retrieval model. Boolean models will only make a binary decision about the relevance of the document by matching the query specification to the document. It does not involve any ranking as it assumes the retrieved set are equivalent in terms of relevance. Thus, the absence of the degree of relevance is the main limitation of this model. However, it has recently been recapturing researchers' interests again, with the proposal of faceted search, which provides more effective information-seeking support to users than best-first search by combining faceted navigation with text search [116]. To model the relevance degree, the Vector Space Model [108] has been proposed.

2.1.1.2 Vector Space Model

The Vector Space Model (VSM) has become a standard tool in IR systems since its introduction over three decades ago [16]. One of the advantages of the VSM is its support of relevance ranking with respect to queries over documents of heterogeneous format.

In the Vector Space Model, a vector is used to represent each document in a collection. Each component of the vector represents a term that occurred in the document. The value of a component indicates the importance of the term within the given document. Thus this formulation can be used to measure the similarity between two vectors, one representing a document, and another representing a query.

Here we need to determine a weight (value) for each term (In the Boolean model, the weight is 1 if the term is present, otherwise 0). If one term occurs multiple times in a document, it indicates that an increased significance of this term to the document is more likely. So incorporating the *term frequency* into weights seems a straightforward form of a weighted representation. We use $tf_{t,d}$ to denote the frequency of a term t occurring in a document d. Term frequency can reflect how well the term describes the document, but it cannot reflect the importance of the term in the collection of documents. Thus, we define the *inverse document frequency* of a term t and a scaling of its weights as follows:

$$idf_t = log\frac{N}{df_t}$$

where N is the total number of documents in the collection, and the document frequency df_t is the number of documents containing the term t. From this definition we can see that the more documents that a term occurs in, the less discriminative power the term possesses, and the less useful it is for a retrieval model.

One formula of document term weighting schemes in the Vector Space Model is given in Equation 2.1[1] as follows:

$$d_{ij} = tf(j,i) * log\frac{N}{df_j} \tag{2.1}$$

In this model, a document can be thought of as one point in a high dimensional space, the same as the query. The standard approach to quantify the distance between two vectors is by Euclidean distance (Equation 2.21). However, it suffers from a drawback: two documents that share similar content can differ significantly, simply because one document is much longer than the other and tends to have large term frequencies. To compensate for this problem, *cosine similarity* between two document d_i, d_j is commonly used as follows:

$$sim(\mathbf{d_i}, \mathbf{d_j}) = \frac{\mathbf{d_i} \cdot \mathbf{d_j}}{|\mathbf{d_i}||\mathbf{d_j}|} \tag{2.2}$$

So the relevance score can be measured as:

$$R(D_i, Q) = sim(\mathbf{d_i}, \mathbf{q}) \tag{2.3}$$

VSM makes the assumption of independence between terms, and it suffers from the term mismatch problem we mentioned earlier, as it cannot capture the semantic information between two or more related terms or between related documents. The generalised VSM (GVSM) has been proposed to solve the semantic similarity problem.

The main idea of GVSM is that if two terms are semantically related, they will co-occur in the same documents frequently. The document vector can be represented by

$$\mathbf{d_i} = \mathbf{d_i^T} \mathbf{D}^T, \tag{2.4}$$

[1]There exists other forms which scale tf-idf factor differently, i.e.,$d_{ij} = \frac{(log(tf(j,i))+1)log(N/df_j)}{\sqrt{\sum_{k=1}^{t}[(log(tf(k,i))+1)log(N/df_k)]^2}}$

where **D** is the document-term matrix of the collection. So the relevance score is given by:

$$R(D_i, Q) = \mathbf{d}_{\boldsymbol{i}}^T \mathbf{D}^T \mathbf{D}\mathbf{q} \tag{2.5}$$

Machine learning can also be employed to learning the term correlation matrix W, and the score function is

$$R(D_i, Q) = \mathbf{d}_{\boldsymbol{i}}^T \mathbf{W}\mathbf{q} \tag{2.6}$$

Here **W** is the term correlation matrix that can be learnt using polynomial semantic indexing [11]. Other approaches to handle the term mismatch problem include semantic networks [51] and Latent Semantic Indexing [44].

2.1.1.3 Other Query-Dependent Models

In addition to the Vector Space Model, there are other popular models used in information retrieval, such as the probabilistic model. Given a query, the probability of a document being relevant to the query, can be regarded as a classification problem. The most well known probability model is BM25 [102]. On the other hand, Language Models [97] directly model the likelihood of the document to generate the query. A survey of these models can be found in [59] and [36].

2.1.1.4 Query-Independent Models

There are other models whose ranking are based on features, which are query independent. Amongst them, the most successful model is PageRank [94], which measures the importance of each page based on link structure derived from web pages. Furthermore, there are many other link analysis methods, such as TrustRank [61].

2.1.2 Evaluation Metric

In the previous sections, we introduced various retrieval models for an IR system. In order to compare such models, evaluation metrics of effectiveness need to be defined.

The most common effectiveness measures are precision and recall, which were introduced in the Cranfield study [119]. Given a document collection and a few queries, it is assumed that assessors can provide relevance judgements, mapping each query to all the relevant documents in the collection, which is practical in small collections. To compare the performance, those queries are issued to the competing retrieval models, result sets are respectively retrieved, and the results can be either relevant or non-relevant. We denote the relevant set of documents for the query by A, all retrieved set of documents for the query by B, $\bar{A}$ by the non-relevant set and $\bar{B}$ by the non-retrieved set.

	Relevant	Non-Relevant
Retrieved	A ∩ B	$\bar{A}$ ∩ B
Non-Retrieved	A ∩ $\bar{B}$	$\bar{A}$ ∩ $\bar{B}$

Table 2.1: Different possible outcomes in the collection.

If we regard the Relevant as Positive and Non-Relevant as Negative, A ∩ B is *true positive* (TP) as the retrieved results are relevant, $\bar{A}$ ∩ B is *false positive* (FP) because the retrieved results are non-relevant, A ∩ $\bar{B}$ is *false negative* (FN) as relevant data is considered non-relevant by the system and $\bar{A}$ ∩ $\bar{B}$ is called *true negative* (TN) because non-relevant data is correctly recognized by the system. Precision and recall are defined as:

$$Precision = \frac{|A \cap B|}{|B|} = \frac{|TP|}{|TP| + |FP|} \tag{2.7}$$

$$Recall = \frac{|A \cap B|}{|A|} = \frac{|TP|}{|TP| + |FN|} \tag{2.8}$$

In this scenario, a retrieval model can be viewed as a binary classifier which can distinguish between relevant and non-relevant results from the whole collection. Precision and recall are widely used in practice when evaluating the performance of a binary classifier.

For the multi-class problem, microaveraged and macroaveraged metric are used. The microaveraged precision is defined as:

$$Microprecision = \frac{\sum_{i=1}^{|C|} |TP|_i}{\sum_{i=1}^{|C|} |TP|_i + \sum_{i=1}^{|C|} |FP|_i}, \tag{2.9}$$

The microaveraged recall is defined as:

$$Microrecall = \frac{\sum_{i=1}^{|C|} |TP|_i}{\sum_{i=1}^{|C|} |TP|_i + \sum_{i=1}^{|C|} |FN|_i}, \tag{2.10}$$

where $|C|$ is the number of the classes, $|TP|_i$ is the number of *true positives* for positive class i, $|FP|_i$ is the number of *false positives* for positive class i and $|FN|_i$ is the number of *false negatives* for positive class i.

The definition of macroaveraged is as follows:

$$Macroprecision = \frac{\sum_{i=1}^{|C|} Precision_i}{|C|}, \tag{2.11}$$

$$Macrorecall = \frac{\sum_{i=1}^{|C|} Recall_i}{|C|}, \tag{2.12}$$

Here precision and recall are defined in the usual way. The difference between microaveraged and macroaveraged metric is that for the former, it gives equal weight to each per-document classification decision, whereas macroaveraged gives equal weight to each class. Therefore, for microaveraged, the accuracy can be poor for classes with few positive examples without affecting the overall numbers much.

In certain cases, a single metric is used to summarise the overall performance of the system. The F measure proposed by Jardine and Rijsbergen [91] measures the harmonic mean of precision and recall, which is defined as follows.

$$F_\beta = \frac{(\beta^2 + 1)RP}{(R + \beta^2 P)}, \tag{2.13}$$

where R and P represent recall and precision respectively, and β is a parameter between 0 to 1. In practice, the most common F_β measure is F_1, i.e.,

$$F_1 = \frac{2RP}{R + P} \tag{2.14}$$

Similarly, the microaveraged and macroaveraged F1 is defined as follows:

$$MicroF1 = \frac{2 \times Microprecision \times Microrecall}{Microprecision + Microrecall}, \tag{2.15}$$

$$MacroF1 = \frac{\sum_{i=1}^{|C|} F1_i}{|C|}, \tag{2.16}$$

Both recall and precision are defined with respect to a set of retrieved results.

Another common evaluation metric is accuracy. The definition of it is as follows:

$$Accuracy = \frac{|TP| + |TN|}{|TP| + |FP| + |TN| + |FN|} \tag{2.17}$$

A further measure of document retrieval system performance is called "expected search length" [39]. Given a list of ranked results, search length measures the number of irrelevant results until the first relevant result is encountered. Such an ordering is called "simple ordering" by William Cooper. Nevertheless, a retrieved set of documents can be divided into multiple levels, each level contains a subset of documents. For example, if a query contains 5 words, one subset could be the documents having all 5 words, another subset could be the documents containing only 4 words and so on. If we assume the documents within each level are randomly ordered, expected search length is defined as the average number of documents that must be examined to retrieve a given number of relevant documents.

For a single search engine, the number of retrieved documents can be extremely large. So we need to choose a cut-off point to compute the precision [2]. For example, precision at top 10, measures the accuracy within the top 10 retrieved documents. However, it does not take the document position into account. A model which ranks relevant documents in higher positions is better than a model that ranks them in the lower positions. However, precision at the top 10 can not distinguish between such models. To compare two ranked lists more accurately, average precision is proposed as follows:

$$averprec = \frac{1}{k}\sum_{i=1}^{k} Precision(r_i), \tag{2.18}$$

where r_i is the set of ranked retrieved results from the top ranks until document d_i and $Precision(r_i)$ is the precision in the top i results.

In recent years, mean average precision (MAP) has been used in many research papers, which is the mean of average precision over all queries in the test set. Other measures include the normalised discounted cumulative gain (NDCG) [69], which is based on two assumptions:

[2] It is usually impossible in practice to measure recall because there is lack of knowledge of the full relevant set for one query.

- Highly relevant documents are more useful than marginally relevant documents.
- The lower the ranked position of a relevant document, the less useful it is for the user, since it is less likely to be examined.

The discounted cumulative gain (DCG) is the total gain accumulated at a particular rank p. It is defined as:

$$DCG_p = rel_1 + \sum_{i=2}^{p} \frac{rel_i}{log_2 i} \tag{2.19}$$

where rel_i is the graded relevance level of the document retrieved at rank i [24]. For the perfect ranking of one given query, the DCG value will be maximized. This maximized DCG value is called ideal DCG value. The normalised discounted cumulative gain (NDCG) for a given query can be defined as:

$$NDCG_p = \frac{DCG_p}{IDCG_p} \tag{2.20}$$

where IDCG is the ideal DCG value for the query.

2.1.3 Text Analysis Process

In the previous section, we introduced retrieval models and evaluation metrics. In retrieval models, term vectors are used to represent documents and queries, which are associated with a text corpus. In addition, we describe the use of term frequency and inverse document frequency to compute the weight associated with each term. In this section, we will simply go through the text analysis process, which will be encountered in the implementation of information retrieval systems.

A typical text analysis process is shown in Figure 2.2.

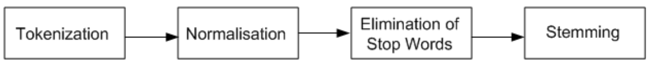


Figure 2.2: Text analysis process

From Figure 2.2, we can see that there are four steps involved in the text analysis process:

1. Tokenization - Parse a stream of text into meaningful elements, called tokens (terms).
2. Normalisation - Convert the terms to lowercase.
3. Elimination of stop words - Eliminate terms that appear very often and do not have any discriminative ability, such as “the”, “a”.
4. Stemming - Reduce the terms to their root form, e.g. remove plurals and strip off suffixes.

Many stemming algorithms exist, and in our work, we use the snowball stemmer (`http://snowball.tartarus.org/texts/introduction.html`).

2.2 Machine Learning

Machine learning can be mainly divided into three topics: supervised learning (classification and regression), unsupervised learning and reinforcement learning [5]. Here we will focus on the first two kinds of learning as they are related to this research.

In supervised learning, one typically wishes to learn models from labeled data; the learnt models, if successful, will make predictions on future data. On the other hand, unsupervised learning, which does not require any label, typically requires us to make a useful description of the data by dividing the data into groups (clusters).

Before we explain the algorithms, we will first introduce the feature types of the instances that can be used as input. In general, there are four types of feature values at the measurement level.

- Nominal. The value of a nominal instance is one of a different collection of names; i.e., nominal values provide only information that can distinguish one instance from another. A class label is one example of a nominal feature.
- Ordinal. The value of an ordinal feature provides information to the order

of an instance. Document rank from a search engine is one example of an ordinal feature.

- Interval. For interval features, the difference between values are meaningful. Calendar dates are an example of an interval feature.
- Ratio. For ratio features, both differences and ratios are meaningful. Term frequency in a document is a form of ratio data.

More often, nominal and ordinal features are called qualitative features while interval and ratio features are called quantitative features. In this thesis, we will focus on quantitative features because they appear most often in our research and can be the input to the machine learning function below.

A machine learning problem can be characterised by the following components:

- The input space contains instances or objects under investigation: The objects in our context are usually documents. They can be represented by a vector, where each element of the vector indicates one feature of the object. The data in the input space cannot usually feed into the learning machine algorithm directly and preprocessing is required to achieve a desired performance.
- The output space contains the learning target with respect to the input objects. For example, given a web page, the output space can be either yes or no indicating whether or not it belongs to *sports* category. However, the output space is not limited to a single decision or a scalar number, rather it may have structure.
- The hypothesis space defines the function space for mapping the input to the output. In order to find the optimal hypothesis, training data is necessary to tune the model. One principle approach to evaluate tuning is to define a loss function (or equivalently a likelihood function[3]) based on the predicted output in comparison to the true output. Thereafter

[3]In Gaussian distribution, maximising the likelihood is the same as minimising the loss function

optimisation methods are applied to find the optimal results. However, empirical results show that if we only optimise a model for the training data, the tuned model can work perfectly for training data, yet make poor predictions for the test data; this is called *over-fitting*. To achieve a better generalization ability, regularisation [34] may be adopted into the loss function. The main idea of regularisation is to introduce the trade-off between the empirical loss and model complexity to the problem. There are at least three different ways of regularization: 1) Explicitly via constraints on model complexity. 2) Implicitly through incremental building of the model. 3) Implicitly through the choice of robust loss functions. Another approach is called the Bayesian model, in this case, loss minimisation corresponds to the likelihood function and the regularisation can be regarded as a prior on the model. One advantage of the Bayesian model is that it can support nonparametric inference [52], thus the model is less restricted than a parametric model.

2.2.1 Unsupervised Learning

Unsupervised learning is also called clustering or exploratory data analysis. In clustering there is no labeled data available for training. The goal of clustering is to make use of data similarity to separate a finite, unlabeled data set into a finite, discrete set of hidden groups or clusters.

Given a data set, the first problem we should address is how to measure the distance between a pair of instances.

We will now present some proximity measures commonly used in clustering. Perhaps the most commonly used distance metric is the *Euclidean distance*, defined as

$$D(\mathbf{x}_i, \mathbf{x}_j) = \left(\sum_{l=1}^{d} |x_{il} - x_{jl}|^2 \right)^{1/2}. \tag{2.21}$$

Euclidean distance can be generalized to a family of metrics, called the L_p norm, defined as,

$$D(\mathbf{x}_i, \mathbf{x}_j) = \left(\sum_{l=1}^{d} |x_{il} - x_{jl}|^p \right)^{1/p}, \tag{2.22}$$

when $p = 2$, it is the Euclidean distance.

The cosine similarity (Equation 2.2) is a useful measure in information retrieval.

Clustering algorithms can be categorised as flat clustering (as in K-means) or hierarchical clustering (as in agglomerative hierarchical clustering).

The K-means algorithm [53] is one of the best known and most popular clustering algorithms. It seeks an optimal partition of the data into K clusters by, for example, minimising the sum of the squared distance between each data point to the mean of the corresponding data points in its cluster. It involves an iterative optimisation procedure to seek an optimal solution, which often leads to a local minimum rather than a global one.

Agglomerative clustering belongs to the category of hierarchical clustering. It starts with n clusters, each of which includes exactly one data point, then a series of merge operations is carried out until all clusters are merged into the same cluster. The merge operation occurs between two clusters with the highest intercluster similarity. Finally it collapses down to as many clusters as requested. This kind of clustering is widely used in document clustering and other IR applications [121].

Although clustering is different from classification, it is possible to incorporate clustering into classification to enhance its performance, e.g. K-means classifier [63].

Unlike classification, to evaluate cluster validity is problematic due to the absence of ground truth knowledge. In this case, it is possible to use the objective function to evaluate the clustering quality, i.e., the objective function is the one that should be minimised by the clustering algorithm.

2.2.2 Supervised Learning

Supervised learning refers to the machine learning task of inferring a function from supervised training data. Based on the type of output space, it can generally be divided into two problems. If the output is a real value, this kind of learning model is essentially a *regression* problem. If the output is one of a finite set of discrete values, it is called *classification.* We will focus on classification problems.

The traditional problem setting for classification is as follows: consider independent and identically distributed data pairs (X_1, Y_1),...,(X_n, Y_n) we have:

$$X_i = (X_{i1}, ..., X_{id}) \in \mathcal{X} \subset \mathbb{R}^d,$$

where $\mathbb{R}^d$ is a d-dimensional real vector and Y_i is a discrete value in some finite set $\mathcal{Y}$. The task of classification is to discover a function $h : \mathcal{X} \to \mathcal{Y}$, which will make as few errors as possible on further data. When we observe a new input X, we predict Y to be $h(X)$.

In classification, it is typically assumed that (X, Y) is a pair of random variables with some joint probability density $P(X, Y)$. The classification problem can then be formally characterised as finding a function h which minimises the error rate defined as

$$L(h) = \mathbb{P}(\{h(X) \neq Y\}).$$

In practice, we use empirical error rate, that is

$$\hat{L}(h) = \frac{1}{n} \sum_{i=1}^{n} I(h(X) \neq Y),$$

where $I(\cdot)$ is the indicator function.

Amongst all classification models, k-nearest neighbour (KNN) is one of the simplest and effective methods. KNN is an instance based learning method. It makes use of the observations in the training set closest to the new input x to predict $\hat{Y}$. Specifically, the KNN model is defined as follows:

$$\hat{Y} = \frac{1}{k} \sum_{x_i \in N_k(x)} y_i, \tag{2.23}$$

where $N_k(x)$ contains the k closest neighbours to the new input instance x in the training set. There is only one parameter in this method, the number of neighbours k; this parameter is usually estimated by cross-validation.

Computing the closeness, or equivalently the distance between two instances can be problematic. For multivariate data points, the standard Euclidean distance (Equation 2.21) can be used. However, it assumes that the features are normalised and are of equal importance. One of the major problems is how to adjust the weights to reflect the different significance of the features. In the information retrieval area, the cosine distance is widely used to measure the distance between two documents.

k-nearest neighbour does not build any model in advance; it only makes a prediction when it receives a new instance. To find the k nearest neighbours for a new instance, it is necessary to calculate the distances from the new instance to every instance in the training set and select the closest. This procedure is linear in the number of training instances. To make the search more efficient, KD-trees [54] and Ball-trees [84] were proposed that make use of spatial data structures to avoid searching all instances. Another approach to alleviate the computation burden is to do approximate KNN [85].

KNN estimates the label of the new input based on the labels of its k nearest neighbours. The decision boundary is arbitrary based on its local features. We can also force the decision boundary to be smooth, and this leads to a global discriminative model.

Amongst these discriminative models, the linear model is widely used because of its simplicity and interpretability. Such models include least squared linear regression, linear discriminant analysis, logistic regression, perceptrons and Support Vector Machines. A good survey of these methods can be found in [63].

Here we will focus on Support Vector Machines.

If these data points are linearly separable, we can define the decision function:

$$h(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b, \tag{2.24}$$

where $\mathbf{w}$ is a d-dimensional vector, b is a bias term, and for $i = 1, ...n$

$$\mathbf{w}^T\mathbf{x}_i + b = \begin{cases} > 0 & \text{for } y_i = 1, \\ < 0 & \text{for } y_i = \text{-1}. \end{cases} \tag{2.25}$$

Any such discriminant function defines a linear decision boundary in the feature space that bisects the two training classes, and is characterised by $h(\mathbf{x}) = 0$. However, Figure 2.3 shows that there are many such hyperplanes separating the two classes. To select the hyperplane best suited to the task, SVMs maximise the distance (margin) between the decision hyperplanes and the training points closest to it. In Figure 2.3, the filled points are the closest data points to the

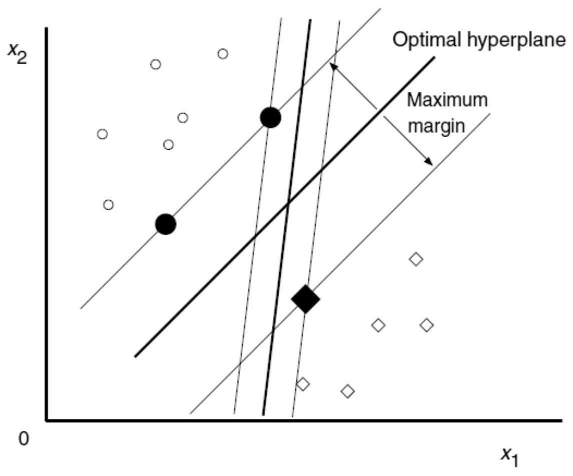


Figure 2.3: Optimal separating hyperplane in a two-dimensional space

decision hyperplane and the optimal hyperplane maximises the margin; these points are called support vectors. It is easy to show that maximising the distance is equivalent to solving:

$$\begin{aligned} &\min_{\mathbf{w},b} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} \\ &\text{subject to} \quad y_i(\mathbf{w}^T\mathbf{x}_i) + b \geq 1 \quad \forall i = 1, \ldots, n \end{aligned} \tag{2.26}$$

In practice, the classes in the training data are not always separable. To handle the general case, we must relax the inequalities in Equation 2.26 using slack variables and modify the object function to penalise any failure to meet the original inequalities. Then Equation 2.26 becomes [41]:

$$\begin{aligned} &\min_{\mathbf{w},b} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_i \xi_i \\ &\text{subject to} \quad y_i(\mathbf{w}^T\mathbf{x}_i) + b \geq 1 - \xi_i \quad \forall i = 1, \ldots, n \\ &\text{and} \quad \xi_i \geq 0 \quad \forall i = 1, \ldots, n. \end{aligned} \tag{2.27}$$

The constant parameter C controls the trade-off between the margin maximisation and the misclassification error minimisation.

The Lagrange (primal) function is then

$$L_p = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_i \xi_i - \sum_i \alpha_i[y_i(\mathbf{w}^t\mathbf{x}_i) + b - (1-\xi_i)] - \sum_i \mu_i\xi_i, \tag{2.28}$$

which we minimise with respect to $\mathbf{w}$, b and ξ_i. Setting the respective derivatives to zero, we get

$$w = \sum_i \alpha_i y_i \mathbf{x}_i, \tag{2.29}$$

$$\sum_i \alpha_i y_i = 0, \quad and \tag{2.30}$$

$$\alpha_i + \mu_i = C. \tag{2.31}$$

Thus substituting Equation 2.29 to Equation 2.31 to Equation 2.27, we obtain the following dual problem. Maximise

$$L(\alpha) = \sum_i^n \alpha_i - \frac{1}{2}\sum_{i,j=1}^n \alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j, \tag{2.32}$$

subject to the constraints

$$\sum_{i=1}^n y_i\alpha_i = 0, \quad C \geq \alpha_i \geq 0 \quad for \quad i = 1,\ldots,n. \tag{2.33}$$

The decision function is then given by

$$h(x) = \sum_{i\in S} \alpha_i y_i \mathbf{x}_i^T\mathbf{x} + b, \tag{2.34}$$

where S is the set of support vectors. This maximum margin problem in essence is a quadratic optimisation problem, and for large scale data sets it is solved using a decomposition strategy followed by some additional optimisation methods [96].

When the training data is not linearly separable, the obtained classifier may not yield high generalisation ability even if we introduce slack variables. To handle this problem, the original input space is usually mapped to a high-dimensional *feature space*. Note that Equation 2.34 requires the inner product of the two original instances. This implies that the inner product of the feature spaces are required. We can define the feature function $g(\mathbf{x}) = (g_1(\mathbf{x}),\ldots,g_l(\mathbf{x}))^T$ that maps the p-dimensional input vector $\mathbf{x}$ into an l-dimensional feature space. Thus, the linear decision function in the features space is:

$$h(x) = \sum_{i\in S} \alpha_i y_i g(\mathbf{x}_i)^T g(\mathbf{x}) + b. \tag{2.35}$$

For text categorisation, the data is in a rich dimensional feature space, and in this case, the performance is similar with or without a mapping. We refer to the case without using a mapping, a *linear SVM.*

In the Vector Space Model, a uni-gram model considers each term independently, which will cause information loss. To partially solve this problem, we can combine every two consecutive terms into an additional new feature, which leads to a bi-gram model. In this way, we can achieve better results by incorporating additional features. We call this model *bi-gram SVM.*

To solve the Support Vector Machine equation in the dual case, we have to estimate the Lagrange parameter α, and several methods such as the sequential minimal optimisation technique (SMO) [96], the cutting plane algorithm [74] and the dual coordinate descent method [67] have been proposed to solve this problem.

SVM can also be extended to solve other problems, such as ranking documents, using a method called ranking SVM [73], or structured prediction [115].

Other well known classifiers include Naive bayes, Rocchio and Decision Trees. Empirical experiments show that Support Vector Machines achieve competitive performance compared to other classifiers [72]. A comprehensive survey of classifiers for text classification can be found in [109].

To evaluate the performance of classifiers, we can use precision, and recall or F1 as mentioned in Section 2.1.2. For multi-class classification problems, one approach is to decompose the problem into several binary class problems and then compute the performance on these sub-problems. There are two methods to measure the results: Macroaveraging and Microaveraging. Macroaveraging calculates the simple average over classes, whereas Microaveraging pools per-document decisions across classes, and then computes an effectiveness measure on the pooled contingency table. Please refer to Section 2.1.2 for details.

2.3 Web Content Clustering and Classification

Clustering and classification have been widely applied to web content. We will discuss web content clustering in Section 2.3.1, followed by web content classification in Section 2.3.2 and end with a review of previous research of search results classification in Section 2.3.3.

2.3.1 Web Content Clustering

Clustering is an approach to organise web information that has been researched in the information retrieval community. The cluster hypothesis [117] states that *closely associated documents tend to be relevant to the same request*, which suggests that if one document from a cluster is relevant to a information request, other documents in the cluster are likely to be more relevant as well. This hypothesis has been justified in several experiments in the information retrieval community.

There are several applications of clustering in IR:

1. Search results clustering. In [66], the authors presented a cluster-based document browsing method, called Scatter/Gather, and validated the cluster hypothesis that relevant documents tend to form clusters. The Grouper system, described in [125], is one of the first systems for which an empirical comparison was presented between web search behaviour using a standard ranked-list approach versus a clustered approach. Their results demonstrated the substantial difference in the patterns of use of clustered presentation and standard ranked-list presentation. Clustering methods typically extract key phrases from the search results for grouping purposes, and attach to each group a candidate cluster label [92, 126]. The search results are treated as a bag of words/phrases, which are ranked according to the statistical features that have been found in them. Two cluster based search engines available on the Web, are Clusty (www.clusty.com) and Carrot-Search (search.carrotsearch.com/carrot2-webapp/search). A survey of clustering search engines results can be found in [29].

2. Collection clustering. In this approach the whole collection is clustered first, and then presents the results to users hierarchically. It is effective for exploratory browsing. Google news is one such example.

3. Cluster-based retrieval. It is similar to collection clustering, however, it aims at improving retrieval rather than browsing. Given a query, it first finds the closest cluster and then performs document search. This method can improve retrieval performance in certain cases.

4. Language modeling. Clustering can provide a benefit to language modeling. In [87], the authors incorporated the clusters into the smoothing function and it was shown to outperform traditional document based retrieval in terms of precision and recall.

2.3.2 Web Content Classification

Classification has been widely applied in information retrieval. As web searchers are subject to information overload, it is necessary, in our opinion, to organise information according to one or more ontologies, so that the information is presented in a more controllable way for users to understand and browse. However, it is infeasible for us to manually assign labels to all pages on the Web. Therefore, automatic classification and clustering techniques are required to handle this problem.

The general problem of web page classification can be divided into more specific problems: subject classification, functional classification, sentiment classification and other types of classification [98]. For example, assigning a relevant topic, i.e., *sports* or *entertainment* to a document is considered to be subject classification. Functional classification deals with the role of web pages. For example, deciding whether a page is a "personal homepage" or a "course page" is an instance of functional classification. In query classification, which is analogous to query type classification, each query is classified as an *informational*, *navigational* or *transactional* query. Sentiment classification focus on the opinion presented in a Web page [95].

In addition to the conventional text classification issues, such as high dimensionality, sparsity, Web page classification has some special characteristics:

- Web pages are semi-structured documents mostly written in HTML, where the information is enclosed between tags. We can remove the tags to convert a semi-structured document to an unstructured representation, but those tags embed useful information for the user. For example, the text between <title> tags usually has more weight attached to it than the text between <p> tags.
- Web pages also contain topological information about the link graph. The underlying assumption is that hyperlinks contain information about human judgements pertaining to the linked document. This feature is not present in typical text classification problems.

We will describe a number of applications of classification in web information retrieval.

- Classification of search results. Refer to Section 2.3.3.
- Ranking functions in IR. The probabilistic model assumes that information retrieval is a classification problem, and in the simple case the judgement of a document's relevance is binary. For example, BM25 makes use of the query term frequency in one document to estimate the relevance score of the document [102]. More recently learning to rank [86] has become popular in information retrieval. One strategy is to estimate pairwise preferences from training data. Ranking SVM is one method that builds a classifier to predict the preference for retrieved results, which is then used for a final ranking.
- Focused crawling. In many cases a search engine is required to be focused on particular topics that are of interest to specific groups of users; such a search engine is called a *vertical search engine*. Chakrabarti et al. proposed focused crawling as an approach with which the crawler only collects the documents of interest as measured by a classifier [30].
- Query classification. The user typically issues short queries when seeking information. To find the exact information need is one of the core chal-

lenges for search engines, and query classification may be useful in this sense. Query classification can also be used for ad placement, and query routing [114]. The task of the KDD-Cup 2005 competition was to classify 800,000 web user's search queries into 67 predefined categories in conjunction with 111 manually labeled queries which act as examples [81]. Shen et al [110] built an SVM classifier with the aid of the Open Directory, and made use of relevance feedback to enrich the query terms providing additional test data, to supplement the three categories involved in search engines to achieve the best results in the competition [110].

- Classification is also useful for query answering systems [123], and web content filtering [33].

Classification and clustering are methods that organise documents according to certain criteria. However, there are significant differences between them.

- Classification requires training data while clustering does not.
- Classification is based on fixed taxonomy, while clustering is not. Clustering is more flexible in this sense, but the cluster may not have a clear interpretation as given by the generated cluster labels.

2.3.3 Search Results Classification

Many main commercial search engines present search results in a ranked list. The ranks of the documents are determined by the relevance to the corresponding query. This relevance measure is described in Section 2.1.1. Unfortunately, those models cannot fully understand the user's intention. Meanwhile, most queries are short, making the query ambiguous or vague. This results in a low precision in the retrieved results and forces the user to sift through the list to find the relevant documents, in particular for informational queries. It will be ideal if current search engines can separate the irrelevant documents from the relevant documents. However, the ranked list interface of search engines cannot solve this issue even ranking algorithms become perfect since users could have different information needs when they issue the same query. Moreover, where users may be satisfied with one relevant result, i.e., for navigational queries or queries in a question answering system, result diversification can be adopted, which will benefit the user population overall within the ranked list interface. Result diversification will be discussed later.

Search Result Classification has been proposed as a solution to this problem. We will review the previous research on this topic.

Chen and Dumais [31] developed a system called SWISH in which the search results were automatically classified into shallow hierarchical categories, where only the top-two-level categories of a Web taxonomy ware used. They collected web pages from LookSmart's Web directory, which consists of 13 top-level categories and 150 second-level categories. A Support Vector Machines (SVM) algorithm was used as the classifier [47]. The reported accuracy of SVM in their work was about 70%. Then they conducted a user study to compare the category-based interface (referred to as "Category Interface" henceforth) with the conventional search interface where pages are arranged in a ranked list (referred to as "List Interface"). They deliberately designed the user interface to assist the search, i.e., presenting additional category information and summaries of the web pages as hover text on the limited screen estate, using heuristics to selectively present a small portion of the most useful information on the screen for each category. The results convincingly demonstrated that the category interface was superior to the list interface in both subjective and objective measures.

Dumais et al. further evaluated seven interfaces for integrating semantic category information with web search results. A automatic classifier was used to categorize the search results. The seven interfaces include: 1) List interface with hover summary. 2) List interface with summary inline. 3) List interface with category names. 4) Category interface with hover summary. 5) Category interface with summary inline. 6) Category interface with no category names. 7) Category interface with no page titles. A user study was conducted and the results confirmed that Category interfaces were faster than List interfaces. Even the List presentation with Category Names interface contains the same information as the Category interface, the performance was much slower with the list. Also Inline summaries were more effective than summaries presented as Hover text in spite of the fact that more scrolling was required. Offer Drori [45] conducted a similar user study on list interfaces. The search results were mostly manually assigned. He found that even in a list interface, displaying the categories provided some benefit.

CatS [100] is a meta-search engine that utilized text classification techniques to improve the presentation of search results. The taxonomy was based on the top 2 levels of the Open Directory Project, while many categories of the second level were merged or discarded. A NaiveBayes classifier was trained using web-page titles and descriptions from the data of the Open Directory Project. Some key aspects of the system (including HTML parsing, classification and displaying of results) were described and five text classification algorithms were compared in the experiments. However, the authors did not provide any utility analysis of the system and no experiment confirmed the benefits of the system.

The above research focused on shallow hierarchical categories. Dikan Xing et al. thought that it may be too coarse for users to browse. To solve this issue, a deep classifier [122] was proposed. A user query was first issued to a web directory, for example to the Open Directory, and the corresponding categories of the results were used to prune the original hierarchy into a decent size while reserving the hierarchical structure and depth. Then sufficient training data were collected to build reliable classifiers (Naive Bayes Classifier) for classifying the retrieved search results. Therefore, this system can provide more informative class labels for users' browsing. Although their approach can improve the system in terms of the accuracy and the speed, the authors did not demonstrate the advantage of the deep classifier in terms of users' search behavior.

Search results classification is related to another two research directions.

- Query performance prediction.
- Search results diversification.

Query performance prediction [64, 43, 128] is an interesting and important topic in Information Retrieval. Amati et al. [6] showed that the use of query performance predictors allowed to devise a selective decision methodology to avoid the failure of query expansion. The basic idea is that a query whose highly ranked documents are about a single topic has a model characterized by unusually large probabilities for a small number of topical terms, so such queries will have a high score. If a query returns a mix of articles about different topics, the articles have a model that is smoother and more like the model of the collection as a whole, therefore those low-coherence queries would get a low score. Cronen-Townsend et al. define the measure related to the lack of ambiguity called clarity score [43]. The score for the query is typically measured by Kullback-Leibler divergence [18] between the query and collection language models [43]. In the absence of topics for retrieved results, those work resort to the approach to measure the divergence between the terms distribution from a query language model or highly ranked documents and the collection models. In our work, the topic information of each retrieved document is available through the classification. Then we can estimate the clarity score based on the distribution of the topics, i.e., if most of the retrieved results are in the same class, such a query will have a high clarity score. So we do not require the information of the collection to measure the clarity score, but one training data set is necessary for the classifier.

As user queries are short and sometimes ambiguous, the users will have various desired information needs even when they issued the same query. Conventional search engines will return one set of retrieved results depending on the users' queries, not the user. Result diversification had been proposed for this problem [57] when users may be satisfied with one relevant result, i.e., navigational queries. Ideally, the document ordering for the query should properly account for the interests of the overall user population [37]. Result diversification is usually carried out by similarity functions or conditional relevance distributions defined over documents [127, 32] or by user feedback [99]. Zhai [127] formalized this problem within a risk minimization framework and proposed two loss functions that lead to a selection of diverse results. Chen et al [32] proposed the idea that documents should be selected sequentially according to the probability of the document being relevant conditioned on the documents that came before. More recently, Davood Rafiel et al [101] model the problem as a constrained expectation maximization and estimate the correlation between pages based on the past search data and textual contents. Rakesh Agrawal et al [3] study this problem in a similar approach to our work. They selected documents sequentially based on marginal utility, which is a measure of the probability that the selected document satisfies the user given that all documents that come before it fail to do so. More specifically, they select the diverse documents based on a probability distribution of categories for the query (which is a query classification question in our work) and the quality value of the documents, which is the probability that a document satisfies a user who issued the query with the intended category. Although previous research is related to our work when considering the relationship of retrieved documents, there are explicit distinctions. Result diversification focuses on the reranking of the search results in a list interface. Our work makes use of a category-based interface. Result diversification will consider the trade-off between relevance and novelty of the documents given the limited user interface; Our approach presents the retrieved documents in term of categories, so the space limitation is not an issue. Moreover, our approach can also benefit informational queries and poor navigational queries.

In addition to the search results classification, there is another issue - ranking categories. Demartini et al. [55] investigated this problem recently. They conducted a user study on a small scale and found that the average similarity between the user query and the documents within each category yields the best category ranking.

Another way to organize the search results is by clustering. Like classification, it aims at partitioning a dataset into subsets that bear similar characteristics. However, it is different for classification. Clustering is unsupervised learning, which does not require training data, i.e., it does not assume any prior knowledge. Clustering does not assume a fix taxonomy, although it seems to be flexible, in practice, it can cause some issues. For example, in the context of search result categorization, clustering should provide a reasonable label for each subset to help users understand the topic of such subset. However, such labels are generated on the fly and users may not understand its underlying meaning [31]. To generate a meaningful label is still an on-going research problem in information retrieval. Classification does not suffer from this issue. This is because it has an explicit fixed class structure and class labels. Moreover, such labels are well defined. The user can gain a better understanding of such labels through practice even if they are not familiar with the labels.

Chapter 3

A Model for Classification-Based Search Engines

In this section, we will discuss the problem of traditional search engines and present a model for classification-based search. In addition, metrics are defined to measure the performance of the system.

3.1 Problem Description

Search engines play a crucial role in information retrieval on the web. Given a query, search engines, such as Google, Yahoo! and Bing, return a ranked list of results, referred to as the result set. For many queries, the result set includes documents on a variety of topics, rather than a single topic. This variation is often due to the fact that a typical query contains about only two to three terms, which is insufficient to locate the desired information unambiguously. For example, the query "Jaguar" will often return documents referring to both the car and the animal. While the user is only interested in one topic, it is not possible for a search engine to know which topic is relevant based on the query alone. Moreover, the standard ranking of the documents in the result set is independent of the topic. Thus, the rank-ordered result set has an arbitrary topic ordering. Referring to the "Jaguar" example, this means that a user must scroll through a ranked list in which many documents may be irrelevant.

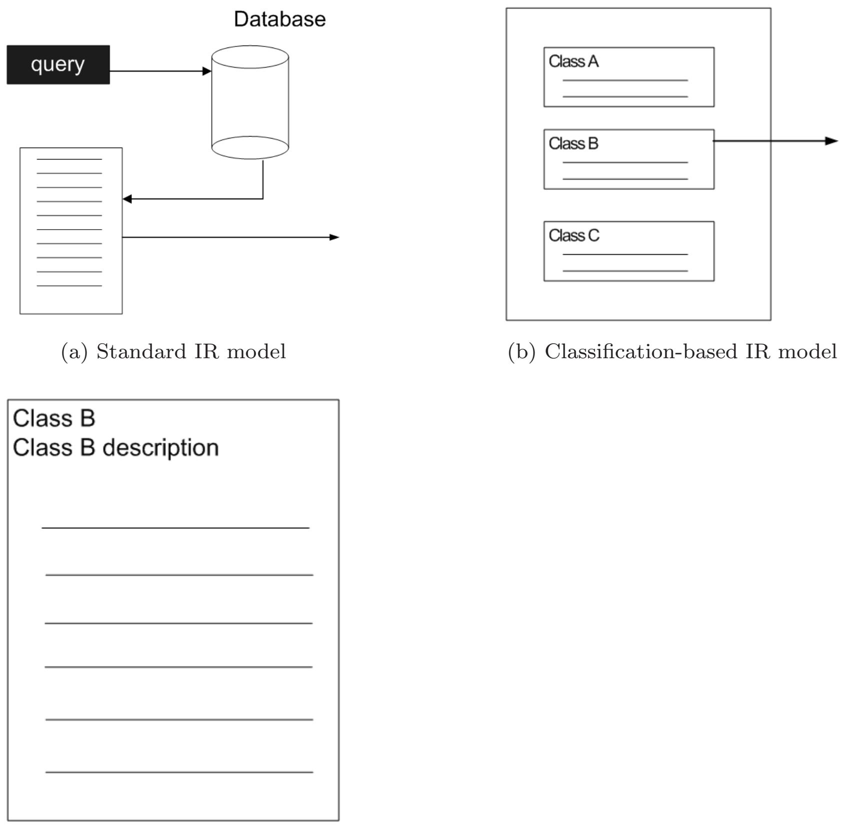
In Chapter 2 we reviewed how grouping strategy can assist the user by organising the documents in the result set into groups, all documents within a group referring to a common topic. Intuitively, for the case that users may be satisfied with one relevant result, we would expect grouping to substantially reduce search time [76], where search time is measured by the number of documents a user must examine before finding the desired document. For the query "Jaguar", a user might be shown two distinct groups of documents: one referring to the animal and the other referring to the car brand. A user can immediately ignore the non-relevant topic and focus his attention to only the relevant topic. For this simple example, this grouping may, on average, halve the number of documents the user must examine.

Although previous researchers have evaluated their prototype systems, there has been no attempt, to our knowledge, of formulating a generic user interaction model for a retrieval system, which allows the benefits of grouping to be quantified in comparison to a standard retrieval system which does not group its results.

3.2 Classification-Based Search Engines

We describe the architecture and ranking methods of a classification-based IR system that we have been developing. We assume the existence of a standard retrieval system that, given a query, returns a ranked list of documents as the result set. Web search engines such as Google, Bing and Yahoo! satisfy this assumption.

Given a ranked set of documents, it is necessary in our proposed system to classify the documents into their respective classes. Figure 3.1 provides a conceptual view of the classification-based information retrieval system we have developed. Figure 3.1a depicts the ranked set of documents provided by a



(a) Standard IR model (b) Classification-based IR model

(c) Results in class B

Figure 3.1: Conceptual illustration of a classification-based IR system

standard IR system. Our system classifies these documents into a number of classes, ranks the classes and then displays a ranked list of classes to the user, as depicted in Figure 3.1b. When a user clicks on a particular class, the ranked set of documents in this class is then displayed to the user, as shown in Figure 3.1c.

We now provide a more formal framework for our system. We assume that there are $|D|$ documents in the original result set, D, and we denote the standard IR rank of a document, $d_k \in D$ as $s(d_k)$; we refer to this rank as the *list rank* (LR). For convenience of presentation we assume that the documents are ordered such that document d_k has list rank $s(d_k) = k$. According to eye tracking experiments[1] and [75], the position in the list can be approximated by the time it takes to find a result. Thus, in the remaining work, rank will be used as a proxy for the search time.

3.2.1 Class Rank

After performing classification on the original result set returned by the standard IR system, the documents are grouped into $|C|$ top-level classes. Each class, c_i, consists of a set of documents, $d_{i,j}$, where $1 \leq j \leq |c_i|$, and $|c_i|$ denotes the number of documents in a class, c_i. Classification hierarchies will be briefly discussed in Section 3.4.

Given the set of classes, C, a rank ordering of the classes is necessary. For a document, $d_{i,j}$, let $\psi(i,j) = k$ denote the corresponding index of the same document in the initial result set as output by the standard IR system. For each document, $d_{i,j}$ in class i, we associate a score $t(d_{i,j}) = k$. Each class, C_i, is then assigned a score

$$\phi(c_i) = -min(t(d_{i,j})) \quad \text{for } 1 \leq j \leq |c_i|,$$

where $\phi(c_i)$ outputs the score for each class; and in order to maximize the score for each class, we take the negative of the min function. In our case, the score of each class, which determines the class rank, is based on the document with highest list rank within the class. For notational convenience, we assume the classes to be ordered such that class c_i has rank i.

The result is illustrated in Figure 3.2. On the left part of Figure 3.2 is the retrieved results set, we assume the asterisked document is the target document, in this case, it is document 5. The results are classified into 3 classes as shown in the middle part. The ranking is illustrated in the right panel and it is ordered by the class score, which is determined by the top ranked document within the class, i.e., d_1, d_3 and d_4 respectively.

We believe that this simple method of ranking classes is novel and, more importantly, minimises the affects of the ranking method on the performance

[1] http://www.useit.com/alertbox/reading_pattern.html

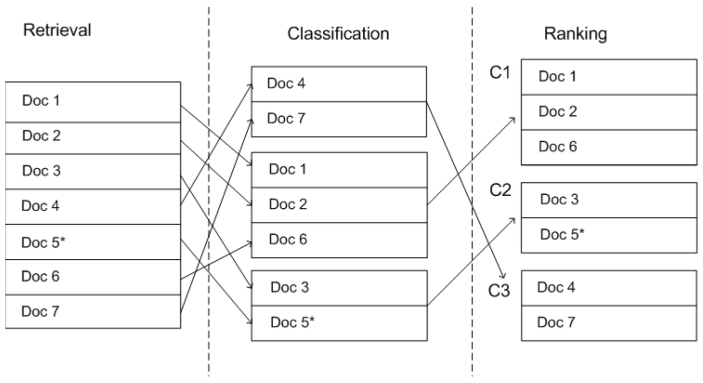


Figure 3.2: The Illustration of Ranks

of the classification-based system. Conversely, if we had developed a more sophisticated ranking system for classes and documents within classes (see Section 3.2.2), then it becomes increasingly difficult to determine whether differences in performance compared with a standard IR system are due to classification, or the new ranking algorithm. We will leave the discussion of other advanced class ranking methods to Chapter 5.

3.2.2 Document Rank

Having ranked each class, it is now necessary to rank the documents, $d_{i,j}$, within each class, c_i. To do so, we assume the existence of a function, $\varphi(d_{i,j})$, that outputs a score for each document. Here we adopt one of the most popular methods, the list ranks of the documents, $t(d_k)$, as output by the standard IR system, as a score for each document and rank the documents accordingly. Thus, the score for document, $d_{i,j}$, in class, c_i is given by $\varphi(d_{i,j}) = -t(d_{\psi(i,j)})$.

Documents within the class are then ranked according to their scores, the highest score being ranked first, as in traditional search engines. For notational convenience, we assume the documents to be ordered such that document $d_{i,j}$ has rank j in class c_i.

3.2.3 Scrolled-Class Rank (SCR)

As we shall see shortly, it is often useful to talk about the *scrolled class rank*, denoted by $s(d_{i,j})$, which we define as the total number of classes and documents a user must examine to find document $d_{i,j}$ by sequentially scrolling through each class and its associated documents in rank order.

In this case, the user will look at i classes, and all of the documents in the previous i-1 classes together with the first j documents of the last class. Thus, the scrolled-class rank of document, $d_{i,j}$ is given by

$$s(d_{i,j}) = i + \sum_{k=1}^{i-1} |c_k| + j. \tag{3.1}$$

Referring to Figure 3.2, the target document is $d_{2,2}$ and the scrolled-class rank is

$$s(d_{2,2}) = 2 + 3 + 2 = 7. \tag{3.2}$$

3.2.4 In-Class Rank (ICR)

When a user selects a class, it may or may not contain the target document. When the document is present in the class, we refer to this as the *in-class rank*. The in-class rank measures the number of class labels and documents that the user examines, when the target document, $d_k = d_{i,j}$ is in class, c_i. The in-class rank is

$$r(d_{i,j}) = i + j, \tag{3.3}$$

since the user must look at the first i-ranked class descriptions and then the first j-ranked documents within the known class.

Therefore, for target document in Figure 3.2, the in-class rank is

$$r(d_{2,2}) = 2 + 2 = 4. \tag{3.4}$$

However, in general, a classification-based IR system introduces a small overhead. If the target document is ranked high, then this overhead may be noticeable. For example, when the target document with a list rank of 1, i.e., it is the top ranked document, the in-class rank is 2 because we must first look at the class before finding the target document.

3.2.5 Out-Class/Scrolled-Class Rank (OSCR) and Out-Class/Revert Rank (ORR)

If the target document is not within the selected class, then the user must perform additional work. We refer to this situation as the “Out-Class”. Upon failing to acquire the target document in the chosen class, the user may choose to either:

(i) *scroll* through the classes and the documents in each class in rank order, or

(ii) *revert* to the standard IR display and sequentially scroll through the ranked result set.

The *out-class/scrolled-class rank* and *out-class/revert rank* are, respectively, the number of documents the user must then examine in order to find the target in case (i) and (ii) above. We now formalise these notions.

The out-class/scrolled-class rank, $p(d_{i,j})$, is the total number of class labels and documents that a user must examine in order to find the document for case (i), where the users chooses to scroll through the classes and documents in rank order, after not finding the target in the selected class. Let c_e denote the class the user erroneously selects. Then the out-class/scrolled-class rank is given by

$$p(d_{i,j}) = \begin{cases} (e + |c_e|) + s(d_{i,j}) & \text{if } e > i, \\ e + s(d_{i,j}) & \text{if } e < i. \end{cases} \tag{3.5}$$

In Figure 3.2, if the user first selects class 3, i.e., $e = 3$, the out-class/scrolled-class rank is

$$p(d_{2,2}) = 3 + 2 + 7 = 12. \tag{3.6}$$

If the user first selects class 1, then the out-class/scrolled-class rank is

$$p(d_{2,2}) = 1 + 7 = 8. \tag{3.7}$$

The out-class/revert rank, $q(d_{i,j})$, is the total number of class labels and documents that a user must examine in order to find the document for case (ii), where the user reverts to the standard result set, i.e. no classification is used in the second phase of the search. The out-class/revert rank is given by

$$q(d_{i,j}) = (e + |c_e|) + t(d_{\psi(i,j)}) = (e + |c_e|) + t(d_k) = (e + |c_e|) + k, \qquad (3.8)$$

where $\psi(i,j) = k$. Note that the out-class/revert rank is a hybrid search strategy that begins with a classification-based strategy and reverts to a ranked-list strategy if the document is not present in the first class selected. This hybrid strategy is different from the presentation in cluster-based search engines, where the user is presented with the ranked listing in a main window and the clusters in another.

If we assume the user selects the class 1 and then revert to standard IR display, the out-class/revert rank in our example is

$$q(d_{2,2}) = 1 + 3 + 5 = 9. \qquad (3.9)$$

3.2.6 Classification

We have, until now, ignored how classification is performed. In the experiments of Section 3.4 we assume two cases.

In the first case we assume we have an oracle based on 12 top level categories of the Open Directory that correctly classifies the documents. Of course, in practice, this is not possible. However, analysis of this case provides us with valuable information regarding the best-case performance of the system. Any other system in which classification errors occur will perform worse.

In the second case, we assume classification is performed based on a k-nearest neighbour (KNN) classifier [46]. That is, after removing the stop word and stemming, all training data and the corresponding class labels are indexed using Lucene[2]. Given a test document, d_j, one query is automatically generated based on term frequency. Then we issue it to training data and we retrieve its k most similar documents. The class label with majority vote is assigned to the test document as its class label. The similarity is calculated based on the

[2]http://lucene.apache.org

following default scoring formulae [1]:

$$\begin{aligned} score_d = \sum_{t \in q} (tf(t\ in\ d) \times idf(t)^2 \times boost(t.field\ in\ d) \times \\ lengthNorm(t.field\ in\ d)) \times coord(q,d) \times queryNorm(q) \end{aligned} \tag{3.10}$$

where $score_d$ is the score for document d. $\sum_{t \in q}$ represents sum for all terms t occurred in query, $tf(t\ in\ d)$ is Term frequency factor for the term (t) in the document (d). $idf(t)$ is invert document frequency of the term. $boost(t.field\ in\ d)$ is field and document boost. $lengthNorm(t.field\ in\ d)$ is the normalization value of a field, given the number of terms within the field. $coord(q,d)$ is Coordination factor, based on the number of query terms the document contains. $queryNorm(q)$ is normalization value for a query, given the sum of the squared weights of each of the query terms. For details we refer to [58] and online java document. Based on our experiments, we found that using 15 most frequent words as a query and selecting 15 nearest neighbours can obtain the best results. In the remaining section, we will use 15 words as a query and retrieve 15 nearest neighbours. Note here the scoring method is a variant of cosine similarity in vector space model as we mentioned in Chapter 2. Although in vector space model, the weights can be boolean value, i.e., 1 if the word occurred in the document and 0 otherwise, or term frequency value, tf-idf weighting scheme is adopted in the experiment and it is believed to produce search results of high quality. For the comparison of other weighting schemes, we refer to Gerard Salton's paper [106].

We believe that the KNN classifier may not be the best classifier and it can be improved further, but it is simple to be applied to our experiment. The performance of the classifier is an important factor in classification-based search systems; however, our target is to evaluate the performance of classification-based search and our evaluation methodology can be applied to any kind of grouping system, no matter how good the grouper is.

3.3 Experimental Methodology

In this section, we will describe the experimental methodology, in particular how we design the target testing experiment, how we generate the query and user/machine models for the respective ranks.

3.3.1 Target Testing

The experimental methodology *simulates* a user performing a known-item search, also referred to as target testing [42].

In our context we make use of target testing to evaluate the performance of a classification-based retrieval system. The motivation is that target testing allows us to evaluate the system automatically without users, and is a precursor to user testing. Additionally, target testing allows us to evaluate the system on numerous queries at a minimal cost in comparison to user testing. However, target testing has some drawbacks in that the queries generated for target testing do not necessarily simulate "real" user queries. Moreover, good performance of the system for target testing does not guarantee similar performance when testing the system with "real" users.

3.3.2 Automatic query generation

For a given document repository, in our case extracted from the Open Directory, we randomly select documents as targets. For each target document, a user query is automatically generated by selecting a number of words from the target document. This can be performed in a variety of ways (cf. [9, 118]). However, the exact procedure is not important. We only require that queries can be generated so that the target document appears in retrieved result set within a designated range of list rank. This allows us to simulate a range of good (high ranking) to poor (low ranking) queries.

3.3.3 User/Machine models

Table 3.1 summarises the models and corresponding user strategies we described in section 3.2. The three user models are (i) the user knows the class (cases 1 and 4); (ii) the user does not know the class (cases 2 and 5) and, (iii) the user thinks he knows the class (cases 3 and 6). Note that there are two cases associated with each user model, because there are two machine models (correct/incorrect classification of the target document). In Table 3.1, we assume that the user employs the search strategies we introduced in Section 3.2.

simulated user/target	correctly classified	misclassified
knows the class	Case 1 : ICR	Case4a : OSCR
		Case4b : ORR
does not know the class	Case2a : SCR	Case5a : SCR
	Case2b : LR	Case5b : LR
thinks knows the class	Case3a : OSCR	Case6a : OSCR
	Case3b : ORR	Case6b : ORR

Table 3.1: Summary of the operating conditions and the number of classes and documents examined in each case.

Notation	Meaning
LR	List rank
SCR	Scrolled-Class Rank
ICR	In-Class Rank
OSCR	Out-Class/Scrolled-Class Rank
ORR	Out-Class/Revert Rank

Table 3.2: Summary of the notations used in Table 3.1.

Table 3.2 summarizes the notations used in Table 3.1. Refer to Section 3.2 for details.

3.4 Experiments

The dataset used in our experiments is derived from the Open Directory Project (ODP), a comprehensive human edited directory of the Web, compiled by a global community of volunteer editors.

We have selected the following 12 top-level classes to construct our testset: Arts, Business, Computers, Games, Health, Kids and Teens, Society, Science, Shopping, Home, Sports and Recreation.

We downloaded all the documents from these 12 top-level classes during September 2006. After removing noisy and trivial data[3], we divided the remaining 792,030 documents into a training set (500,430 documents) and a test set (291,600 documents). The training set was used for classification using a k-nearest neighbour classifier.

We randomly selected 600 target documents from the test set, and for each target document we generated 10 queries. The queries were designed so that the list rank of the target document retrieved by the standard IR system fell into one of 10 intervals of rank, namely 1-5, 6-10, 11-15, 16-20, 21-25, 26-30, 31-35, 36-40, 41-45, and 46-50. Thus, for each target document, we used a set of 10 queries that ranged from "very good" (list rank between 1-5) to "very poor" (list rank 36-50). The experimental results were averaged over all 600 target documents.

The underlying IR system is based on the open-source search software, Lucene. For stemming we make use of the open-source stemmer, Snowball [4]. The default document ranking algorithm from Lucene was used.

3.4.1 Experimental results

We performed three groups of experiments. In the first two, we restricted searches to documents contained only in the Open Directory. In our first experiment, we are therefore able to use the Open Directory as an oracle to classify the

[3]Any document of length less than 50 bytes was removed. Any document that consisted of images, pdf, excel and word was also removed.

[4]http://snowball.tartarus.org

result set. The second experiment used a k-nearest neighbour classifier trained on a subset of the Open Directory to classify the result set. In the final set of experiments, documents were retrieved from Google, and classification was performed using the k-nearest neighbour classifier.

3.4.1.1 Classification Based on Random Classifier

A random classifier serves as a base line for our system. Given 12 classes, a random classifier will predict each document with equal probability of being one of the 12 classes. Therefore, the prediction is data independent. Hence, the distribution of the number of class selected and the distribution of class size follows a certain distribution for a random classifier. More specifically,

- For each query, the number of classes selected for retrieved result set is proportional to the number of ways to select k classes and the number of ways to assign the retrieved result set into the k classes.
- For each class, the number of documents in that class is also a binomial distribution [17].

The number of classes selected is as follows:

$$\binom{12}{k} \sum_{k_1,\ldots k_k} \binom{n}{k_1, \ldots, k_k}, \quad \textbf{subject to } k_1 + \ldots + k_k = k \textit{ and } k_i > 0$$

Where $\binom{12}{k}$ is the number of ways to select k classes. The sum of multinomial coefficient measures the number of ways to assign the retrieved result set into k classes. This is due to the fact that to assign n documents into k classes is equivalent to assign n different ball into k different boxes and it is a classical combinational problem, the standard solution is multinomial coefficient.

For each class, the probability of one document being classified into it is $\frac{1}{12}$, and the probability of the document not being in the class is $\frac{11}{12}$. For one document, this is one Bernoulli trial. If we carry on a sequence of independent Bernoulli trials, which means we try to predict a sequence of assignments of documents into classes, we obtain a binomial distribution [17],

$$\binom{n}{r}(\frac{1}{12})^r(\frac{11}{12})^{n-r},$$

where n is the number of retrieved results, r is the random variable that measures the number of results in one class.

We simulate the case that each result is randomly classified into one of the 12 classes, and the target document will be correctly classified into the correct class with a probability of $\frac{11}{12}$. Figure 3.3a shows the performance of a random classifier. The probability of the target document being one class is uniform across all ranks. So the cumulative distribution increases linearly from 0 to 1. Moreover, the random classifier perform worse than all other classifiers except Scrolled-Class Rank (worst case) when initial list rank is greater than 27.

3.4.1.2 Classification Based on an Oracle

Each of the 600 documents has been manually classified into one of the 12 classes. Thus, Open Directory provides us with an oracle with which to classify all documents in the original result set. This allows us to first examine the best-case performance of our classification-based IR system, i.e. when there are no machine classification errors and the simulated user knows the correct class (case C1 in Table 3.1).

We can also introduce and control error rates for both the user and the machine classifier. Note that, from Table 3.1, user errors and machine classification errors both result in the same search length (cases C3, C4, and C6). Moreover, the two cases where the user is aware that they do not know the class (cases C2 and C5) are unaffected by the machine misclassification. Thus, when we report error rates, we do not distinguish between human and machine error rates. Rather, the error rate represents the combination of the two.

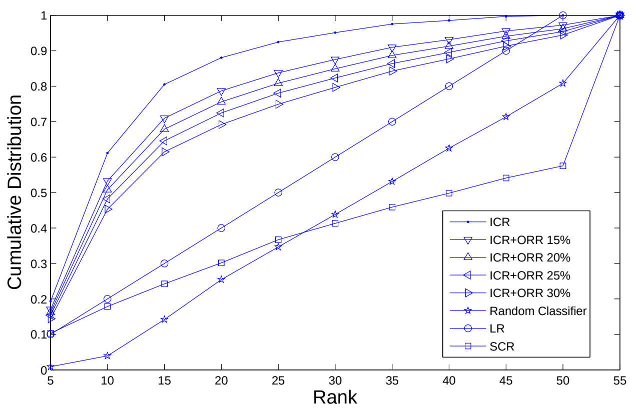
Figure 3.3a summarises the results for these cases. It shows the *cumulative* probability of finding the target document as a function of the rank of the target document. We note that for the standard IR system, the rank corresponds to the list rank (LR). For the error-free case, the rank corresponds to the in-class rank (ICR). For non-zero error rates, the rank corresponds to the ranks summarised in Table 3.1.

For the standard IR system, the list rank (LR) is a straight line, since the list rank is evenly distributed within the 10 intervals described above. We see that for the error-free case (ICR), the classification-based IR system performs significantly better than the standard IR system (LR). In particular, we observe that approximately 60% of all target documents have a rank of 10 or less. That is, for an ideal user and no machine misclassification (case C1), the user must look at no more than 10 classes and documents in order to locate the target document.

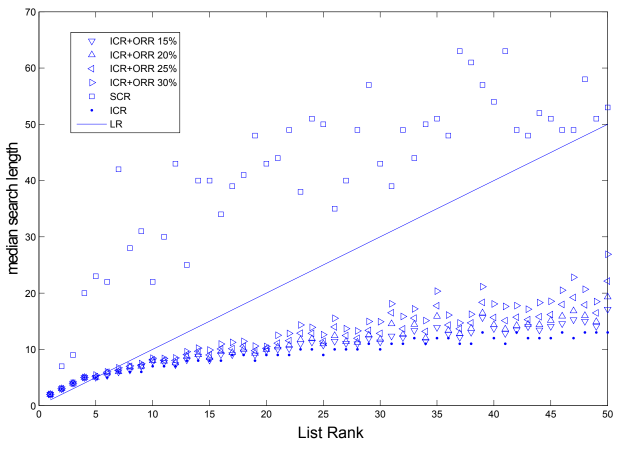
The oracle also allows us to control the misclassification rate. The rates we describe can best be thought of as the combined user and machine error rates. We introduced an error rate of $x\%$ as follows: for $x\%$ of the 600 queries, the user randomly selects a class that does *not* contain the target document, and then uses the out-class/revert (ORR) ranking strategy to locate the document. For the remaining $(100 - x)\%$ of queries, the user chooses the correct class and the target document is found using the in-class ranking (ICR) strategy. Thus, the curves for non-zero error rates represent a combination of two strategies, ICR and ORR.

For an overall error rate of 15%, we observe a decline in performance, as expected. However, at this error rate, the classification-based IR system still performs significantly better than the standard system. For example, over 50% of all target documents have a rank of 10 or less. As the overall error rate increases, the performance degrades. However, this degradation is rather smooth and even with an error rate of 30%, the performance remains significantly better than that of the standard IR system.

Finally, for completeness, Figure 3.3a also shows the cumulative distribution for the scrolled-class rank (SCR). This curve is significantly worse than the list rank of the standard IR system. Figure 3.3a shows that in cases C2 and C5, when the user does not know the class, he is best advised to abandon the classification-based IR system and immediately return to the standard system, i.e. follow the second strategy of list-rank (LR) in Table 3.1. Moreover, in cases C3, C4 and C6, where we have either a user or machine error, if the user does not find the target in the class he knows or thinks is the correct class, the advice is the same, i.e., revert to the standard system following the second strategy of



(a) cumulative distribution



(b) Median search length of query results

Figure 3.3: Results using the Open Directory oracle classifier

out-class/revert (ORR) in Table 3.1. That is, a hybrid-based search strategy performs better than either a category-based or ranked-listing alone.

It is important to recognise that the cumulative distribution does not present the full story. Figure 3.3b plots the median search length as a function of the list rank. Note that here we use the median search length distribution since the search length is highly biased by a small number of outliers, while the median is more robust to this bias. It is clear that for target documents with a low list rank (less than 5), the median search length using a classification-based system is slightly longer, on average, due to the overhead of inspecting the class description or when an error occurs. Thus, for very good queries, a classification based system actually increases the search length slightly. Conversely, for poorer queries, the median rank of the target document in the classification-based system is always shorter, on average. Interestingly, both the standard IR system and the classification-based system have regions of superior performance. Only when the initial query is poorer, i.e. the list rank is below a certain threshold, does the classification-based system offer superior performance.

Figure 3.3b also shows, as expected, that this threshold increases as the misclassification rate increases and it is more evident for poor queries. Thus, for example, for a misclassification rate of 25%, the list rank must be greater than 7 before a classification-based system is superior.

It is also worth noting that in Figure 3.3b the quality of the initial queries is uniformly distributed, by design. Thus, 10% of queries have an initial list rank between 1-5, another 10% between 6-10, and so on. In practice, the distribution of queries is a function of (i) the user, (ii) the distribution of documents in the database, and (iii) the document scoring function [118]. Thus the benefits of a classification-based system will depend strongly on the distribution of the queries. It is interesting to note that a number of studies, such as [13, 113], that have reported poor correlations between user judgments of document rankings and those produced by search engines, suggesting that classification-based systems may be useful in practice.

3.4.1.3 k-Nearest Neighbour Classification

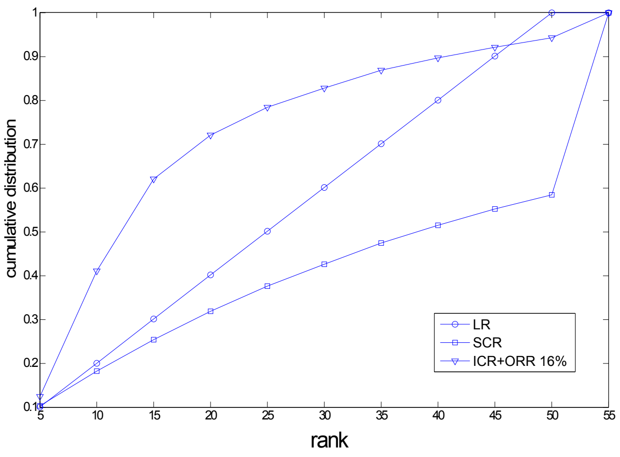
The experiments of the previous section show that very good performance can be expected from a classification-based system, especially for poorer queries. The definition of a "poorer query", is a query for which the list rank is larger than a given threshold, varies with the misclassification rate. Simulated misclassification rates of 15-30% suggest that (i) performance degrades gracefully as the error rate increases, and (ii) useful performance improvements can still be obtained with relatively large error rates.

To investigate what the misclassification rate we could expect from a classifier, we implemented a simple non-disjoint k-nearest neighbour (KNN) classifier. The repository of documents remains the same, permitting us to measure the misclassification rate at 16%. Figures 3.4a and 3.4b show the cumulative distribution and median search length, respectively, in this case. Clearly, even at this error rate, significant improvements can be obtained, depending on the query distribution.

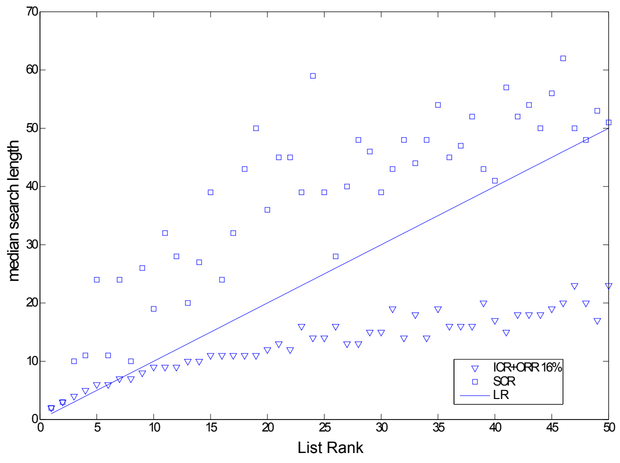
3.4.1.4 k-Nearest Neighbour Classification in a More Realistic Scenario

To investigate what misclassification rate expected from a classifier in a realistic scenario, we implemented a k-nearest neighbour classifier over a real search engine. Due to the absence of an oracle for retrieved results, we can only estimate the classifier's accuracy based on the target document, not on all retrieved documents. It may not fully reflect the performance of our classifier, but this measure can be used to approximate our system's performance.

Compared to the previous results shown in Figure 3.4a, Figure 3.5a shows that the misclassification rate of the k-nearest neighbour is about 30%, which is worse than the previous results. The performance degeneration is due to the fact we classify the target document using snippets generated from Google, rather than the full document in previous experiment. However, we are still able to see that these results are consistent with the previous conclusions. Moreover, for this more realistic case the classifier trained from Open Directory snippets



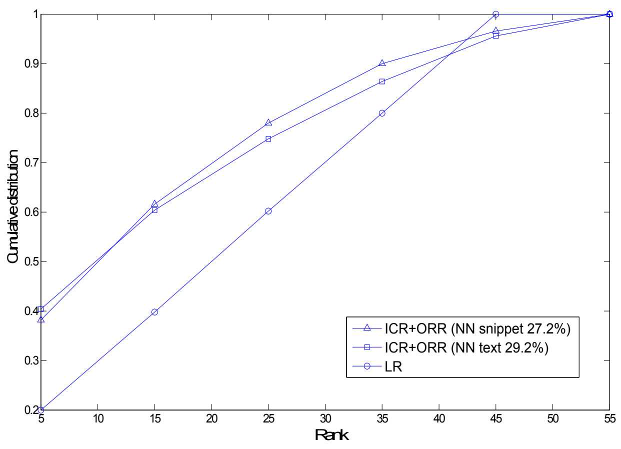
(a) cumulative distribution



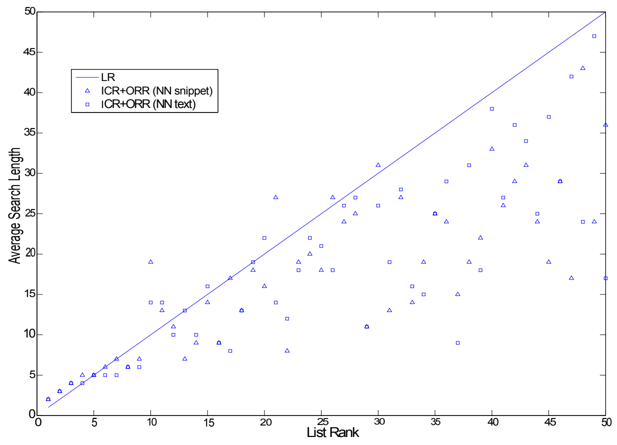
(b) Median search length of query results

Figure 3.4: Results for the KNN classifier with non-disjoint classes

reduces the misclassification rate to about 28%, which is slight better than that trained on the full text of web pages. Figure 3.5b shows that for poor



(a) cumulative distribution



(b) Median search length of query results

Figure 3.5: Results for the KNN classifier in real case

queries, the combined class rank will achieve a better performance than list rank. However, the trend in the curve is not as clear as the previous curve in Figure 3.4b. It is also evident that the curve has high variance. Nevertheless, our general conclusion that the hybrid-based search strategy performs better than a category-based or ranked-list alone, is still valid.

3.5 Discussion

We have examined how a hybrid model of an IR system might benefit a user. Our study was based on several novel ideas and assumptions.

In order to investigate the best-case performance, in Section 3.4 we constructed a system using a subset of the Open Directory. All documents in this subset have been manually classified and therefore provide “ground truth” for comparison. The advantage of our approach to rank classes is two-fold. It is simple, however, more importantly, this ranking closely approximates the list rank, thus allowing comparison between the class rank and list rank. In addition, we identified three classes of simulated users and rational user search strategies. By basing our evaluation using known-item search, we are able to simulate a very large number of user searches and therefore provide statistically significant experimental results. We acknowledge that real users may perform differently, and the experiments in Chapter 5 imply the correlation between our simulations, which provide an empirical upper bound on performance for real users and the behavior of real users.

Our experimental results in Section 5.3.3 not only demonstrate the advantage when the user correctly identifies the class and there is no machine error, but also suggest the strategy that the user should take to achieve the optimal performance when the user does not know the class or when there are user and machine errors.

Using the Open Directory, we were also able to control the error rates of both the user and the machine classification. Simulation results showed that the performance degrades gracefully as the error rate increases, and that even for error rates as high as 30%, significant reductions in search time can still be achieved. However, these reductions only occur when the query results in the target document having an initial list rank above a minimum rank, and this minimum rank increases with the error rate. In practice, a better classification performance can be achieved by incorporating classification at the stage of indexing web pages. Moreover, user error can be reduced when the user gains a better understanding of the system.

We believe that a category-based system may be more beneficial for informational queries [22] in terms of reducing the click distance [125], where the user will probably inspect several relevant search results. But for the case users may be satisfied with one relevant result, our experiments justified its advantage.

Chapter 4

Query Classification

Understanding the meaning of web search queries is a key task which is at the heart of web search research. However, it is one of the most difficult problems in information retrieval. A related issue is to understand the topics of web search queries, or the query classification problem, which is one component of our classification-based search engine. We will make use of query classification for class ranking in Chapter 5. The topics of web search queries can facilitate the understanding of general search queries, but classification of users' queries is also a challenging task due to the fact that queries are usually short and often ambiguous. One common solution is to enrich the query with external data. In this chapter, we will first explain the various data sources that can be used to enrich the query, followed by the classifier setup, experimental results will be presented next, and following that we will present a topic-specific analysis of search queries.

4.1 Introduction

There are various applications that motivate the research of query classification. These include paid placement advertising, classification/clustering of query results, query expansion, personalised search, and federated search. Query classification is a difficult task since queries usually consist of only a few terms, often leading to significant ambiguity.

To alleviate the problem of short queries, it is common to enrich the query with supplementary keywords, during both the training and testing phases. The supplementary keywords are derived from one of two broad sources. The first source is a form of pseudo-relevance feedback [104], in which it is assumed that the top-n documents retrieved in response to the query are relevant. Typically, the snippets associated with the top-n documents, and provided as part of the retrieved results, are used as the source of enrichment. This is discussed in more detail in Section 4.3. The second source of enrichment data comes from related information such as a thesaurus, or co-occurrence information present in search engine query logs.

Previous work [14] and our own experiments indicate that better performance is achieved based on the method of pseudo-relevance feedback. However, in some applications it may be desirable to perform query classification prior to, or in parallel with retrieval. For example, for paid placement advertising, it may be useful to perform query classification and the associated auction in parallel with the search. In addition to this, for federated search, if query classification is used to select which databases to access, then classification cannot be based on retrieval results. The motivation for our work is to study the performance of query classification in the absence of pseudo-relevance feedback when deploying the classifier.

In this chapter we investigate the performance of a query classification algorithm that uses a secondary source of information, namely Yahoo's suggested keywords, for query enrichment during testing. The main novelty of this work is the use of pseudo-relevance feedback data only during the training phase. That is, during training, we use retrieved Google snippets to enrich the query (pseudo-relevance feedback), but during testing/deployment, we use Yahoo's suggested keywords for query enrichment. In addition, we also adopt Yahoo! Explore Concepts to enrich queries, which can be regarded as pseudo-relevance feedback but with fewer terms of high quality. Section 4.3 describes the procedure in detail, while section 4.4 describes the SVM classifier used. The experimental results (see Section 4.5.3) indicate an improvement in performance over a symmetric learning strategy. Note specifically, if we train and test using Yahoo's suggested keywords, (symmetric training and testing), we achieve a classification accuracy rate of 44%. In contrast, if we train using pseudo-relevance feedback and testing using Yahoo's suggested keywords, we achieve an improved classification accuracy rate of 46%.

4.2 Related Work

There are three broad research themes within the query classification community. The first theme relates to the acquisition of training data. In many cases, labelled training data is sparse and/or expensive to acquire. To alleviate this issue, researchers have investigated using alternative sources of training data [110, 23]. This kind of alternative training data may not be optimal due to the absence of a direct mapping between the class structures. For example, some researchers have included training data from the Open Directory Project or the Yahoo! Web Directory, which has been used in web page classification tasks; However, their class structure may not be consistent with the one we are using.

The second theme relates to various different ways to enrich queries. There have been many proposals, some of which use pseudo-relevance feedback [110, 23], while others use secondary information sources [70]. Empirical results indicate that query enrichment can significantly improve performance [14]. Query enrichment based on pseudo-relevance feedback is usually superior to enrichment based on secondary sources. However, despite this superiority, there is a need to perform query classification prior to performing the retrieval, which precludes the use of enrichment based on pseudo-relevance feedback. Pseudo-relevance feedback is one type of relevance feedback. Relevance feedback modifies the query by making use of partial knowledge of known relevant and nonrelevant documents in order to improve search. The Rocchio algorithm is a relevance feedback mechanism introduced in and popularized by Salton's SMART system around 1970 [105]. The Rocchio algorithm generates a modified query $\vec{q}_m$ as follows [107],

$$\vec{q}_m = \alpha \vec{q}_0 + \beta \frac{1}{|D_r|} \sum_{\vec{d}_j \in D_r} \vec{d}_j - \gamma \frac{1}{|D_{nr}|} \sum_{\vec{d}_j \in D_{nr}} \vec{d}_j, \tag{4.1}$$

where D_r and D_{nr} are the set of known relevant and nonrelevant documents. $\vec{d}_j$ represent document vectors, and $|D_r|$, $|D_{nr}|$ is the corresponding Euclidian vector length of relevant document set and nonrelevant document set. The original query vector is q_0, and α, β, γ are the weights attached to each document. There are two possibilities in using query expansion [107],

- Full query expansion, where all terms contained in the previously retrieved relevant items are added to formulate the new feedback query.
- Partial query expansion, where only some of the terms present in the previously identified relevant items are incorporated into the query.

The partial query expansion is a feature selection strategy in order to remove the noise and Harman showed that adding only selected terms from retrieved relevant documents was more effective than adding all the terms, at least for the collections used in [62]. However, the user seldom provides a system with the relevance judgement needed in relevance feedback, pseudo-relevance feedback is proposed [35]. In pseudo relevance feedback, a small set of documents (usually the top n retrieved documents) are assumed to be relevant. Then a certain number of terms and phrases are added according to their weights in the relevant document set. However, even the top-ranked documents are not possible to cover the search topic exactly, and this incoherence characteristic of the top-ranked documents will cause query drift. Some research have been conducted to improve automatic query expansion for this problem[89].

The third theme relates to the number of classes in the query classification taxonomy. In many cases, the goal has been to classify a query into one or more of several broad classes. In this case the number of classes is typically less than 100. For example, Beitzel *et al.* [15] classify test queries into 18 classes , while for the 2005 KDDCUP, there were 67 predefined categories [110]. In contrast, other work, motivated by paid placement advertising, requires a much larger number of classes. For example, the taxonomy used in [23] has 6000 classes.

Our work focuses on the second theme, i.e., query enrichment. Our data set is derived from the same source (AOL) as that used in Beitzel *et al's* research [15]. Their method made use of a computational linguistics approach, called selectional preferences, combined with perceptron training and exact matching, which they call the *pre-retrieval* approach, since it does not require pseudo-relevance feedback. In a later refinement [14], they found that enriching the queries with snippets from search engine results outperformed their pre-retrieval solution, increasing the F1 score by 15%. As the experimental conditions of [14] are similar to ours, we compare our results to theirs in Section 4.5.3.7.

4.3 Query Enrichment

In this section, we describe two strategies for query enrichment. The first is based on pseudo-relevance feedback, (see Section 4.3.1), using Google snippets and Yahoo! Explore Concepts. The second is query enrichment based on Yahoo's suggested keywords (see Section 4.3.2).

4.3.1 Pseudo-Relevance Feedback

Pseudo-relevance feedback is a commonly employed approach for enrichment. It assumes that the top-n documents in the retrieved result set are relevant to a user's information need; here the documents refer to snippets or full web pages of retrieved URLs and are represented within the Vector Space Model (Section 4.4.1). We assume that the retrieved documents are snippets returned as part of the search engine result set. Note that the work of [110] used a similar methodology.

4.3.1.1 Yahoo! Explore Concepts

In addition to direct employment of pseudo relevance feedback, there are other more refined versions of pseudo relevance feedback. Yahoo! provides a resource called Yahoo Search Assist[1], which contains several pre-computed

[1]http://tools.search.yahoo.com/newsearch/searchassist

concepts related to the original user query called *Explore Concepts*. For example, if we search for “data mining”, Yahoo! will suggest related concepts such as “OLAP”, “knowledge discovery” and “business intelligence”. Explore Concepts is a source of high quality information related to the query that we can make use of.

Explore Concepts are generated in the following manner [7]:

- A concept dictionary is built from concept-rich sources such as query logs, web sites and entity name feeds. The majority of these concepts are either short noun phrases or proper names such as people and places.
- A term vector (metadata) is pre-computed when the search engine indexes crawl web pages; the term vector contains co-occurrences of terms appearing in the concept dictionary and in indexed web pages, which are intended to capture the key concepts represented in the document.
- The weights of the terms in the term vectors of the top-n retrieved results are averaged, and the top-m terms with the highest weights are retained in a *result set term vector*.
- *Similar phrases* are generated with the aid of a large search engine log to extend the initial set of candidate phrases.
- The result set term vector and the similar phrases together yield a set of candidate similar terms and are compared to the original query term via the cosine similarity measure based on their own result set term vector, which is used to rank the final similar phrases.

Thus, Yahoo! Explore Concepts can be regarded as a keyword extraction method of pseudo-relevance feedback. On average, Yahoo! displays about 15 related phrases associated with a query and we use those related phrases to enrich original queries. In the asymmetrical learning set up in Section 4.5.3.7, our experimental results indicate that Yahoo! Explore Concepts provides better results.

4.3.2 Term Co-occurrence from Secondary Sources

User query logs provide a valuable source for query classification, in particular, they provide different aspects of original queries. For example, “machine learning” is strongly correlated to “machine learning algorithm” and “machine learning research” found in query logs. The correlated terms serve as an alternative method for query enrichment. Rather than compute the co-occurrence information directly from a query log, we used Yahoo’s related suggestions[2], which are based on query logs, as the secondary source of enrichment. In practice, term co-occurrence information can be performed prior to search, while pseudo-relevance feedback can only be performed after performing the search.

4.4 The SVM Query Classifier

Section 4.4 describes the classifier used in our experiments. Section 4.4.1 then describes the data representation and Section 4.4.2 describes details relating to the use of support vector machines for multi-label multi-class classification. Section 4.4.3 describes our evaluation criteria, based on microaveraged precision, recall and F1.

Support Vector Machines (SVM) provide a state-of-the-art classification technique, which has successfully been applied to many text classification problems. In this work we apply SVM to query classification, and make use of the Liblinear[3] toolkit, which is an open-source package for solving large-scale regularised linear classification problems [50].

4.4.1 Data representation

We represent a document by the vector

$$\overrightarrow{d_i} = (log(1 + tf(1, i)) * idf_1, \dots, log(1 + tf(n, i)) * idf_n), \tag{4.2}$$

[2]http://developer.yahoo.com/search/web/V1/relatedSuggestion.html

[3]http://www.csie.ntu.edu.tw/∼cjlin/liblinear/

where $tf(j,i)$ is the frequency of term t_j in document d_i, and idf_j is defined by

$$idf_j = log(\frac{N}{df_j}),$$

where df_j is the number of documents in the collection which contain the term t_j and N is the cardinality of the document collection.

In the basic model, known as a uni-gram model, each feature represents a single term and is independent of other features. We can extend the uni-gram model by using pairs of adjacent terms as additional features, to obtain a model known as a bi-gram model. This leads to the *bi-gram SVM* (BSVM), which we use here. (In our experiments a bi-gram model includes uni-gram data.)

Each user query gives rise to a collection of snippets, which leads to two possible ways of representing an enhanced query vector, i.e.

$$\overrightarrow{q} = \frac{\sum_{i=1}^{N} \overrightarrow{d_i}}{N} \tag{4.3}$$

and

$$\overrightarrow{q} = \sum_{i=1}^{N} \overrightarrow{d_i}, \tag{4.4}$$

where N is the number of snippets, and $\overrightarrow{d_i}$ is the document vector for the ith snippet. Equation 4.3, which averages the vectors, is denoted by *AS*, while Equation 4.4, which sums the vectors, is denoted by *SS*. The enhanced query vector $\overrightarrow{q}$conveys the information of original query in vector space model and is used as the input for our classifier.

Equation 4.3 is directly related to the Rocchio algorithm, i.e., Equation 4.3 is equivalent to Equation 4.1 when setting β to 1, and α, γ to 0. Setting γ to 0 means irrelevant documents are discarded and setting α to 0 means ignoring the original queries. Here we use the full query expansion as we mentioned in Section 5.1.1, not the partial query expansion. Previous research [71, 21] observed that either no improvement or even small degradation of Support Vector Machines' performance when partial query expansion are used. Moreover, the partial query expansion will introduce another parameter, the number of selected terms, which increase the complexity of the system. We also notice that most top-ranked documents will contain the original search query and by setting γ to 0 will not affect the feedback performance.

4.4.2 Multi-label, Multi-class Classification

Query classification is inherently a multi-class, multi-label problem.

For query classification, we need to potentially deal with many classes, although SVM is inherently a two-class classifier. To reduce a multi-class problem to a binary problem, a common technique is to build multiple one-versus-the-rest classifiers with one category being the positive class and the remaining categories being the negative classes. This method will cause an *unbalanced* class distribution as the negative class is usually much larger than the positive class. In order to make it *balanced*, we can either use sampling to make the class size comparative or set several penalty parameters in the SVM formulation. In our experiment, we adopt the latter approach. The SVM toolkit, Liblinear, implements such a penalty function, which we use by setting a parameter, w, with a value equal to the ratio of each class size.

Query classification is also a multi-label problem [110], since each query can potentially be associated with several labels. When we convert a multi-class problem into a two-class problem, the multi-label problem can be solved simultaneously. If a query is associated with a number $|c|$ of labels, it is treated as a positive query when we build or test a model for one of the $|c|$ classes. For example, suppose that a query has two labels, say C_1 and C_2, then when we construct the model for C_1 or C_2, the query is merely regarded as a positive query for training. Furthermore, when we use the model of C_1 or C_2 to predict the query in the testing phase, its correct label should be positive.

4.4.3 Evaluation Criteria

To evaluate the performance, we use microaveraged precision, microaveraged recall and microaveraged F1, which are defined in Chapter 2. They are used because the overall performance measure will not be affected much by the poor performance of classes with few positive examples.

4.5 Experiments

In section 4.5.1 we describe the data set used, while in section 4.5.2 we explain the experimental setting. Finally, in section 4.5.3 we describe the experimental results.

4.5.1 Experimental Data Set

We make use of two manually classified subsets of an AOL search log [15]. The first one contains 9,913 manually classified queries resulting from a Master's level Information Science class assignment at Bar-Ilan University during 2007. The participants had prior knowledge of classification, cataloguing and indexing. The ontology we have adopted for classification consists of 31 top level categories, which constitutes a reasonable searcher's ontology. The 31 categories are listed in the "Category" column of Table 4.1. However, in the experiment, we discard four classes (*url*, *misspelling*, *noise* and *other*) which do not contain topic information at the semantic level; for more detail, we refer the reader to Section 4.6.

	Category	**AOL Category**		**Category**	**AOL Category**
1	Art		17	Misspelling	Misspellings
2	Auto	Autos	18	Nature	
3	Companies Business	Business	19	News	News
4	Computing	Computing	20	Noise	
5	Directories		21	Other	Other
6	Education		22	People	
7	Employment		23	Places	Places
8	Entertainment	Entertainment	24	Pornography	Porn
9	Finance Economy	Personal Finance	25	Religion	
10	Food and drink		26	Science	Research
11	Games	Games	27	Shopping	Shopping

Continued on next page

	Category	**AOL Category**		**Category**	**AOL Category**
12	Government, organizations and non-profit institutions	organization and institutions	28	Society& Community	
13	Health and Medicine	Health	29	Sports	Sports
14	Holiday	Holidays	30	Technology	
15	Home	Home	31	URL	URL
16	Law & Legislation		32		Travel

Table 4.1: The set of classes and the mapping between our classes and those of AOL

In the data set, we also incorporate labelled log data from AOL's research lab, which uses a similar ontology to ours. Table 4.1 tabulates that mapping of the AOL ontology to the one we use. After we remove duplicate queries, 17,862 distinct queries remain, and these form our labelled data set. Then we randomly divide the data set into 10 subsets, and learn the model from 9 subsets and test it on the remaining subset. The cross-validation process is repeated 10 times, with each of the 10 subsets use exactly once as the test data. The 10 results from the folds are averaged and reported. This is called 10 folds cross validation [63].

4.5.2 Experimental Setting

The notation we employ for the experiments is shown in Table 4.2. We used the settings in Table 4.2 to enrich both the training and test data. Here 2g is used to approximate a similar number of keywords as Explore Concepts has. By default, the data was represented as averaged snippets (Equation 4.3), using a balanced class setting and bi-gram SVM. The evaluation is based on the standard 10-fold cross-validation.

Notation	Description
q	query term without any enrichment
s	Enriched query with Related Suggestions from Yahoo!
e	Enriched query with Explore Concepts from Yahoo!.
(n)g	Enriched query with top n Google results, here n is 2, 5, 10 or 20, respectively, i.e., 2g, 5g, 10g, 20g

Table 4.2: Query enrichment terminology

4.5.3 Experimental Results

In the following tables TE and TR denotes the test and training data, respectively, and the performance measures are microaveraged.

4.5.3.1 Comparison of difference data representation

The first question we answer is which data representation is better. Averaged snippets (AS, see Equation 4.3) or summed snippets (SS, see Equation 4.4). Table 4.3 shows that AS significantly[4] outperforms SS.

Data rep.	TE	TR	recall	precision	F1
AS	10g	10g	52.57%	59.70%	**55.91%**
SS	10g	10g	44.97%	55.76%	49.79%

Table 4.3: The results for different data representations

The discrepancy between the two representation is due to the nonlinear transformation in Equation 4.2.

[4]This was tested by a paired-sample t-test at the 5% significance level.

4.5.3.2 Bi-gram SVM versus Uni-gram SVM

In Section 4.4.1 we mentioned that a bi-gram model has more features than a uni-gram model, and that the former has generally achieved better performance on text classification problems. We evaluate the two classifiers using 10g.

Model	TE	TR	recall	precision	F1
uni-gram	10g	10g	53.76%	50.64%	52.16%
bi-gram	10g	10g	52.57%	59.70%	**55.91%**

Table 4.4: The results for uni-gram and bi-gram SVM

The results in Table 4.4 show that bi-gram SVM is better than uni-gram SVM at the expense of a larger feature space.

4.5.3.3 Balanced Class versus Unbalanced Class

The class size distribution affects of the performance of many classifiers. If one class size is too small, then to minimise the error rate, the classifier will assign the label of the big class to all the data. Thus, to alleviate this problem, the penalty of misclassifying the data for the big class is increased. The comparison of the results is given in Table 4.5, and as can be seen, balancing the class distribution leads to significantly better results. Note, however that the precision of balanced classes is much worse than unbalanced classes, it is due to the fact that the unbalanced class model only predicts the most probable queries as positive.

Class dist.	TE	TR	recall	precision	F1
unbalanced	10g	10g	42.26%	71.53%	53.13%
balanced	10g	10g	52.57%	59.70%	**55.91%**

Table 4.5: The results of balanced and unbalanced classes

4.5.3.4 The Random Classifier

This baseline classifier randomly assigns one label to each user query. For the 27-classes classification, the probability of a *true positive*, *false positive* or *false negative* is $\frac{1}{27}$,$\frac{26}{27}$ and $\frac{26}{27}$, respectively. So for a random classifier precision, recall and F1 are 3.7%.

4.5.3.5 Symmetric Learning using Relevance Feedback for Query Enrichment

We first consider the case in which we enrich the query using relevance feedback during both training and testing, i.e. symmetric learning. In this case we averaged the term weights from the top-n snippets to construct an enhanced query vector, as described in Section 4.4.1, for both the training and test data.

The experimental result for this baseline is shown at Table 4.6, for various numbers of pseudo-relevant documents ranging from 2 to 20.

TE	TR	recall	precision	F1
2g	2g	47.22%	47.46%	47.32%
5g	5g	51.42%	57.06%	54.09%
10g	10g	52.57%	59.70%	**55.91%**
20g	20g	51.65%	60.62%	55.78%

Table 4.6: Classification results using symmetric training and testing with enrichment based on various numbers of pseudo-relevant snippets retrieved from Google.

The results in Table 4.6 show that best performance is obtained when the query is enriched using the top-10 snippets from the Google retrieved set. For 10g/10g, the F1 measure is approximately 56%. However, the improvement is not significant here.

4.5.3.6 Symmetric Learning using a Secondary Source of Information for Query Enrichment

Here we consider the performance when we enrich the query using a secondary source of information, namely Yahoo's suggested keywords, row s/s[5] in Table 4.7. For comparison, Table 4.7 also includes the resuls when no enrichment is applied (row q/q), the results when Yahoo! Explore Concepts is adopted (row e/e), and the case when only the top-2 results are used for pseudo-relevance feedback (row $2g/2g$). The last case is included because the number of enrichment terms is close to that provided by Yahoo suggested keywords. Significantly more terms are available if 5, 10 or 20 documents are used for enrichment using pseudo-relevance feedback.

TE	TR	recall	precision	F1
q	q	53.78%	17.01%	25.83%
e	e	50.81%	38.14%	43.56%
s	s	50.00%	39.14%	43.89%
2g	2g	47.22%	47.46%	**47.32%**

Table 4.7: Classification results using (i) no enrichment, (ii) enrichment based on Yahoo! Explore Concepts, (iii) enrichment based on Yahoo's suggested keywords, and (iv) the top-2 Google snippets. Training and testing is symmetric.

As expected, Table 4.7 shows performance without enrichment (q/q) is the worst. Enrichment using Yahoo's suggested keywords and Yahoo! Explore Concepts significantly improve performance, from an F1 score of about 26% to almost 44%. However, the difference between the latter two is too close to reach a significant difference. However, enriching with only the top-2 pseudo-relevant documents is superior (about 47%). Moreover, enrichment using 10g/10g (Table 4.6) is much better, with an F1 score of almost 56%. The results empirically justify the importance of an enrichment strategy for query classification.

We now investigate how asymmetric learning affects performance.

[5]This results is for queries which have related suggestions

4.5.3.7 Asymmetric Learning

In this approach we enrich the query using pseudo-relevance feedback with the top-10 Google snippets during training, but only enrich the query with asymmetric data, i.e., Yahoo's suggested keywords, Yahoo! Explore Concepts during testing.

TE	TR	recall	precision	F1
q	10g	27.24%	54.44%	36.31%
e	10g	43.86%	60.64%	50.90%
s	10g	37.84%	57.54%	45.65%
2g	10g	41.58%	59.10%	48.81%
10g	10g	52.74%	58.42%	**55.43%**

Table 4.8: Asymmetric training and testing. Training is performed using queries enriched with the top-10 Google snippets (10g). Note that row 10g/10g represents symmetric training, and testing and differs from the score reported in Table 6 due to pruning of the test set (see text for details.)

Table 4.8 indicates that performance using Yahoo's suggested keywords during testing is improved from an F1 score of 44% to an F1 score of 46%. This improvement is statistically significant at the 5% significance level. Note that Yahoo's suggested keywords do not provide suggestions for all queries in our test set. For our experiments, training was performed with *all* queries, as before. However, testing was only conducted on queries that could be enriched. The queries which can not find Yahoo's suggested keywords are discarded. This reduced the number of queries from 17,862 to 7,654. For comparison purposes, Table 4.7 also includes a row (10g/10g) reporting the scores for classification on this reduced test set, and as expected it yielded the best result. We also notice that Yahoo! Explore Concepts achieved the second best results when a similar amount of enrichment is used. In particularly, it is better than the 2g/10g case, since 2g contains the specific context of the query terms, which can not cover the broader concepts of the search queries. However, it is at the expense of extra computation as we describe in section 4.3.1.

4.6 Topic Analysis of Search Queries

We mentioned that understanding the meaning of queries is a key task which is at the heart of web search. One application of query classification is to help comprehend search engine logs. The analysis of search engine logs is important in order to understand how users interact with a search engine. Conventional analysis of search engine log data looks at various metrics such as query and session length aggregated over the full data set. Here we segment the data according to a topic taxonomy of web search and compute the metrics by topic basis.

Analysing query logs is one of the approaches that allows us to gain a deeper understanding of the ways in which users interact with search engines. Basic analysis of search logs provides mainly statistical outputs (e.g., most frequent query terms, number of query terms in a query, and length of sessions). Such analysis is of interest, since it provides insight into how an "average user" interacts with a search engine [112]. As an example of such analysis on a Microsoft query log from spring 2006, see [124]. A general set of guidelines for conducting search log analysis was given by [68].

Zhang and Moffat [124] provided some basic descriptive statistics of the 2006 MSN query log. Their computations were based on the whole set of queries. In this section we examine the topic specific statistical characteristics of the queries in the same log data by first applying to them the query classification algorithm that we have developed, in order to partition the data into categories.

The study considers a number of questions, including:

- Is the number of results viewed per query dependent on the query topic?
- Are the positions of the clicked results dependent on the query topic?
- Are there any significant differences in the average query length for different topics?
- Are there any significant differences in the session length for different topics?

- How do the topics distribute during the week?

We used the classifier presented in Section 4.4 to assign the top three labels to each of the queries from the MSN query log, which we obtained from Microsofts Live Labs. Prior to classification we pre-processed the query log by taking out non-alpha-numeric characters, such as “-”, from queries, stemming the queries, and then removing any single character queries. We classified queries in the URL class separately using a regular expression to detect the format of a URL.

We then enriched the input queries with Google top-5 results; a query which did not return any results was considered as noise. A query was consider to be in the “Other” class if its probability, as reported by the classifier, was below the prior probability of appearing in the largest class; this probability is the proportion, according to the classifier, of queries in the largest class. As a post-processing step after classification we sampled classes which had more the 500,000 queries to limit them to that number; in the future we intend to analyse full size classes.

4.6.1 Results

We will discuss the analysis of search queries based on topic in the following section.

4.6.1.1 Topic categories

The categories and the number of queries assigned to them appear in Table 4.9. Here a query belongs to a certain category if the category is among the top three labels assigned to the query. Note that the pornography category does not include the adult query set supplied by Microsoft in a separate log file.

Category	**# queries in category**	%total (14,923,285)
ART[AR]	2383	0.02%
Auto[AU]	639571	4.29%

Continued on next page

Category	**# queries in category**	%total (14,923,285)
Companies/Business[CB]	11479	0.08%
Computing[CO]	2026737	13.58%
Directories[DI]	18763	0.13%
Education[ED]	7868	0.05%
Employment[EM]	4356	0.03%
Entertainment[EN]	3079044	20.63%
Finance & Economy[FE]	506576	3.39%
Food and Drink[FD]	12005	0.08%
Games[GA]	440804	2.95%
Government,organisations,non-profit institutions[GO]	738652	4.95%
Health & Medicine[HM]	902993	6.05%
Holiday[HL]	477984	3.20%
Home[HO]	557131	3.73%
Law & Legislation[LL]	484	0.00%
Misspelling[MI]	1433872	9.61%
Nature[NA]	801	0.01%
News[NE]	1635456	10.96%
Noise[NO]	501925	3.36%
Other[OT]	451120	3.02%
People[PE]	33085	0.22%
Places[PL]	1917586	12.85%
Pornography[PO]	275539	1.85%
Religion[RE]	258	0.00%
Science[SC]	4356	0.03%
Shopping[SH]	3291744	22.06%
Society & Community[SO]	2998	0.02%
Technology[TE]	3177	0.02%
Sports[SP]	490097	3.28%

Continued on next page

Category	**# queries in category**	%total (14,923,285)
URL[UR]	1847778	12.38%

Table 4.9: The Categories

In table 4.9, category abbreviations are given in the brackets of the first column and the sum of the last column may exceed 100% since each query is categorised into up to three classes. For the current analysis, we combined *Law*, *Religion* and *Society & Community* into a single category, *Society-Combined* [SC]; and *Science*, *Technology* and *Nature* were also merged to create the category *Science-Combined* [ST]. Categories containing more than half a million queries were sampled randomly, in such a way that if a query was chosen then all the queries that appeared in the same session (session IDs were assigned by Microsoft) in the given category were also included. The size of each sample was approximately 500,000. Table 4.11 displays the sizes of the sampled sets (for categories up to size 507,000 there was no sampling) and the number of clickthroughs for the sampled queries.

4.6.1.2 User Assessment of the Classification

To assess the quality of the classifications, a group of 30 users assessed the classification results. Each user was presented with 470 queries, random samples of 10 queries from the smaller classes and 20 queries from the larger ones. The evaluators were instructed to mark the queries as "Y" if they thought that it was reasonable to classify the query in this category and "N" otherwise. They were also asked to suggest alternative categories for the queries and were told that a query may be classified into multiple classes although they were shown only one category for each query. If the meaning of the query was not clear to them, we advised then to submit the query to a search engine. To compute the margin of error for a 95% confidence interval for the given sample sizes (300 for the smaller classes and 600 for the larger ones), we assume a binomial distribution and a worst case scenario of $p = 0.5$ that overestimates the sample size, i.e. we assume apriori that both a "Y" and a "N" were equally likely. Using the standard formula that the sample size is approximately equal to $4p(1-p)/error^2$, we obtained an error margin of 6% for the smaller classes and 4% for the larger ones.

In order to assess the result we used macro- and micro- averaging as before, averaging across the two dimensions of classes on the one hand, and users on the other. The average results are consistent across the board, with a precision, recall and F1 of approximately 74.5%; the details are shown in Table 4.10. Note that this is over 20% higher than the experimental results when testing the classifier, and moreover we have trained the classifier on an AOL log but have used it in practice to classify an MSN log. This result convincingly shows that our methodology of applying topical analysis to query log data is sound.

Category	**Average satisfaction**	**Standard deviation**
ART[AR]	90%	13%
Auto[AU]	80%	14%
Companies/Business[CB]	89%	18%
Computing[CO]	51%	16%
Directories[DI]	78%	19%
Education[ED]	80%	18%
Employment[EM]	85%	12%
Entertainment[EN]	70%	14%
Finance & Economy[FE]	78%	16%
Food and Drink[FD]	91%	15%
Games[GA]	75%	11%
Government,organisations,non-profit institutions[GO]	62%	17%
Health & Medicine[HM]	75%	9%
Holiday[HL]	80%	15%
Home[HO]	65%	21%
Misspelling[MI]	76%	14%
News[NE]	50%	23%
Noise[NO]	82%	22%

Continued on next page

Category	**Average satisfaction**	**Standard deviation**
Other[OT]	44%	16%
People[PE]	87%	16%
Places[PL]	69%	18%
Pornography[PO]	79%	8%
Science-Combined[ST]	44%	18%
Shopping[SH]	71%	16%
Society-Combined[SC]	88%	16%
Sports[SP]	77%	14%
URL[UR]	88%	22%

Table 4.10: Satisfaction with categorization

4.6.1.3 Clickthrough

Table 4.11 shows the statistics for the clickthrough data. We see that in all except two cases (Home and Food) the total number of clickthroughs is less than the size of the query set. For the category *Noise* the size of the clickthrough set is especially small, but this is in accordance with the definition of this set.

Category	**sample size**	**# clicks**	**clicks per query ratio**
ART[AR]	2383	2165	0.91
Auto[AU]	500004	444110	0.89
Companies/Business[CB]	11479	10447	0.91
Computing[CO]	500004	373100	0.75
Directories[DI]	18763	17235	0.92
Education[ED]	7868	7320	0.93
Employment[EM]	4356	4032	0.93
Entertainment[EN]	500003	385219	0.77
Finance & Economy[FE]	506575	426952	0.84
Food and Drink[FD]	12005	12271	1.02

Continued on next page

Category	**sample size**	**# clicks**	**clicks per query ratio**
Games[GA]	440804	404595	0.92
Government,organisations,non-profit institutions[GO]	500003	401666	0.80
Health & Medicine[HM]	500004	462020	0.92
Holiday[HL]	477984	407551	0.85
Home[HO]	557131	570984	1.02
Misspelling[MI]	500003	199539	0.40
News[NE]	500001	383647	0.77
Noise[NO]	501925	19146	0.04
Other[OT]	451112	320530	0.71
People[PE]	33085	29664	0.90
Places[PL]	500002	388861	0.78
Pornography[PO]	275539	261746	0.95
Science-Combined[ST]	503908	416988	0.83
Shopping[SH]	500003	421752	0.84
Society-Combined[SC]	3740	3101	0.83
Sports[SP]	490097	414194	0.85
URL[UR]	500004	338833	0.68

Table 4.11: Sampled queries, number clickthroughs and clickthrough ratio

The low clickthough ratio is also understandable for the category *Misspelling*. The average clickthrough ratio was 0.81 and the standard deviation is 0.2. Thus all the results except for the above two that were mentioned are within one standard deviation. The values were considerably below the average for *Computing*, *URL* and *Other*. The results are somewhat surprising for *URL*, because it is assumed that when URL queries are entered to the search box, the user uses the search engine as a navigation tool, and thus we expect the user to click on the desired result. One possible explanation could be that many of these URLs are misspelled resulting in a relatively low clickthrough ratio. This point will be further investigated in the future research.

Another interesting point regarding clickthroughs, is the relatively large number of queries for which there is no clickthrough at all. This can be seen in Figure 4.1.

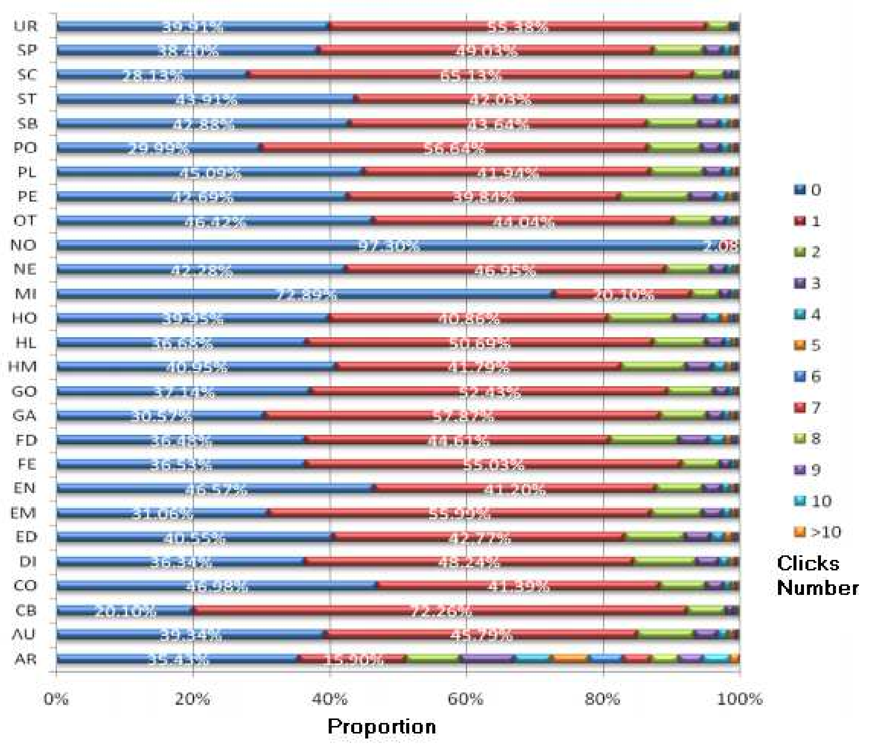


Figure 4.1: Distribution of number of clicks per query for the different categories

The percentage of the 0-clickthrough queries is 41.65% on average, and even if we exclude the categories Noise and Misspelling, the average is still 38.17%, while the average percentage of queries with one clickthrough is 47.66%. Thus 0 and 1-clickthrough queries cover on average more than 85% of the queries. There are a few cases where the number of 0-clickthrough queries is higher than the number of 1-clickthrough queries: *Computing*, *Entertainment*, *Other*, *People*, *Places*, *Shopping* and the obvious *Misspelling* and *Noise*. On the other hand, there are also a few cases where the number of 1-clickthrough is considerably higher than the average: *Companies*, *Entertainment*, *Games, Pornography*, *Society-combined* and *URL*. Note that *URL* has a low clickthrough ratio, and a high 1-clickthrough rate. This seems to imply, that in many cases there is no clickthrough at all, and if there is clickthrough, the user usually clicks on a single search result.

Next we limit the discussion only to the set of queries that had at least one clickthrough. Figure 4.2 displays the differences between the average number of clickthroughs. The median was 1 for all cases, the average is 1.42 and the standard deviation is 0.15. In Figure 4.2, the topics with high clickthroughs (more than one standard deviation above the average) are highlighted in red, while those with considerably lower than the average are highlighted in green. The high average clickthrough categories are *Art*, *Food* and *Home*; while the low average classes are *Companies*, *Finance & Economy* and *URL*.

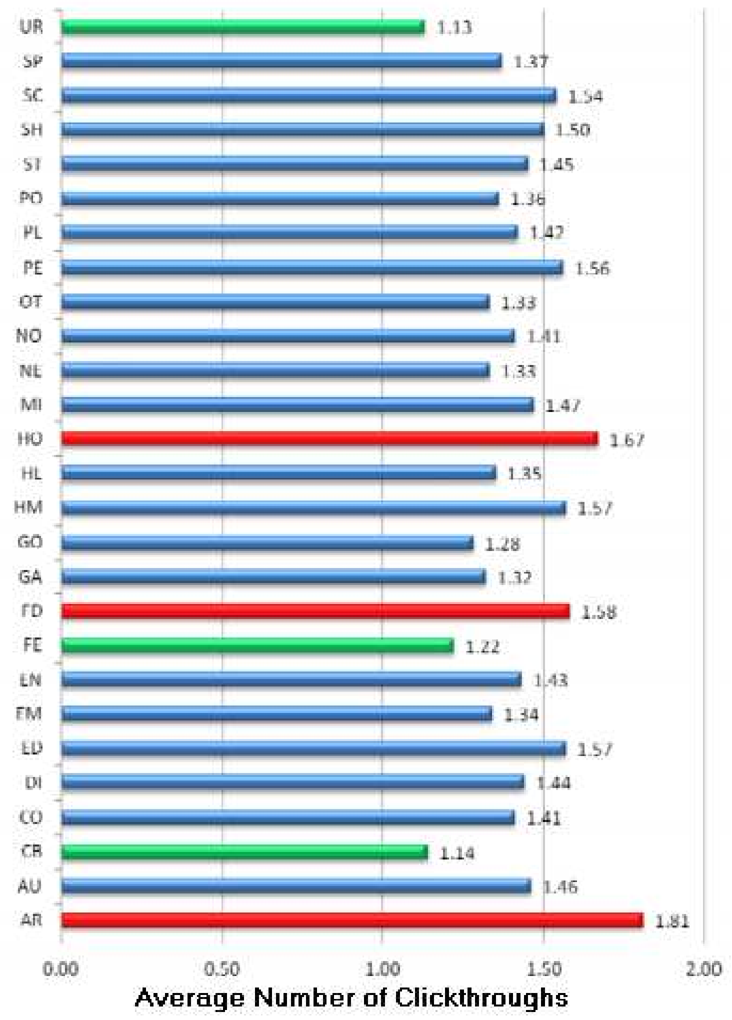


Figure 4.2: Average clickthrough rate of clicked queries

4.6.1.4 Clickthrough position

Here we look at the distribution of the position of the clicked result for all clickthroughs (see Figure 4.3) versus the first clickthrough for all the queries that were clicked (see Figure 4.4). The average percentage of clickthroughs at the top position is 48.2% with a standard deviation of 12.3% when we consider all the clicks. When considering only the first clickthrough for each clicked query, the top position is chosen in 59.9% of the cases on average, with a standard deviation of 10.1%. Thus there is a more pronounced tendency to click on the result in the top position when choosing to inspect the first search result (this is known as the presentation bias - see [12]). In this case too, we can see some topic-specific differences. The number of clicks on the top position is low for: *Art*, *Home* and *People*, and it is high for *Finance*, *Society-combined* and *URL*. When considering the first clickthrough only, the number of clicks on the top position is low for *Art* and *People* and high for *Employment*, *Finance*, *Society-combined* and *URL*. By low are referring to more than a standard deviation below the average.

4.6.1.5 Query length

For each topic, we measured the average and the median number of terms in the queries that were classified into the specific category. The results are displayed in Table 4.12.

Category	**average query length**	**median query length**
ART[AR]	1.77	2
Auto[AU]	2.59	2
Companies/Business[CB]	1.80	2
Computing[CO]	2.52	2
Directories[DI]	2.40	2
Education[ED]	2.80	2
Employment[EM]	2.09	2
Entertainment[EN]	2.52	2

Continued on next page

Category	**average query length**	**median query length**
Finance & Economy[FE]	2.11	2
Food and Drink[FD]	2.70	2
Games[GA]	2.02	2
Government,organisations,non-profit institutions[GO]	1.52	1
Health & Medicine[HM]	2.77	2
Holiday[HL]	2.69	2
Home[HO]	2.79	2
Misspelling[MI]	2.65	2
News[NE]	2.67	2
Noise[NO]	1.71	1
Other[OT]	2.03	2
People[PE]	2.24	2
Places[PL]	3.16	3
Pornography[PO]	1.98	1
Science-Combined[ST]	2.73	2
Shopping[SH]	2.48	2
Society-Combined[SC]	1.62	1
Sports[SP]	2.52	2
URL[UR]	1.00	1

Table 4.12: Average and median number of terms per query

Categories with especially low averages/medians are highlighted in green, and those with high averages/medians are highlighted in red. The mean of the averages is 2.29 and the standard deviation is 0.5. The longest queries on average (and also in terms of the median) belong to the category *Places*. Although even for Places most of the queries are short, the distribution is skewed as it is for other topics. Typical three term queries in Places are “Maui’s Fleming Beach” and “holiday inn express”, and an example of a longer query is “sunnyside fishing camp keewatin canada”. Such queries arise when users are looking for information on specific places they know about.

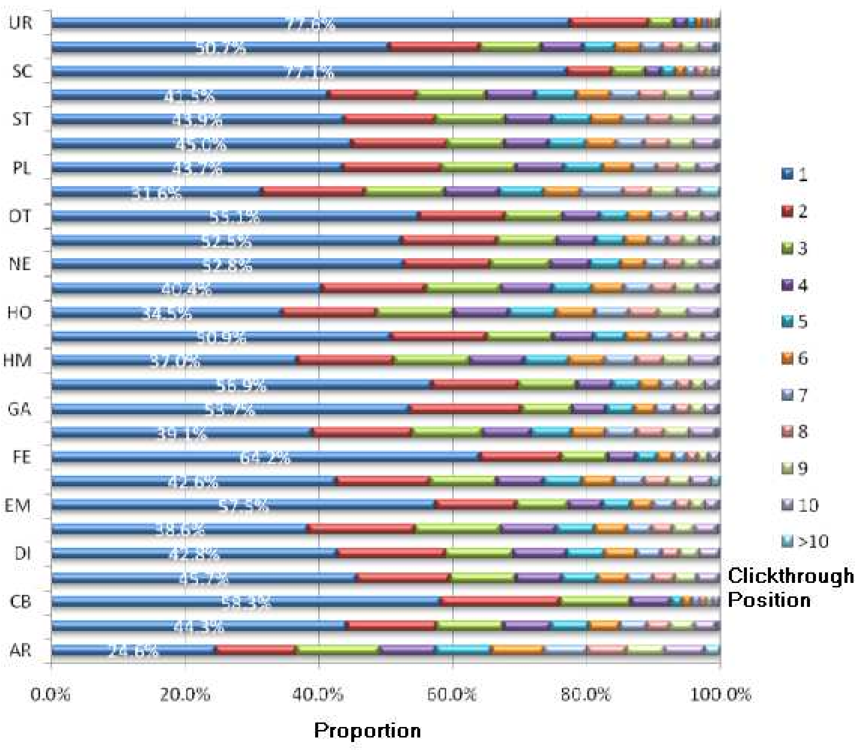


Figure 4.3: Clickthrough position - all clickthroughs

4.6.1.6 Session length

We measured the length of the sessions in terms of the number of queries that were issued within the same session.

Here we analyse the queries in the sessions, but only for those queries that were assigned to the given category, noting that sessions IDs were obtained from the log data. If for example a user asked five queries in a session, say, q1 ,q2, , q5, and q1, q4 and q5 belong to category *A* and q2 and q3 to category *B*, then when analysing category *A* we consider only q1, q4 and q5, and we consider the session to consist only of those queries. That is, the length of this session with respect to category *A* is three and we calculate the time difference between q1 and q4 and between q4 and q5 as time between the queries in this category *A* session.

The results are displayed in Table 4.13. The average session length for all the

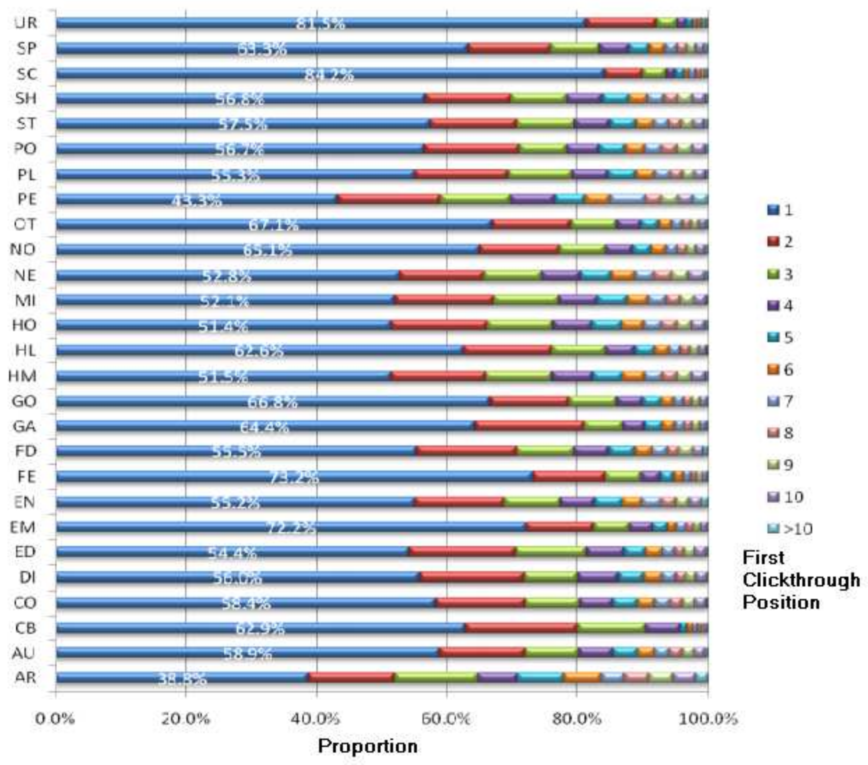


Figure 4.4: Clickthrough position - first clickthrough for each clicked query

categories is 1.64 with standard deviation 0.40. Values one standard deviation below and above the average are highlighted.

Category	**average session length (queries)**	**median session length**
ART[AR]	1.39	1
Auto[AU]	1.85	1
Companies/Business[CB]	1.17	1
Computing[CO]	2.00	1
Directories[DI]	1.34	1
Education[ED]	1.39	1
Employment[EM]	1.22	1
Entertainment[EN]	2.56	2

Continued on next page

Category	**average session length (queries)**	**median session length**
Finance & Economy[FE]	1.33	1
Food and Drink[FD]	1.30	1
Games[GA]	1.41	1
Government,organisations,non-profit institutions[GO]	1.51	1
Health & Medicine[HM]	1.91	1
Holiday[HL]	1.52	1
Home[HO]	1.57	1
Misspelling[MI]	1.94	1
News[NE]	1.89	1
Noise[NO]	1.47	1
Other[OT]	1.25	1
People[PE]	1.36	1
Places[PL]	2.27	1
Pornography[PO]	1.55	1
Science-Combined[ST]	2.00	1
Shopping[SH]	2.58	2
Society-Combined[SC]	1.19	1
Sports[SP]	1.48	1
URL[UR]	1.82	1

Table 4.13: Session length

We observe that there are some topic-specific differences. Somewhat to our surprise the shortest sessions are not in the category *URL*, *Misspelling* or *Noise*. An explanation for this could be, that if someone uses the search engine as a navigation tool, he/she does so consistently and therefore issues several URL queries in a session. An explanation of the calculated average session length for *Misspelling* could be that if a user makes a spelling mistake once, he/she is likely to continue making such mistakes.

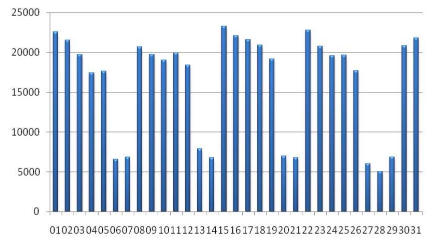
4.6.1.7 Temporal characteristics

Here we calculate the daily query volumes per category for May 2006 (the log provided by Microsoft covers this period). It had been noted by Zhang and Moffat [124], that there is a clear drop in the number of issued queries during weekends (6 & 7 May, 13 & 14 May, 20 & 21 May and 27, 28 & 29 May - Monday, May 29 2006 was Memorial Day in the US, a public holiday).

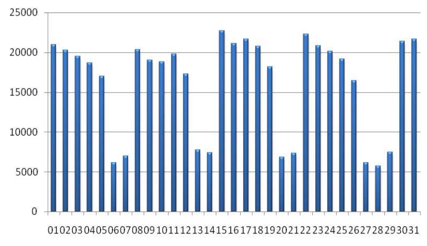
When looking at the category specific distributions, we also notice this drop, but the extent of the drop varies with the category. We examined all the large categories plus *Pornography.* The largest drop in the number of issued queries was observed for *Finance* and for *Government* (see Figures 4.5a and 4.5b). In these two categories the number of queries submitted on weekends and holidays is less than 40% of the number of queries submitted on the other days. Note that for both categories, in most cases Monday (or the first working day of the week) is the peak day. This seems to indicate that most of the queries in these categories are work related.

There is a second, much larger group of categories, where the number of queries submitted during weekends and holidays is between 40% and 60% of the number of queries submitted on weekdays. Categories belonging to this group include: *Computing* (Figure 4.5c), *Shopping* (Figure 4.5d) and *Auto* (Figure 4.5e). It seems that there is continued interest in these topics both during weekdays and weekends.

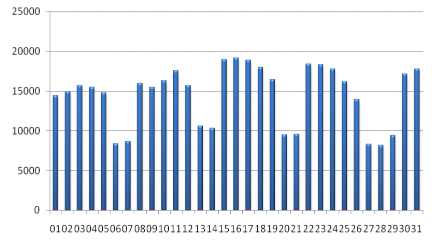
In the third group there is hardly any drop in the weekend, i.e., the volume during weekends and holidays is at least 60% of the volume on weekdays. Categories in this group include *Games* (Figure 4.5f) and *Pornography* (Figure 4.5g) and *Entertainment* (Figure 4.5h). These topics can be characterised as recreation, so it is easy to understand why the volume of queries remains high for these topics during weekends as well.



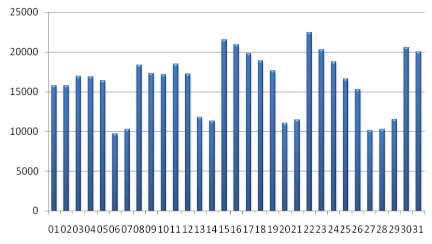
(a) Finance



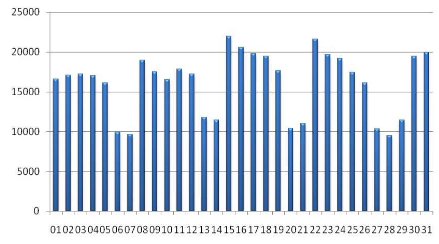
(b) Government, organizations and non-profit institutions



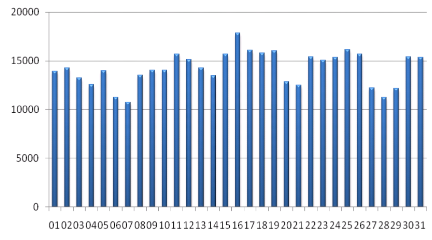
(c) Computing



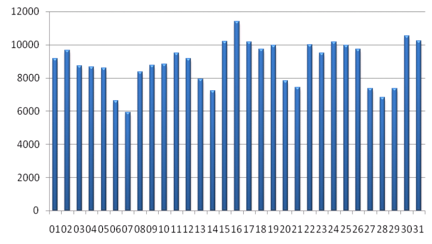
(d) Shopping



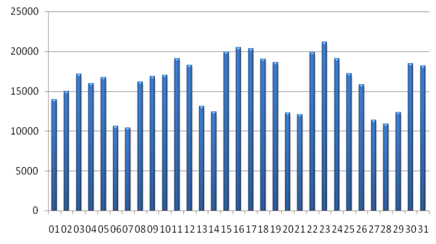
(e) Auto



(f) Games



(g) Pornography



(h) Entertainment

Figure 4.5: Daily volume of the queries in specific categories

4.7 Discussion

In this chapter, we examine the query classification problem. In some applications, e.g. federated search, there is a need to perform query classification prior to performing the search. In such a case, enriching the query with results derived from pseudo-relevance feedback is not possible. Without pseudo-relevance feedback, query enrichment must be accomplished using secondary sources of information. This leads to the case of asymmetric learning.

We considered the use of Yahoo's suggested keywords and Explore Concepts in Section 4.3 as such a source of secondary information. Table 4.6 and Table 4.7 show that the performance, as measured by microaveraged F1, is significantly less than can be obtained using pseudo-relevance feedback. Specifically, our experiments showed that using a symmetric testing and training procedure, enriching with Yahoo's suggested keywords produced an F1 score of about 44%, which is substantially less than enrichment with the top-10 snippets from Google, when the F1 score is about 56%. Even if only enriching with a similar number of terms, i.e. using the top-2 snippets, the F1 score is 47%, which is about 3% better than using Yahoo suggested keywords.

To improve performance, we investigated an asymmetric learning strategy in Table 4.8. Training was performed with queries enriched by the top-10 snippets from Google, but testing was performed using queries enriched by Yahoo's suggested keywords. In this case, performance increased from 44% to 46%. This is well below that which can be obtained using a symmetric learning procedure and enrichment based on the top-10 snippets. However, if we compare this to the performance achieved when only the top-2 snippets are used for enrichment, i.e. a similar number of enrichment terms are used, the performance is comparable. This suggests that if a sufficiently rich source of secondary information is available, it may be possible for enrichment based on secondary sources to be competitive with those based on pseudo-relevance feedback. In addition, we also compared Yahoo's suggested keywords to Yahoo! Explore Concepts, which is a method of keyword extraction comparable to that of the pseudo-relevance feedback. Given the similar amount of features, Yahoo! Explore Concepts yielded the best results. This implies that if we can improve the quality of the enrichment data, the performance can also be improved.

Query classification provides another dimension to understand search query logs. We have carried out an analysis on a substantial search engine log in Section 4.6, where the analysis was computed on a topic-by-topic basis, from a top level ontology pertaining to web search. In order to be able to analyse the data, we classified the queries using an algorithm we have developed in section 4.4. Although this algorithm is far from perfect, we believe that its results are indicative of the topics, given the relatively high user satisfaction with the results of the classification. Our results show that, for each of the metrics we have presented, there are outlier classes which exhibit deviant behavior. We note that as expected, the statistics for *Noise* and *Misspelling* are often different in figure 4.1 and table 4.11, but in many cases we discovered other classes with deviant behavior. For example, the longest queries are for *Places* in table 4.12, the largest clickthrough rate for *Art* in figure 4.2 and the largest session rates for *Shopping* and *Entertainment* in table 4.13. In table 4.5 we also discovered that the daily volume of queries is also sensitive to the topic.

We suggest that some of the differences we have observed could have an impact on the way search engines deal with queries from different categories.

Chapter 5

Ranking Classes of Search Engine Results

In this chapter, we will study the problem of class ranking, in particularly, we investigate the query independent features associated with the classification process, such as class size, top ranked document within each class and query dependent features, i.e., the class probability distribution of a test query derived from query classification in Chapter 4. A proof of concept for a classification-based search engine is presented as a demonstration of the system.

5.1 Introduction

In chapter 3, we compared the performance of traditional search engines with classification-based search. Using target testing, we quantified the benefits of grouping results in comparison to a standard search engine in terms of the rank of the target documents in the result set. However, there are two issues we have not yet addressed.

The first issue concerns target testing that the queries are computer-generated, which may not fully reflect real users' search intentions and the correlation between the queries and the target documents.

The second is related to the ranking method. Despite considerable research on document categorisation and document ranking, there has only been a small amount of research effort on how to rank categories.

To address these issues, we investigate various approaches of ranking classes and how such rankings affect the number of classes and documents a user must inspect prior to finding a document satisfying his or her information need. In particular, the information need is conveyed in users' queries from a web query log and the desired documents are embedded in the clickthrough data. The main applications of our work are not only grouping retrieved data, but also the investigation of ranking models for them.

5.1.1 Related Work

There has been significant work on clustering and classification of search results [29, 31, 126]. Our interest in this chapter is specifically focused on how previous research dealt with the problem of ranking classes. There are two main approaches in this respect.

The first approach [31] is to rank classes based on their size, i.e. the top-ranked class contains more documents than the other classes. This approach is simple and assumes that a class's relevance is purely a function of the number of documents it contains.

The second approach [125, 126] is to rank classes based on various features associated with the documents in each class. For example, in [125] the authors ordered the clusters by their "estimated coherence", defined as a function of the number of documents containing each phrase and the number of words that make up its phrase. In [126], all n-grams ($n \leq 3$) were first extracted from search results as candidate phrases, and salient scores were calculated by a regression model trained from manually labelled data. The features includes phrase frequency/inverted document frequency, phrase length, intra-cluster similarity, cluster entropy and phrase independence. The phrases with top salient scores, called salient phrases, are the names of candidate clusters. Finally, they are further merged according to the overlapping part of the two clusters. The salient scores are used to rank the clusters.

A second important consideration is how documents are ranked within a class. Several alternatives have been proposed. However, in the work described here, the relative ranking of documents within a class is taken to be the same as their relative ranking in the original result set. We believe that this is an important experimental design consideration. In particular, if the ranking of documents is altered within a class, then it is very difficult to determine whether any improvement is due to (i) the class ranking, (ii) the new document ranking, or (iii) a combination of (i) and (ii). Thus, in order to eliminate this potential ambiguity, we maintained relative document rankings within classes. Therefore any improvement must be only due to the class ranking.

5.2 Class-Based Ranking Method

Before we discuss the various class ranking algorithms we examined, it is useful to first describe how we evaluated performance. For a conventional system, in which the result set is displayed as a ranked list of documents, i.e. we have a list of documents $\{d_1, d_2, \cdots d_N\}$, if the k-th document is the desired document, then the user must look at k documents $(d_1 \cdots d_k)$. We refer to k as the "list rank" of the document, since it was ranked k in the one-dimensional list of retrieved documents. Clearly, the lower the list rank, the quicker the user will find the desired document.

The performance of a classification-based system is more complicated to define. Consider the case where the user is looking for document, $d_{i,j}$, where i denotes the rank of the class the document is contained in, and j is the document's rank within this class. Thus, a user must look at i class labels and then j document snippets in order to find document, $d_{i,j}$, a total of $(i + j)$ classes and documents. We referred to this as *in-class rank* in Chapter 3.

For any classification-based system, we compare a document's in-class rank to its original, corresponding list rank, k. We say that the classification-based system outperforms the list-based system if $i + j < k$, i.e., the user looks at fewer classes and documents.

Note that we have implicitly assumed that (i) documents are correctly assigned to classes, and (ii) that users always choose the correct class. In practice, this will not always be the case. However, this assumption simplifies our analysis and permits us to determine the best-case performance of class-based retrieval systems, therefore *in-class rank* provides the upper bound of the systems. We refer the reader to Chapter 3 for more discussion of when this assumption does not hold.

Given an initial retrieval set, $D = \{d_1 \cdots d_{|D|}\}$, that has been grouped into a set of classes, $C = \{c_1 \cdots c_{|C|}\}$, we now wish to determine the relative ranking of each class. The information we have available is the query, q, and the documents, D. We employ a straightforward Bayesian approach, i.e. we wish to estimate the probability, $P(c|q)$, that class, c is of interest, conditioned on the query, q,

$$P(c|q) \propto P(c)P(q|c). \tag{5.1}$$

The value of $P(c|q)$ is used to determine the class rank(CR). We now consider how each of these two terms might be estimated.

5.2.1 Query-Dependent Rank

The probability, $P(q|c)$, is the likelihood that class c generates query q. According to Bayesian rule, it can be regarded as solving the query classification problem [110]. This problem has received significant attention [23, 26] since the 2005 KDD Cup competition [80]. Solutions to this problem typically enrich the query terms using keywords from the top-ranked documents in the result set; See chapter 4 for more detail on query classification.

For a test query, the class probability distribution can be modeled as

$$P(q|c) \approx \frac{1}{1 + exp(-(w_c^T x + b))}, \tag{5.2}$$

where, x is term vector containing the query and enrichment terms, and the weights w_c and intercept b are derived from L2-regularised logistic regression [82], based on a set of labelled examples. Alternative estimates for $P(q|c)$ are possible, but are not considered here.

5.2.1.1 Query-Based Rank (QR)

If each class is only ranked based on Equation (5.2), i.e. we ignore the query-independent term, $P(c)$, in Equation (5.1), we refer to it as *query-based rank (QR)*.

5.2.2 Query-Independent Rank

To estimate $P(c)$, we make use of the available documents in the retrieved result set. Note that we can estimate $P(c)$ based on the class distribution of the collection, but this is beyond our scope and it is not as accurate as the retrieved result set. We further assume that only documents contained in the class, D_c, affect the probability of the class, i.e.

$$P(c) \approx P(c|D_c). \tag{5.3}$$

We believe this assumption is reasonable as the class probability is mainly determined by the information within the class, not by the other classes. Thus, Equation (5.1) becomes

$$P(c|q) \approx P(c|D_c)P(q|c). \tag{5.4}$$

We now considered several ways to estimate the conditional probability $P(c|D_c)$; More precisely, we present functions to approximate the likelihood of the class c to be examined rather than probability, as the results do not sum to 1.

5.2.2.1 Document-based Rank (DR)

One approach to estimating $P(c_i|D_c)$ is to base the probability on the top-ranked document in the class, c_i. The reader is reminded that the original rank order of documents in the result set is retained within a class.

The j-th ranked document in class, c_i, is denoted $d_{i,j}$. The document's corresponding list rank, i.e. its rank prior to classification, is denoted $s(d_{i,j}) = s(d_k) = k$.

We then define the conditional probability, $P(c|D_c)$ as

$$P(c|D_c) = f(s(d_{i,1})), \tag{5.5}$$

where $c = c_i$, and $s(d_{i,1})$ is the list rank of the top document in class c_i. The function, $f(x)$, can be any monotonically decreasing function in the value x. Here we consider the inverse function defined by

$$f(x) = \frac{1}{x} \tag{5.6}$$

and the logistic function defined by

$$f(x) = \frac{1}{1 + exp(x)}. \tag{5.7}$$

If we only rank classes based on the query-independent factor of Equation (5.1), then both functions, $f(x)$, will rank the classes in the same order. In the subsequent experiments, we therefore only consider the inverse function, and rank classes according to

$$P(c|D_c) = \frac{1}{s(d_{i,1})} \tag{5.8}$$

We refer to this as the *document-based rank* (DR).

5.2.2.2 Size Rank (SR)

In contrast to ranking classes based on the top-ranked document in the class, we also consider the case where the conditional probability $P(c|D_c)$ is based on the class size. That is, the bigger the class, i.e. the more documents assigned to the class, the more important the class is considered to be. Thus, we have

$$P(c) \approx P(c|D_c) = \frac{|c|}{\sum_i |c_i|}, \tag{5.9}$$

where $|c|$ is the number of elements in the class c, and the denominator is the size of result set.

Again, if we only rank classes based on the query-independent factor of Equation (5.1), and the class ranks are based on the size of the classes, as defined in Equation (5.9), then we refer to this as the *Size Rank* (SR).

5.2.3 Additional Class Ranking Models

From Equation (5.2) and the definitions for $P(q|c)$ and $P(c)$, we can now define a variety of different models for ranking classes based on both the query-dependent and query-independent probabilities.

5.2.3.1 Query/ Inverse Rank (QD_IR)

If the class ranking is determined by the product of Equations (5.2) and (5.6), then we obtain

$$P(c|q) \approx \frac{1}{1 + exp(-(w_c^T x + b))} \times \frac{1}{s(d_{c,1})}. \tag{5.10}$$

We call this rank the *Query/Inverse Rank(QD_IR).*

5.2.3.2 Query/Logistic Rank (QD_LR)

The *Query/Logistic Rank (QD_LR)* is correspondingly defined as

$$P(c|q) \approx \frac{1}{1 + exp(-(w_c^T x + b))} \times \frac{1}{1 + exp(s(d_{c,1}))}. \tag{5.11}$$

5.2.3.3 Query/Size Rank (QSR)

Similarly, if the class ranking is determined by the product of Equations (5.2) and (5.9), then we have

$$P(c|q) \approx \frac{1}{1 + exp(-(w_c^T x + b))} \times \frac{|c|}{\sum_i |c_i|}. \tag{5.12}$$

We call this rank the *Query/Size Rank (QSR).*

5.2.3.4 Summary of Ranking Methods

We distinguish the methods for estimating $P(c|q) \approx P(q|c)P(c|D_c)$ according to the different methods presented above. The *list rank* (LR) is the original rank of a document in the result set, i.e. before any classification. We then consider two query-independent methods of ranking classes, based on (i) the class size, i.e. *size rank* (SR), and (ii) the top-ranked document in each class, i.e. *document rank* (DR). We also consider ranking classes based only of the query-dependent term, i.e. the *query-based rank* (QR). Finally, we consider ranking classes based on a combination of query-dependent and query-independent terms. In all these cases, the query-dependent term is based on Equation (5.2), and we vary the query-independent term. Specifically, we consider (i) *query/size rank* (QSR) in which the conditional probability, $P(c|D_c)$ is based on the size of a class, and (ii) *query inverse rank* (QD_IR) and *query logistic rank* (QD_LR), both of which are based on a function of the top-ranked document in each class, and where this function is the inverse function or the logistic function, respectively. The various methods are summarised in Table 5.1.

Notation	Meaning
LR	List Rank, the rank of the results returned by the search engine.
SR	Size-based Rank computed according to (Equation (5.9))
DR	Document-based Rank computed according to (Equation (5.6)).
QR	Query-Based Rank computed according to (Equation (5.2))
QSR	Query/Size Rank computed according to (Equation (5.12))
QD_IR	Query/Inverse Rank computed according to (Equation (5.10)).
QD_LR	Query/Logistic Rank computed according to (Equation (5.11)).

Table 5.1: The summary of the ranks

5.3 Experiments

In this section, we will conduct the experiments to evaluate the impact of the the different rank methods we have presented in the previous section.

5.3.1 Experimental Setting

Evaluation of information retrieval systems requires knowledge of a document's relevance with respect to a query. One indirect source of such information is query logs. These logs consist of queries together with associated clickthrough data. Previous research [88] showed that retrieval evaluation based on query logs yields similar performance to retrieval evaluation based on traditional human assessors. We used a subset of an MSN query log (the RFP 2006 dataset), collected in spring 2006. The log contains approximately 15 millions queries. Each query has associated with it either (i) no clickthrough data (*no-click*), (ii) one clickthrough data (*one-click*), or (iii) multiple clickthrough data (*multiple-click*).

We ignored queries for which there is no associated clickthrough data (approximately 6.1 million queries) as they do not contain any feedback. In general, there are two types of search session that will be abandoned. Firstly, if the session information is rare, which is measured by the frequency of a query or a query clickthrough pair. Such session provides less or even no value for user browse model, query suggestion and other tasks, so it will be abandoned [49, 27]. Secondly, if the session does not contain any useful information for research tasks, it will be abandoned. Dupret et al [48] built a session utility model based on clickthroughs, therefore any session without a clickthrough is discarded, which is close to our work. Boldi et al [20] built a query-flow graph with a directed edge from one query to another query by making use of queries in one session. The sessions containing one query are discarded as they do not provide useful information for their task. For one-click queries, of which there are approximately 7.2 million, we assume the query was satisfied by the document clicked on. For multiple-click queries, of which there are approximately 1.6 million, we assume that the query was satisfied by the last document clicked on. We realise that this will not always be true, but assume that it is true sufficiently often to provide us with reliable results. Note that this assumption has been partially justified by other researchers [75], in the context of multiple-click queries.

The query log does not include information describing the result set returned in response to the query. Rather, the clickthrough data only identifies those documents in the result set that the user clicked on. Of course, in order to evaluate the various classification based ranking methods, we need access to the complete result set. We acquired this information by issuing the query to a search engine, specifically Microsoft Live Search on May 2009, which was subsequently replaced by Bing. Note that for some queries, the URLs clicked on in the query log are not returned by the search engine, either because its rank is beyond our retrieved result set or the URL is no longer available. We discarded such queries. In the case where the URL is returned in the result set, we assume that the result set returned by Live Search is similar to the result set observed by the user during the collection of the log. We acknowledge that this is a major assumption, which cannot be verified due to the evolution of the web and users' search behavior. To conduct a known item test, we have to assume that the clickthrough data still attract users' attention even some new relevant results may occur in the retrieved data set in practice. We can also carry out our experiment on a standard Web TREC collection, which provide static data set. However, TREC collect is mainly used to evaluate the retrieval algorithms, while our work is built upon the underlying retrieval methods. Our classifier predicts the label based on the retrieved snippets online and the TREC collection does not provide such information. Previous research [111] also presented some issues to use TREC algorithm for web search. Nevertheless, TREC is another option for us. Future work is needed to repeat these experiments on a TREC collection or more recent data set.

The experimental methodology will now be described. The total number of unique queries which have a clickthrough is 3,545,500. Among them, there are 658,000 multiple-click queries, whose top-20 search results have been downloaded by us before Microsoft upgraded Live to Bing. We took a random sample of 20,000 one-click queries and 20,000 multiple-click queries, whose clickthrough occurred both in the query log and in our retrieved data set.

For each query, the list rank of the relevant document (i.e. the document clicked on for one-click or the final document clicked on for multiple-click) were recorded. Next, the documents in the result set were classified into one of 27 classes; these classes are enumerated in the Chapter 4; see Section 4.6 for more details about this ontology for searchers. In order to classify the documents we compute $P(c|q)$ from Equation (5.2), using logistic regression. The training data is obtained from two manually classified subsets of an AOL search log [15]. For details about the training data, please refer to Chapter 4. We enriched the query with the top-10 snippets in the result set, the titles of the top-10 documents and their URLs to form the vector x. Then for the test query, we enriched the query with the same information and predicted the probability via Equation (5.2).

To keep our data consistent, for a given query, we record the list rank of the given clickthrough in the result set, as it may be different from the one recorded in the log data. Then query classification is carried out by enriching the query with the top-10 results. In this manner we attain the class probability distribution. After that we assign each result into its class.

5.3.2 Experimental Methodology

The experimental procedure, including data pre-processing have been described in the Section 5.3.1. We assume the last clickthrough of multiple-click queries and the clickthrough of one-click queries as the target document. We then compare the list rank with the class rank methods described in the Section 5.2. The classification of search results is carried out using the same query classifier as we described in Chapter 4. The class with highest probability is the predicted label for the result. And document ranking within each class is determined by the original list rank.

5.3.3 Experimental Results

In Section 5.3.3.1, we present the results for multiple-click queries. The results of One-click queries are presented in Section 5.3.3.2.

5.3.3.1 Results for Multiple-Click Queries

Each target document has an original list rank from 1 to 20. Table 5.2 shows for each list rank, the mean value of the corresponding in-class rank. Column 1 of Table 5.2 provides the list rank of the target documents for the top-20 documents, i.e., in this case there is no classification, only a traditional list of retrieved documents. Column 2 to 7 provide the equivalent in-class ranks (ICR), i.e., the total number of classes and documents a user must examine in order to find the target document. If the class-based rank is less than the LR, then the classification based system outperforms a traditional system. The smallest value indicates the least number of documents that a user must inspect before finding the desired document.

LR	**QR**	**DR**	**SR**	**QSR**	**QD_LR**	**QD_IR**
1	3.00	**2.00**	2.88	2.79	2.08	2.13
2	3.55	3.00	3.39	3.32	**2.93**	3.01
3	3.95	3.64	3.73	3.68	**3.58**	3.63
4	4.45	4.23	4.20	4.20	**4.15**	4.18
5	4.81	4.80	4.63	**4.58**	4.71	4.70
6	5.28	5.26	**4.95**	4.99	5.19	5.18
7	5.65	5.86	**5.41**	5.42	5.74	5.68
8	6.11	6.24	5.86	**5.85**	6.15	6.12
9	6.37	6.58	**6.13**	**6.13**	6.49	6.44
10	6.83	7.05	**6.54**	6.56	6.97	6.89
11	7.46	7.69	**7.16**	7.18	7.60	7.57
12	7.89	8.07	**7.52**	7.59	8.01	7.93
13	8.38	8.58	**7.99**	8.05	8.50	8.41
14	8.41	8.66	**8.07**	8.13	8.57	8.50
15	8.99	9.17	**8.56**	8.68	9.10	9.05
16	9.55	9.84	**9.25**	9.32	9.74	9.64
17	9.64	10.03	**9.23**	9.29	9.94	9.83
18	10.50	10.82	**10.12**	10.17	10.72	10.63

Continued on next page

LR	**QR**	**DR**	**SR**	**QSR**	**QD_LR**	**QD_IR**
19	10.69	10.95	**10.27**	10.36	10.88	10.80
20	11.27	11.52	10.98	**10.95**	11.41	11.32

Table 5.2: The comparison of class based rank for the last click through according to list rank (multiple-click).

One reason why the class-based rankings do not yield an improvement for list ranks of 5 or less, is that in-class ranks introduce a small overhead, i.e. an extra click to examine the class the result is in. Thus, if the desired document is ranked first, i.e. its list rank is 1, and, for class-based ranking, this document is the first document in the first class, the user must examine one class and one document, thereby incurring a cost of 2.

It is interesting to note that for an initial list rank of 5 or less, the best classification-based methods are provided by document-based ranking methods, specifically DR and QD_IR. However, for list ranks greater than 5, classification methods based on class size perform best. For initial list ranks between 5 and 10, we observe that SR and QSR are the best, and for an initial list rank greater than 10, SR performs best in most cases. This might be due to the fact that for list ranks greater than 10, the initial query is, by definition, poor, and therefore ranking classes based only on the query-independent component is usually superior. However, the difference in performance between the two methods is actually quite small.

Figure 5.1 shows the cumulative distribution of target documents for each method. We see that for list rank, approximately 25% of target documents are at list rank of 1, and 35% have a list rank less than or equal to 2. No class ranking system has a class rank of 1 because of the overhead it introduces. Approximately 25% of clicked document have a in-class rank of 2. The list rank and in-class rank cross at rank 4. Approximate 50% of the documents have a rank of 4 or less for all systems. Conversely, 50% of clicked documents have a rank greater than 4, and in those cases, a classification based system performs better.

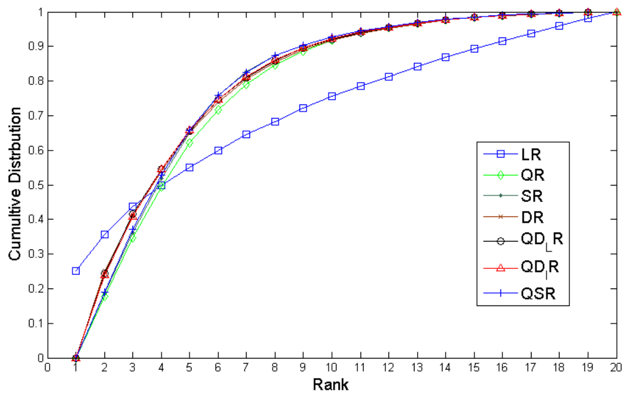


Figure 5.1: The cumulative distribution of the rank of click through based on ranking position (multiple-click).

5.3.3.2 Results for One-Click Queries

The performance for one-click queries is very similar to the multiple-click queries.

LR	QR	DR	SR	QSR	QD_LR	QD_IR
1	3.11	**2.00**	2.85	2.79	2.08	2.14
2	3.57	3.00	3.32	3.28	**2.94**	3.04
3	4.13	3.68	3.74	3.76	**3.61**	3.68
4	4.56	4.25	4.20	4.21	**4.18**	4.22
5	4.98	4.78	**4.62**	4.65	4.71	4.73
6	5.40	5.31	**5.04**	5.05	5.25	5.24
7	5.87	5.85	**5.51**	5.52	5.77	5.74
8	6.22	6.16	**5.88**	5.92	6.06	6.06
9	6.40	6.51	**6.05**	6.09	6.43	6.36
10	7.01	7.11	**6.63**	6.64	7.04	7.01
11	8.03	8.06	**7.54**	7.67	8.03	8.03
12	7.95	8.03	**7.38**	7.47	7.98	7.93
13	8.23	8.25	**7.78**	**7.78**	8.17	8.14
14	8.71	8.67	**8.28**	8.29	8.60	8.56

Continued on next page

LR	**QR**	**DR**	**SR**	**QSR**	**QD_LR**	**QD_IR**
15	9.04	9.18	**8.61**	8.67	9.13	9.10
16	9.55	9.85	9.28	**9.26**	9.72	9.57
17	10.05	10.37	9.81	**9.79**	10.31	10.17
18	9.88	10.14	9.65	**9.60**	10.09	9.91
19	10.95	11.12	**10.38**	10.60	11.05	11.01
20	10.95	11.17	**10.51**	10.58	11.14	11.03

Table 5.3: The comparison of class based rank for the last click through according to list rank (one-click).

Table 5.3 shows the mean value of the respective in-class rank for each list rank. Column 1 of Table 5.3 provides the list rank of the clicked document in the list-ranked results set. We can see that all in-class ranks perform worse than the list rank when list rank is less than or equal to 4, which is similar to the results in Table 5.2. The in-class rank outperforms the list rank when the list rank is greater than 4. In those cases, once again, ranking classes based on class size, i,e, SR and QSR, exhibit the better results.

Figure 5.2 shows the cumulative distribution of target documents for each method, for one-click queries.

Compared to the Figure 5.1, the list rank more strongly dominates the top ranks, i.e., approximate 47% of target documents are at list rank of 1 and about 70% of target documents are at ranks of three or less. As before, ranking classes based on a document-based ranking provides the best performance for the classification methods when the list rank is less than 5.

If the initial query is good, i.e. the target document has a list rank less than 5, then displaying results as a traditional one-dimensional list is superior. However, for queries where the initial list rank is 5 or more, classification based ranking offers better results. It would therefore be interesting to investigate a hybrid method for displaying the result set, in which the top-ranked document is displayed first, followed by categorised results.

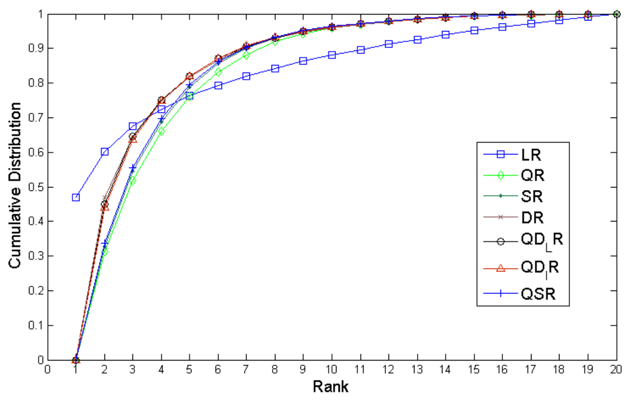


Figure 5.2: The Cumulative Distribution of Click Through Based on Ranking Position (one-click).

5.4 Learning to Rank

The ranking model in the previous section is unsupervised. Recently, learning to rank [86] has becomes very topical as it can potentially deliver state of the art performance. Here we conduct the preliminary experiments in order to learn to rank classes of search results. The algorithms we make use of are RankNet [25] and RankingSVM [73]. We note that both algorithms use a pairwise approach, i.e.,a pair of documents are represented as feature vectors and the output is the pairwise preference between each pair of documents. Such pairwise approaches suffer from some limitations. i.e., the positional information is invisible to their loss functions, and they ignore the fact that some documents (or document pairs) are associated with the same query [83], compared to the listwise approaches, such as ListNet [28]. Moreover, given a small number of features, those machine learning algorithms may not outperform conventional ranking algorithms. Our target is to evaluate the classification-based search , but to be complete, we conduct the following experiment and present the result here. In future, we plan to carry out some research by making use of larger query log data, more features and advanced algorithms on this topic.

The dataset we used was described in the previous section. From the data set, we randomly chose 5000 queries, all of which contains a single click through as the training sample; another 4000 queries were similarly chosen as the test data.

The class of the single clickthrough is taken to be the target class, and we randomly select other class to form a pair of classes. Then the learning to rank algorithm learns the preference model from these pairs. The generated model is used to rank the classes of the results of queries from the test data.

Learning to rank methods work well for high dimensional data sets, typically with hundreds of features or more. However, to be able to compare these experiments with the unsupervised one, carried out in the previous sections, we use the following features: (1) top document rank within a class, (2) mean document rank within a class, (3) class size, and (4) class probability based on query classification.

The experimental results are shown in Table 5.4.

LR	QR	SR	QSR	QD_LR	$NETR$	$SVMR$
1	2.95	2.68	2.77	**2.07**	2.27	2.17
2	3.59	3.27	3.44	**2.92**	3.11	3.04
3	3.79	**3.51**	3.59	3.61	3.61	3.62
4	4.38	**4.05**	4.20	4.12	4.25	4.23
5	4.70	**4.35**	4.48	4.69	4.56	4.68
6	5.19	**4.82**	4.96	5.23	5.09	5.19
7	5.67	**5.27**	5.40	5.72	5.57	5.65
8	6.28	**5.79**	5.94	6.30	6.11	6.12
9	6.39	**5.94**	6.20	6.45	6.22	6.26
10	6.66	**6.16**	6.35	6.68	6.55	6.58

Table 5.4: The comparison of unsupervised class ranking and supervised class ranking

In the table $NETR$ denotes the class rank generated by RankNet, and $SVMR$ denotes the class rank generated by RankingSVM. It can be seen that the results of the supervised ranking methods are similar to the unsupervised one. We can see that supervised ranking methods do not outperform the unsupervised ones.

However, for initial list rank of two or less, the corresponding class rank is close to the best class rank, and the overall performance is close the *QSR*. Although the supervised ranking methods do not prove to be advantageous, we believe that this is mainly due to the limited feature space. In future work, we plan to explore the performance of supervised ranking methods in higher dimensional feature space.

5.5 A Proof of Concept for a Classification-Based Search Engine

A proof of concept for a classification-based search is necessary for our work. We implemented a prototype of the system as described below.

5.5.1 Classification-Based Search Engines - an Architecture

To demonstrate our models for classification-based search, we require a search engine to integrate a classification module. In this section, we will describe an architecture of such a classification-based search engine.

For the search component, we made use of a metasearch engine, which combines results from one or more search engines and presents one ranked list to the user through a common interface. The primary advantage of metasearch engine over each individual search engine is improved coverage. Earlier research [78] concluded that all search engines cover only a small portion of the size of the web. Searching multiple search engines simultaneously via metasearch can improve the coverage significantly. If combined with vertical search engines, metasearch can improve the precision of individual search engine to some extent, this relates to the query classification problem in Chapter 4.

Another advantage is the metasearch does not require any crawler and indexing because metasearch provides a free platform to access the commercial search engines. One issue with metasearch engines is that they are at the mercy

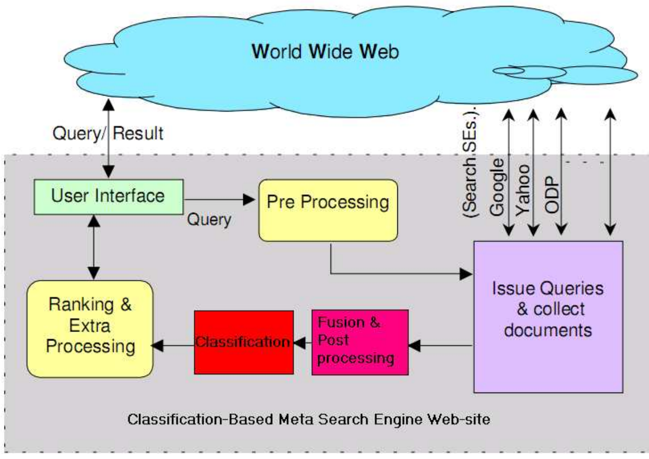


Figure 5.3: An architecture of a classification-based metasearch engine.

of the search engines they wish to query, and some have banned metasearch engines from “free riding” on top of them. This leads some metasearch engines to negotiate a commercial arrangement with the major search engines [79].

The classification-based metasearch engine is similar to a typical metasearch engine except that it consists of one extra module - the classifier, hence the ranking module involves the additional computation of ordering classes and documents within each class.

The architecture of a classification-based metasearch search is shown in Figure 5.3. The classification module involves two parts: query classification and results classification.

Apart from the low level architecture, the user interface is important for a search system. An important task of a user interface is to aid users in the expression of their information needs, in the formulation of their queries, in the understanding of their search results, and in keeping track of the progress of their information seeking efforts [65]. We should keep the user interface as simple as possible so that the user will focus on the content and not be attracted or confused by any other complex function. This is the reason the standard user interface for search engine has remained the same for the last decade.

For web applications like search, the most important functionalities are a

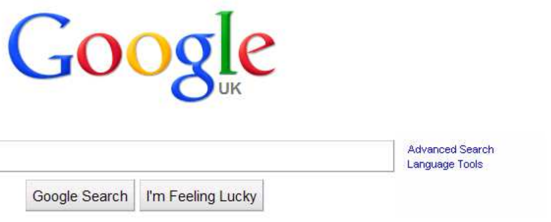
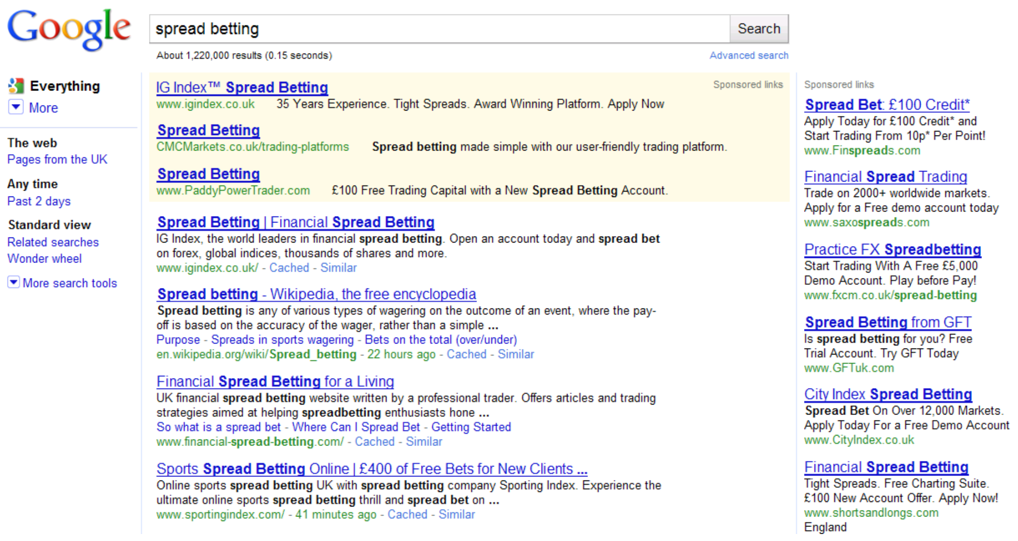


Figure 5.4: Google Search User Interface

hyperlink and a "Submit" button. In this case, data and functionality are not clearly separated and a rich tool set is not necessary for a simple search experience. One example is the Google user interface shown in Figure 5.4. When the user issues a query, Figure 5.5 shows how Google presents the results to the user. Google provides the function of restricting results by place and time



About 1,220,000 results (0.15 seconds)

Sponsored links

IG Index™ Spread Betting
www.igindex.co.uk 35 Years Experience. Tight Spreads. Award Winning Platform. Apply Now

Spread Betting
CMCMarkets.co.uk/trading-platforms Spread betting made simple with our user-friendly trading platform.

Spread Betting
www.PaddyPowerTrader.com £100 Free Trading Capital with a New Spread Betting Account.

Spread Betting | Financial Spread Betting
IG Index, the world leaders in financial spread betting. Open an account today and spread bet on forex, global indices, thousands of shares and more.
www.igindex.co.uk/ - Cached - Similar

Spread betting - Wikipedia, the free encyclopedia
Spread betting is any of various types of wagering on the outcome of an event, where the pay-off is based on the accuracy of the wager, rather than a simple ...
Purpose - Spreads in sports wagering - Bets on the total (over/under)
en.wikipedia.org/wiki/Spread_betting - 22 hours ago - Cached - Similar

Financial Spread Betting for a Living
UK financial spread betting website written by a professional trader. Offers articles and trading strategies aimed at helping spreadbetting enthusiasts hone ...
So what is a spread bet - Where Can I Spread Bet - Getting Started
www.financial-spread-betting.com/ - Cached - Similar

Sports Spread Betting Online | £400 of Free Bets for New Clients ...
Online sports spread betting UK with spread betting company Sporting Index. Experience the ultimate online sports spread betting thrill and spread bet on ...
www.sportingindex.com/ - 41 minutes ago - Cached - Similar

Sponsored links

Spread Bet: £100 Credit*
Apply Today for £100 Credit* and Start Trading From 10p* Per Point!
www.Finspreads.com

Financial Spread Trading
Trade on 2000+ worldwide markets. Apply for a Free demo account today
www.saxospreads.com

Practice FX Spreadbetting
Start Trading With A Free £5,000 Demo Account. Play before Pay!
www.fxcm.co.uk/spread-betting

Spread Betting from GFT
Is spread betting for you? Free Trial Account. Try GFT Today
www.GFTuk.com

City Index Spread Betting
Spread Bet On Over 12,000 Markets. Apply Today For a Free Demo Account
www.CityIndex.co.uk

Financial Spread Betting
Tight Spreads. Free Charting Suite. £100 New Account Offer. Apply Now!
www.shortsandlongs.com
England

Figure 5.5: Search Engine Results Pages (SERP)

on the left panel. In the middle panel is the main retrieved results for the query, which may include small number of sponsored links. The main sponsored links are listed on the right panel. This user interface is very clear and effective.

There are other tips that contribute to an effective user interface, such as Google Suggest, which has the ability to narrow in on good search terms based on partial user input. In our implementation, we ignore the query suggest

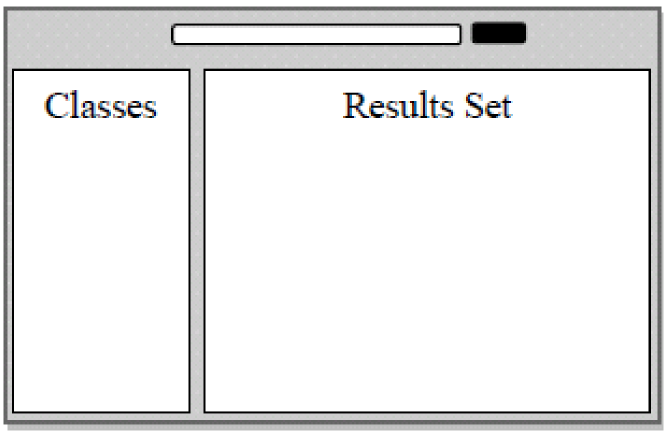


Figure 5.6: The Layout of the User Interface

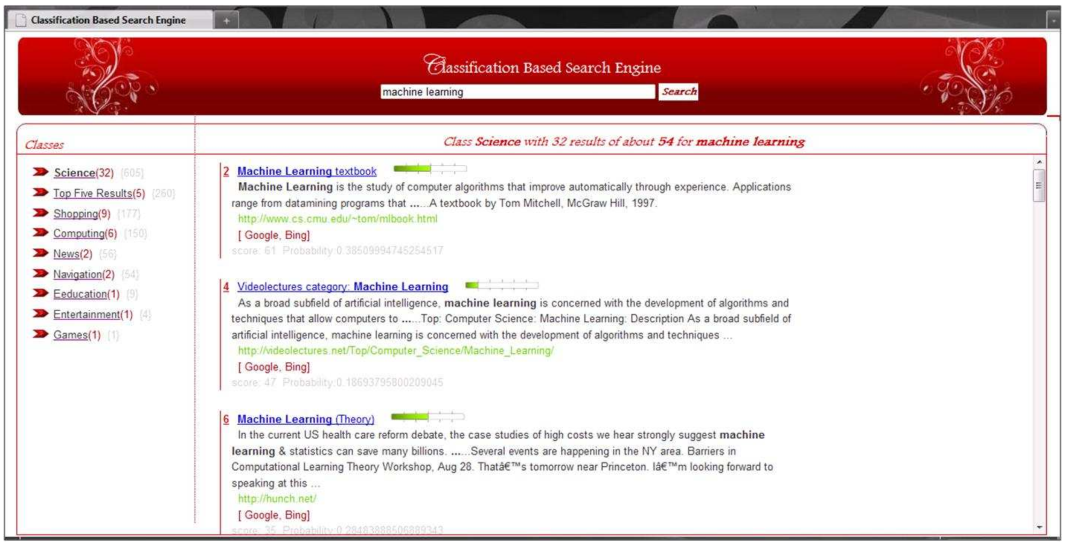
feature, but we implement logging function in the system.

5.5.2 Classification-Based Search Engines - an Implementation

In this section, our proposed search engines will be introduced. The main value here is to provide the basis functionality of classification-based search engines, and as described previously, there are many implementation issues we have not addressed, such as Google Suggest functionality and the limitation of API. The prototype system was implemented as an MSc project [4].

5.5.2.1 Proposal for User Interface

There are three components involved in a user interface: the search box, the classes and the results set for each class. We adopted the approach as in Figure 5.6. This layout lists the class labels at the left panel and the corresponding results set at the right panel. The class labels are organised in a logical way so that most relevant class will be presented first. The results set are stored in an object at Javascript level so that it is simple and convenient to display the results. This layout approach is widely adopted by many cluster search engine, including Yippy (`http://www.yippy.com/`), SnakeT (`http:`


Classification Based Search Engine

Classification Based Search Engine

machine learning Search

Classes

- Science(32) {605}
- Top Five Results(5) {260}
- Shopping(9) {177}
- Computing(6) {150}
- News(2) {56}
- Navigation(2) {54}
- Eeducation(1) {9}
- Entertainment(1) {4}
- Games(1) {1}

Class **Science** with 32 results of about **54** for **machine learning**

2 **Machine Learning** textbook
Machine Learning is the study of computer algorithms that improve automatically through experience. Applications range from datamining programs that **...**...A textbook by Tom Mitchell, McGraw Hill, 1997.
http://www.cs.cmu.edu/~tom/mlbook.html
[Google, Bing]
score: 61 Probability:0.38509994745254517

4 Videolectures category: **Machine Learning**
As a broad subfield of artificial intelligence, **machine learning** is concerned with the development of algorithms and techniques that allow computers to **...**...Top: Computer Science: Machine Learning: Description As a broad subfield of artificial intelligence, machine learning is concerned with the development of algorithms and techniques ...
http://videolectures.net/Top/Computer_Science/Machine_Learning/
[Google, Bing]
score: 47 Probability:0.18693795800209045

6 **Machine Learning** (Theory)
In the current US health care reform debate, the case studies of high costs we hear strongly suggest **machine learning** & statistics can save many billions. **...**...Several events are happening in the NY area. Barriers in Computational Learning Theory Workshop, Aug 28. Thatâ€™s tomorrow near Princeton. Iâ€™m looking forward to speaking at this ...
http://hunch.net/
[Google, Bing]
score: 35 Probability:0.28483888506889343

Figure 5.7: The Screen Shot of User Interface

//snaket.di.unipi.it/) and Carrot (http://project.carrot2.org/). A screen shot of this layout from the prototype is shown in Figure 5.7.

The project is implemented in Java and Javascript. Glassfish (http://glassfish.java.net/) is the application server containing the service.

5.5.2.2 Metasearch with an SVM Classification Module

JavaScript Object Notation (JSON) is used as a vehicle for the retrieved results, since both Google Search Ajax API and Bing API 2.0 support JSON.

The classification module is trained from two parts of the manually classified AOL Log as we mentioned in Chapter 4 and one subset of manually classified Live Data. The predictive model used bi-gram linear SVM, also described in Chapter 4.

Each result is associated with a score to reflect the relative relevance for a user query. In this project, we adopted a simple approach to rank the result called Borda Count algorithm [38]. The procedure works as followed: the web pages of each search engine are scored in such a way that if n results are retrieved, then the first document will have the score n and the second will have $n-1$ and so on until the last document will have the score 1; the scores of the corresponding results from both search engines are then added up.

We conducted a small sample user test on the prototype as part of the MSc project. The six participants have a variety of backgrounds, including English, IT, Engineering and Dentistry. Each user was issued with three queries and asked to find their desired information in our system. They were then asked to provide some feedback about the system. The following is a summarisation of the user testing.

- Users found the results they had in mind in the class with the highest score that is displayed first to the user but not necessarily in the class they expected.
- Users had to check up to a maximum two classes until they found the document they had in mind.
- The feature of classification, interface design, font size, and colour theme was rated good in general, however the speed of the system was rated negatively. Some of the users thought that the speed, although slow, was tolerable since they did not have to sequentially scan the entire results set if they had an expectation of what class a result may belong to.
- Users in general were very excited to use the search engine and they appraised the idea of classifying search engine results.

5.6 Discussion

We proposed a probabilistic model for ranking classes, and derived six ranking functions from this model in section 5.2. Two models, SR and DR, were query-independent, and one model, QR, was query-dependent. A combination of these resulted in the models QSR, and QD_IR and QD_LR.

Within each class, the rank order of documents was identical to that in the original list rank. We believe this is an important experimental control in order to be certain that any improvements in ranking are solely due to the classification methods under investigation.

In section 5.3.3 we examined a subset of queries derived from an MSN log recorded in Spring 2006. This subset consisted of 20,000 queries for which one-click was associated with each query, and 20,000 queries for which multiple-clicks were associated with each query. The two data sets were examined independently, but the experimental results are consistent across both. In particular, we observed that for target documents with an initial list rank less than 5, the classification-based methods offered no advantage. This is partly due to the fact that these methods introduce a small overhead, i.e. to even examine the first document in the first class requires two, rather than one click. However, for target documents with an initial list rank of 5 or more, classification methods are better. Of the six methods examined, the two based on class size, SR and QSR, performed best from table 5.3 and table 5.2. The difference between these two methods is small.

For the case where the target document has an initial list rank of 5 or less, the document-based classification methods performed best. However, they were inferior to traditional list rank, i.e. no classification.

The fact that the traditional list rank performs well for good queries, i.e. where the initial rank of target documents is less than 5, while classification-based methods perform well for poorer queries, i.e. where the initial rank of target documents is greater than 4, suggests that some form of hybrid method should be investigated. For example, one could display the top-ranked document followed by categorised results. This would be an interesting line of future investigation.

A key assumption of our experimental results is that the retrieved results obtained using Live Search are similar to those observed by users at the time the query log was collected in Spring 2006. It is not possible to verify this assumption, and it would be interesting to repeat our experiments on more recent data.

In our work, we also assume that classification is perfect, i.e. that documents are correctly classified and that users correctly identify the target class. In practice, this will not be the case, and our experimental results must be considered as a best case scenario.

We also implemented a prototype of classification based search engine. In a real implementation, we encountered the false positive problem, which is difficult to avoid. Encountering false positives will lower the users' confidence in the system. Two approaches to handle this issue are:

- User explicit feedback.
- Classification during indexing.

If the user can provide direct feedback to our classifier system, it will improve the performance. However, most users are reluctant to do the extra work and implicit feedback, e.g., clickthrough data can be used. Moreover, classification during indexing is a reasonable approach to reduce the false positives.

We know that the one level class structure provides a coarse granularity. Some queries can be more specific and domain focused, which requires a deep class hierarchy. This requires further research. Clustering and tag information from social bookmarking website such as delicious can be applied to this problem.

Chapter 6

Concluding Remarks and Future Directions

In this thesis, we investigated the issue of improving search engines by employing classification. It is motivated by the following considerations: 1) Information growth on the web is increasing at an exponential rate, and 2) discovering structure within an information domain has been proved to be an effective way in practice of organizing large data sets. For example, the time-consuming manual classification of information has successfully been applied by librarians for information organisation with a long history. Classification also deals with the unavoidable problem that the retrieved results from traditional search engines are topic-independent, therefore limiting users' search experience for ambiguous queries.

To address this issue, in this thesis we have proposed classification as a means to organise the retrieved results from a search engine. At the theoretical level, document classification is derived from statistical learning theory and can control the generalization ability of the learning process. In practice, advanced classifiers, such as Support Vector Machines, have obtained better accuracy on a variety of datasets compared to traditional classifiers, such as decision tree or Naive bayes classifier[72]. Such automatic classifiers form the foundation of our research.

Chapter 3 presented a simulation-based evaluation framework that can be used to measure the effect of adding a classification component to a search engine. This methodology explored six possible user behaviors in a category-based results presentation for the identification of the target document in table 3.1. By design, for every target document, ten queries, each corresponding to different levels of query quality were generated automatically so that the correlation between the effect on category-based systems and query quality could be analysed in section 3.4.1. This method therefore reduced the need for the expensive and time-consuming user testing, although in practice user testing is still very important. A summary of the contributions and results from this chapter are as follows:

- We first proposed six user models corresponding to user behaviors in the classification-based framework. For each user model, we defined related class-based ranks in section 3.2. Although precursory research has confirmed by user testing, that category-based retrieval can improve users' searching time, to our knowledge, there has been no work discussing the ranking of document within a classification-based framework, which allows us to quantitatively analyze the benefits of a category-based strategy. In this context it is worth mentioning that eye tracking experiments have shown that document ranking is correlated with search time. Our results in section 3.4.1 not only demonstrated the advantages when the user correctly identifies the class and there is no machine error, but also suggested the strategy the user should take to achieve the optimal performance when the user does not know the class or when there are user and machine errors.
- Classification will introduce errors. We first analysed the correlation between the performance of classifiers and the performance of classification-based search engines in section 3.4.1.1. Our work suggests that the performance of classification-based search engines are negatively affected by the accuracy of the classifier. However, significant reductions in search time can still be achieved when the error rates are even as high as 30%.
- We envisage that a classification-based search engine will create only a small overhead, in particular for good queries in section 3.4.1.1; here the

good queries means the rank of the desired document is within the top 5 results. We suggested a hybrid system to address this problem.

Chapter 4 introduced the topic of query classification and an application of query classification, i.e., topic specific analysis of search queries. The chapter considered the main challenge of query classification, in particular for online prediction of the class the user is interested in, in the case where the available information is usually limited to the user query. The method we employ is to enrich the query with additional keywords as we described in section 4.3. We argued that for a query classifier, the quality and size of the training data will positively affect the performance. We enriched each query with the co-occurrence terms from Yahoo's Related Suggestions, Explore Concepts from Yahoo! and the top 2,5,10 and 20 snippets of results from Google. Then different combinations of enrichment were considered for training query and test query in section 4.5. A bi-gram SVM model was built to predict the label of the query. The results suggested that:

- Enrichment by pseudo-relevance feedback, i.e., top n snippets of results from Google, was better than enrichment by suggested keywords and Explore Concepts for a symmetric setting in table 4.7.
- In the symmetric setting, the performance of enrichment from Google results increased with the number of snippets at the first stage, however degraded when enriched with the top 20 results. We believe this may be due to more noise that is introduced into the data when too many snippets are used as shown in table 4.6.
- For the asymmetrical learning setting in section 4.5.3.7, we found that enrichment based on secondary sources to be competitive with those based on pseudo-relevance feedback, which implies that if we can improve the quality of the enrichment data, the performance of query classification can also be improved.

We applied query classification into the analysis of a search query log to gain the additional insight of the users' intent in section 4.6. We found a few interesting results in section 4.6.1 such as the longest queries were for *Places* class, the largest clickthrough rate occurred at *Art* class and the largest session rates were for the *Shopping* and *Entertainment* class. For temporal characteristics, the query distribution of *Finance* and *Government* class dropped abruptly from working day to weekend, and in addition, in most cases Monday was the peak weekday. This seems to indicate that most of the queries in these categories were work related. In contrast, there was hardly any drop in the weekend for the *Game*, *Pornography* and *Entertainment* classes. These topics can be characterised as recreation, so the volume of those queries remained high during weekends.

Chapter 5 made use of the class distribution of query classification along with newly derived class features to provide a ranking for classes. Most previous work had used class size or the alphabet of the class label to rank the class. In contrast to the method we used in Chapter 3, in Chapter 5 we made use of search queries from a Microsoft Live search log. The log provides real user queries along with the clickthrough data. In section 5.3, two types of queries were studied in the experiment: one-click queries and multiple-click queries. The desired information of one-click queries is the result clicked on, while for multiple-click queries, we assumed the last clickthrough was the target document. We then analysed the retrieved results in terms of list rank and classification-based rank.

Experimental evaluation of each of our six class-based ranks methods we proposed in section 5.3.3 showed that size rank and query/size rank performed best for the queries whose rank of target document is greater than four. And for the case where the target document has an initial scroll rank of four or less, the the class-based rank making use of document-based information yielded the best result; however, it is inferior to traditional list rank due to the small overhead introduced by classification system. To address the overhead issue, we suggested a hybrid search interface: we put top ranked results first, followed by class results. This will slightly affect the performance of classification-based search engines, but it can still have a better performance for documents with low rank and the same performance as traditional search engines for documents with high ranks, which form the bulk of clickthroughs.

A prototype classification-based search engine was built to evaluate the feasibility of our idea and small scale user testing was carried out to confirm our results in section 5.5.

The work presented in this thesis examined specific strategies in the use of a classification-based search engine. However, much work still remains to be done under this topic.

The need for a reasonable depth of class structure is one that has received little attention in IR literature. Chapter 3 described a shallow class structure to organize the results along with the definition of class-based rank. Deep structure classification can convey more information, however, it may increase the search time to the target document, and thus the class-based rank of the document. Moreover, displaying a deep hierarchy is problematic, and may occupy more screen space. It is desirable in this case to learn how to effectively present the results in an efficient way, also user testing may suggest better methods for displaying the results.

The classifier used here is by no means perfect. The work described in Chapter 3 confirmed that the accuracy of the classifier will affect a classification-based search engine. In particular, false positive will have a negative effect on the system. Ideally, we expect that full document classification performed offline during indexing will improve the classifier's accuracy. In such a case, semi-supervised learning can aid the classifier. However, this approach is not feasible for a metasearch engine. We believe user feedback can be used to partially solve the problem. This requires that the user would provide feedback and the predictive model will be updated frequently based on the feedback.

For the query classification task, the effectiveness of each measure varied depending on the size and quality of enrichment data. Understanding the nature of this relation between the performance of the classifier and the data is obviously an interesting question as we know that retrieved results varied across a range of topics. Extending this idea to topic modeling [19] is an interesting direction.

Learning to rank classes has received little attention in IR literature, although learning to rank results has been popular for the last decade [86]. We conducted preliminary experiments to investigate the effect of the new features derived from class. Nevertheless, learning to rank class can improve the system when more features are considered. This will reweight the importance of each feature. However, this problem requires further exploration.

Throughout the thesis, the experiments did not involve any user testing except for a limited test of the prototype search engine. It would be interesting to conduct a large scale user test and to compare it with the known item test carried out in this thesis.

Commercial search engines present the results independently of the topics. Moreover, cluster-based search engines have to face the unavoidable problem of cluster label generation and the discrepancy between the clustering algorithm and human cognition. Identifying the benefits of the classification-based approach quantitatively and the new direction presented in this thesis is therefore important. The author hopes that this thesis proves to be useful in addressing these and related topics in future research on classification-based search.

Bibliography

[1] Lucene. http://lucene.apache.org/java/.

[2] E. Agichtein, E. Brill, and S. Dumais. Improving web search ranking by incorporating user behavior information. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 19–26, 2006.

[3] R. Agrawal, S. Gollapudi, A. Halverson, and S. Ieong. Diversifying search results. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, WSDM '09, pages 5–14, New York, NY, USA, 2009. ACM.

[4] A. Alasiry. Classification of search engine results. Master Project, 2009.

[5] E. Alpaydin. *Introduction to Machine Learning, second Edition.* MIT press, Massachusetts, USA, 2010.

[6] G. Amati, C. Carpineto, G. Romano, and F. U. Bordoni. Query difficulty, robustness and selective application of query expansion. In *Proceeding of the 26th European Conference on IR Research*, pages 127–137. Springer, 2004.

[7] P. Anick, V. Murthi, and S. Sebastian. Similar term discovery using web search. In *Proceedings of the International Language Resources and Evaluation (LREC)*, pages 1209–1213, Marrakech, Morocco, May 2008.

[8] J. Arguello, F. Diaz, J. Callan, and J.-F. Crespo. Sources of evidence for vertical selection. In *Proceedings of the 32nd international ACM SIGIR*

conference on Research and development in information retrieval, SIGIR '09, pages 315–322, New York, NY, USA, 2009. ACM.

[9] L. Azzopardi and M. D. Rijke. Automatic construction of known-item finding test beds. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 603–604.

[10] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval.* Addison Wesley, Essex, England, 1999.

[11] B. Bai, J. Weston, D. Grangier, R. Collobert, K. Sadamasa, Y. Qi, C. Cortes, and M. Mohri. Polynomial semantic indexing. In *In Advances in Neural Information Processing Systems*, 2009.

[12] J. Bar-Ilan, K. Keenoy, M. Levene, and E. Yaari. Presentation bias is significant in determining user preference for search results–a user study. *Journal of the American Society for Information Science and Technology*, 60(1):135–149, 2009.

[13] J. Bar-Ilan, K. Keenoy, E. Yaari, and M. Levene. User rankings of search engine results. *Journal of American Society for Information Science and Technology*, 58(9):1254–1266, 2007.

[14] S. M. Beitzel, E. C. Jensen, A. Chowdhury, and O. Frieder. Varying approaches to topical web query classification. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 783–784, 2007.

[15] S. M. Beitzel, E. C. Jensen, O. Frieder, D. D. Lewis, A. Chowdhury, and A. Kolcz. Improving automatic query classification via semi-supervised learning. In *Proceedings of the Fifth IEEE International Conference on Data Mining*, pages 42–49, 2005.

[16] M. W. Berry, Z. Drmac, and E. R. Jessup. Matrices, vector spaces, and information retrieval. *SIAM Review*, 41(2):335–362, 1999.

[17] D. P. Bertsekas and J. N. Tsitsiklis. *Introduction to Probability, second Edition.* Athena Scientific press, 2008.

[18] C. M. Bishop. *Pattern Recognition and Machine Learning.* Springer, New York, NY, USA, 2006.

[19] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[20] P. Boldi, F. Bonchi, C. Castillo, D. Donato, A. Gionis, and S. Vigna. The query-flow graph: model and applications. In *Proceeding of the 17th ACM conference on Information and knowledge management*, CIKM '08, pages 609–618, New York, NY, USA, 2008. ACM.

[21] J. Brank, M. Grobelnik, N. Milic-Frayling, and D. Mladenic. Interaction of feature selection methods and linear classification models. In *In Proceedings of the ICML-02 Workshop on Text Learning.* Forthcoming, 2002.

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

[23] A. Z. Broder, M. Fontoura, E. Gabrilovich, A. Joshi, V. Josifovski, and T. Zhang. Robust classification of rare queries using web knowledge. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 231–238, 2007.

[24] D. M. Bruce Croft and T. Strohman. *Search Engines: Information Retrieval in Practice.* Addison Wesley Press, Boston, MA, USA, 2009.

[25] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 89–96, 2005.

[26] H. Cao, D. H. Hu, D. Shen, D. Jiang, J.-T. Sun, E. Chen, and Q. Yang. Context-aware query classification. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 3–10, 2009.

[27] H. Cao, D. Jiang, J. Pei, Q. He, Z. Liao, E. Chen, and H. Li. Context-aware query suggestion by mining click-through and session data. In *Proceeding*

of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '08, pages 875–883, New York, NY, USA, 2008. ACM.

[28] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, ICML '07, pages 129–136, New York, NY, USA, 2007. ACM.

[29] C. Carpineto, S. Osiński, G. Romano, and D. Weiss. A survey of web clustering engines. *ACM Computing Surveys*, 41(3):1–38, 2009.

[30] S. Chakrabarti, M. van den Berg, and B. Dom. Focused crawling: a new approach to topic-specific web resource discovery. *Computer Networks*, 31(11-16):1623–1640, 1999.

[31] H. Chen and S. Dumais. Bring order to the web: Automatically categorizing search results. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 145–152, 2000.

[32] H. Chen and D. R. Karger. Less is more: probabilistic models for retrieving fewer relevant documents. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '06, pages 429–436, New York, NY, USA, 2006. ACM.

[33] Z. Chen, O. Wu, M. Zhu, and W. Hu. A novel web page filtering system by combining texts and images. In *Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 732–735, Washington, DC, USA, 2006.

[34] V. Cherkassky and F. Mulier. *Learning from Data - Concepts, Theory, and Methods.* Wiley-Interscience, Hoboken, NJ, USA, 2007.

[35] J. A. A. S. Chris Buckley, Gerard Salton. Automatic query expansion using smart : Trec 3. In *In Proceedings of The third Text REtrieval Conference (TREC-3*, pages 69–80, 1994.

[36] P. R. Christopher D. Manning and H. Sch´utze. *Introduction to Information Retrieval.* Cambridge University Press, New York, NY, USA, 2008.

[37] C. L. Clarke, M. Kolla, G. V. Cormack, O. Vechtomova, A. Ashkan, S. Büttcher, and I. MacKinnon. Novelty and diversity in information retrieval evaluation. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '08, pages 659–666, New York, NY, USA, 2008. ACM.

[38] M. J. S. Clive L. Dym, William H. Wood. Rank ordering engineering designs:pairwise comparison charts and borda counts. *Research in Engineering Design*, 13(4):236–242, 2002.

[39] W. S. Cooper. Expected search length: A single measure of retrieval effectiveness based on the weak ordering action of retrieval systems. *American Documentation*, 19(1):30–41, 1968.

[40] W. S. Cooper and M. E. Maron. Foundations of probabilistic and utility-theoretic indexing. *Journal of the ACM*, 25(1):67–80, 1978.

[41] C. Cortes and V. Vapnik. Support-vector networks. In *Machine Learning*, pages 273–297, 1995.

[42] I. J. Cox, M. L. Miller, S. M. Omohundro, and P. N. Yianilos. Target testing and the pichunter bayesian multimedia retrieval system. In *Proceedings of the 3rd International Forum on Research and Technology Advances in Digital Libraries*, page 66, 1996.

[43] S. Cronen-Townsend, Y. Zhou, and W. B. Croft. Predicting query performance. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '02, pages 299–306, New York, NY, USA, 2002. ACM.

[44] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41:391–407, 1990.

[45] O. Drori and N. Alon. Using document classification for displaying search results lists. Technical report, Leibniz Center for Research in Computer Science, Hebrew University, Jerusalem, Israel, 2002.

[46] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, New York, NY, USA, 2000.

[47] S. Dumais and H. Chen. Hierarchical classification of web content. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 256–263, 2000.

[48] G. Dupret and C. Liao. A model to estimate intrinsic document relevance from the clickthrough logs of a web search engine. In *Proceedings of the third ACM international conference on Web search and data mining*, WSDM '10, pages 181–190, New York, NY, USA, 2010. ACM.

[49] G. E. Dupret and B. Piwowarski. A user browsing model to predict search engine click data from past observations. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '08, pages 331–338, New York, NY, USA, 2008. ACM.

[50] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. Liblinear: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

[51] C. Fellbaum. *WordNet: An Electronic Lexical Database*. MIT press, 1999.

[52] P. M. Fernando and O. A. Quintana. Nonparametric bayesian data analysis. *Statistical Science*, 19:95–110, 2004.

[53] E. W. Forgy. Cluster analysis of multivariate data: efficiency vs interpretability of classifications. *Biometrics*, 21:768–780, 1965.

[54] J. H. Friedman, J. L. Bentley, and R. A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, 3(3):209–226, 1977.

[55] I. B. Gianluca Demartini, Paul-Alexandru Chirita and W. Nejdl. Ranking categories for web search. In *Proceeding of the 30th European conference on Advances in information retrieval*, pages 564–569. Springer, 2008.

[56] E. J. Glover, S. Lawrence, M. D. Gordon, W. P. Birmingham, and C. L. Giles. Web search–your way. *Communications of the ACM*, 44(12):97–102, 2001.

[57] W. Goffman. A searching procedure for information retrieval. *Information Storage and Retrieval*, 2(2):73–78, 1964.

[58] O. Gospodnetić and E. Hatcher. *Lucene in Action (2nd Edition)*. Manning Publications, Greenwich, CT, USA, 2010.

[59] D. A. Grossman and O. Frieder. *Information Retrieval: Algorithms and Heuristics*. Springer, Dordrecht, The Netherlands, 2004.

[60] A. Gulli and A. Signorini. The indexable web is more than 11.5 billion pages. In *Special interest tracks and posters of the 14th international conference on World Wide Web*, pages 902–903, 2005.

[61] Z. Gyöngyi, H. Garcia-Molina, and J. Pedersen. Combating web spam with trustrank. In *Proceedings of the Thirtieth international conference on Very large data bases*, pages 576–587. VLDB Endowment, 2004.

[62] D. Harman. Towards interactive query expansion. In *Proceedings of the 11th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '88, pages 321–331, New York, NY, USA, 1988. ACM.

[63] T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning, Second Edition*. Springer, New York, NY, USA, 2009.

[64] B. He and I. Ounis. Query performance prediction. *Information Systems*, 31:585–594, November 2006.

[65] M. A. Hearst. *Search User Interfaces*. Cambridge University Press, 2009.

[66] M. A. Hearst and J. O. Pedersen. Reexamining the cluster hypothesis: Scatter/gather on retrieval results. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 76–84.

[67] C.-J. Hsieh, K.-W. Chang, C.-J. Lin, S. S. Keerthi, and S. Sundararajan. A dual coordinate descent method for large-scale linear svm. In *Proceedings of the 25th international conference on Machine learning*, pages 408–415, 2008.

[68] B. J. Jansen. Search log analysis: What it is, what's been done and how to do it. *Library and Information Science Research*, 28:407–432, 2006.

[69] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems*, 20(4):422–446, 2002.

[70] H. Jian, W. Gang, L. Fred, S. Jian-tao, and C. Zheng. Understanding user's query intent with wikipedia. In *Proceedings of the 18th international conference on World wide web*, pages 471–480, 2009.

[71] T. Joachims. Text categorization with suport vector machines: Learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning*, pages 137–142, London, UK, 1998. Springer-Verlag.

[72] T. Joachims. *Learning to Classify Text Using Support Vector Machines: Methods, Theory and Algorithms.* Kluwer Academic, Norwell, MA, USA, 2002.

[73] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142, 2002.

[74] T. Joachims. Training linear svms in linear time. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 217–226, 2006.

[75] T. Joachims, L. Granka, B. Pan, H. Hembrooke, and G. Gay. Accurately interpreting clickthrough data as implicit feedback. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 154–161, 2005.

[76] K. Kummamuru, R. Lotlikar, S. Roy, K. Singal, and R. Krishnapuram. A hierarchical monothetic document clustering algorithm for summarization and browsing search results. In *Proceedings of the 13th International Conference on World Wide Web*, pages 658–665, New York, NY, USA, 2004. ACM Press.

[77] F. W. Lancaster. *Information Retrieval Systems: Characteristics, testing and evaluation.* John Wiley and Sons, New York, 1979.

[78] S. Lawrence and C. L. Giles. Accessibility of information on the web. *Intelligence*, 11(1):32–39, 2000.

[79] M. Levene. *An Introduction to Search Engines and Web Navigation.* Addison Wesley, Essex, England, 2005.

[80] Y. Li and Z. Zheng. Kdd cup 2005. Online at `http://www.acm.org/sigs/sigkdd/kdd2005/kddcup.html`, 2005.

[81] Y. Li, Z. Zheng, and H. K. Dai. Kdd cup-2005 report: facing a great challenge. *SIGKDD Explorations*, 7(2):91–99, 2005.

[82] C.-J. Lin, R. C. Weng, and S. S. Keerthi. Trust region newton methods for large-scale logistic regression. In *Proceedings of the 24th international conference on Machine learning*, pages 561–568, 2007.

[83] M.-Y. Liu. *Learning to Rank for Information Retrieval.* Springer, 2011.

[84] T. Liu, A. W. Moore, and A. Gray. New algorithms for efficient high-dimensional nonparametric classification. *The Journal of Machine Learning Research*, 7:1135–1158, 2006.

[85] T. Liu, A. W. Moore, A. Gray, E. Gray, and K. Yang. An investigation of practical approximate nearest neighbor algorithms. In *Neural Information Processing Systems*, pages 825–832. MIT, 2004.

[86] T. Y. Liu. *Learning to Rank for Information Retrieval.* Now Publishers Inc., 2009.

[87] X. Liu and W. B. Croft. Cluster-based retrieval using language models. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 186–193, 2004.

[88] Y. Liu, Y. Fu, M. Zhang, S. Ma, and L. Ru. Automatic search engine performance evaluation with click-through data analysis. In *Proceedings of the 16th international conference on World Wide Web*, pages 1133–1134, 2007.

[89] M. Mitra, A. Singhal, and C. Buckley. Improving automatic query expansion. In *Proceedings of the 21th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 206–214, New York, NY, USA, 1998. ACM.

[90] S. Mizzaro. How many relevances in information retrieval? *Interacting With Computers*, 10:305–322, 1998.

[91] C. J. v. R. N. Jardine. The use of hierarchic clustering in information retrieval. *Information Storage retrieval*, 7:217–240, 1963.

[92] S. Osinski and D. Weiss. Carrot 2: Design of a flexible and efficient web information retrieval framework. In *Proceedings of the third International Atlantic Web Intelligence Conference*, pages 439–444, Berlin, 2005. Springer.

[93] A. Oulasvirta, J. P. Hukkinen, and B. Schwartz. When more is less: the paradox of choice in search engine use. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '09, pages 516–523, New York, NY, USA, 2009. ACM.

[94] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report, Stanford Digital Libaray Technologies Project, 1998.

[95] B. Pang, L. Lee, and S. Vaithyanathan. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing*, pages 79–86, Morristown, NJ, USA, 2002. ACL.

[96] J. C. Platt. Fast training of support vector machines using sequential minimal optimization. *Advances in kernel methods: support vector learning*, pages 185–208, 1999.

[97] J. M. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 275–281, 1998.

[98] X. Qi and B. D. Davison. Web page classification: Features and algorithms. *ACM Computing Surveys*, 41(2):1–31, 2009.

[99] F. Radlinski, R. Kleinberg, and T. Joachims. Learning diverse rankings with multi-armed bandits. In *Proceedings of the 25th international conference on Machine learning*, ICML '08, pages 784–791, New York, NY, USA, 2008. ACM.

[100] M. Radovanović and M. Ivanović. Cats: A classification-powered meta-search engine. In *Advances in Web Intelligence and Data Mining. Studies in Computional Intelligence 23*, pages 191–200, Secaucus, NJ, USA, 2006. Springer-Verlag New York, inc.

[101] D. Rafiei, K. Bharat, and A. Shukla. Diversifying web search results. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 781–790, New York, NY, USA, 2010. ACM.

[102] S. Robertson, S. Walker, S. Jones, M. Hancock-Beaulieu, and M. Gatford. Okapi at trec-3. In *Proceedings of the Third Text REtrieval Conference (TREC 1994)*, pages 109–126, 1996.

[103] S. E. Robertson. The probability ranking principle in ir. *Readings in information retrieval*, pages 281–286, 1997.

[104] J. Rocchio. Relevance feedback in information retrieval. In *The SMART Retrieval System: Experiments in Automatic Document Processing*, pages 313–323, Upper Saddle River, NJ, 1971. Prentice-Hall Press.

[105] G. Salton. *Relevance feedback and the optimization of retrieval effectiveness. In Smart system - experiments in automatic document processing.* Prentice Hall Inc, Englewood Cliffs, NJ, 1971.

[106] G. Salton and C. Buckley. Term weighting approaches in automatic text retrieval. Technical report, Ithaca, NY, USA, 1987.

[107] G. Salton and C. Buckley. Improving retrieval performance by relevance feedback. *Journal of the American Society for Information Science*, 41:288–297, 1990.

[108] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[109] F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, 2002.

[110] D. Shen, R. Pan, J.-T. Sun, J. J. Pan, K. Wu, J. Yin, and Q. Yang. Query enrichment for web-query classification. *ACM Transactions on Information Systems*, 24(3):320–352, 2006.

[111] A. Singhal and M. Kaszkiel. A case study in web search using trec algorithms. In *Proceedings of the 10th international conference on World Wide Web*, WWW '01, pages 708–716, New York, NY, USA, 2001. ACM.

[112] A. Spink and B. J. Jansen. *Web search: Public searching of the Web.* Springer, 2004.

[113] L. T. Su. A comprehensive and systematic model of user evaluation of web search engines: 2. an evaluation by undergraduates. *Journal of American Society for Information Science and Technology*, 54(13):1193–1223, 2003.

[114] A. Sugiura and O. Etzioni. Query routing for web search engines: Architecture and experiments. In *Proceedings of the 9th international World Wide Web conference on Computer networks*, pages 417–429, 2000.

[115] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the twenty-first international conference on Machine learning*, page 104, 2004.

[116] D. Tunkelang. *Faceted Search.* Morgan and Claypool Publishers, San Rafael, USA, 2009.

[117] C. van Rijsbergen. *Information retrieval (2nd ed).* Butterworths, London, 1979.

[118] V. Vinay, I. J. Cox, N. Milic-Frayling, and K. Wood. Evaluating relevance feedback algorithms for searching on small displays. In *Proceeding of the 27th European Conference on IR Research*, pages 185–199. Springer-Verlag, 2005.

[119] C. J. M. W. Cleverdon and M. Keen. Aslib cranfield research project: Factors determining the performance of indexing systems. Cranfield Institute of Technology, Cranfield, England, 1966.

[120] X. H. Wang and C. X. Zhai. Learn from web search logs to organize search results. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 87–94, New York, NY, USA, 2007.

[121] P. Willett. Recent trends in hierarchic document clustering: a critical review. *Journal of Information Processing and Management*, 24(5):577–597, 1988.

[122] D. Xing, G.-R. Xue, Q. Yang, and Y. Yu. Deep classifier: automatically categorizing search results into large-scale hierarchies. In *Proceedings of the international conference on Web search and web data mining*, pages 139–148, 2008.

[123] H. Yang and T.-S. Chua. Effectiveness of web page classification on finding list answers. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 522–523, 2004.

[124] A. M. Yuye Zhang. Some observations on user search behavior. In *Proceedings of the 11th Australian Document Computing Symposium*, pages 1–8, 2006.

[125] O. Zamir and O. Etzioni. Grouper: A dynamic clustering interface to web search results. *Computer Networks*, pages 1361–1374, 1999.

[126] H. J. Zeng, Q. C. He, Z. Chen, W. Y. Ma, and J. W. Ma. Learning to cluster web search results. In *Proceedings of the 27th annual international*

ACM SIGIR conference on Research and development in information retrieval, pages 210–217, New York, USA, 2004.

[127] C. Zhai. Risk minimization and language modeling in text retrieval. Technical report, PhD dissertation, Carnegie Mellon University, Pittsburgh, PA, 2002.

[128] Y. Zhou and W. B. Croft. Query performance prediction in web search environments. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '07, pages 543–550, New York, NY, USA, 2007. ACM.