Query Log Mining in Search Engines

Marcelo Gabriel Mendoza Rocha

Presented to the University of Chile in fulfilment
of the thesis requirements to obtain the degree of
Ph. D. in Computer Science

Department of Computer Science - University of Chile
June 2007

Thesis Committee

Ph. D. Ricardo Baeza-Yates
Advisor, University of Chile, Yahoo! Research Latin America and Spain
Ph. D. Carlos Hurtado
Co-Advisor, University of Chile
Ph. D. Mark Levene
Member of the committee, Birkbeck College, University of London
Ph. D. Gonzalo Navarro
Member of the committee, University of Chile
Ph. D. Miguel Nussbaum
Member of the committee, Catholic University of Chile

Abstract

The Web is a huge read-write information space where many items such as documents, images or other multimedia can be accessed. In this context, several information technologies have been developed to help users to satisfy their searching needs on the Web, and the most used are search engines. Search engines allow users to find Web resources formulating queries (a set of terms) and reviewing a list of answers.

One of the most challenging goals for the Web community is to design search engines that allow users to find resources semantically connected to their queries. The huge size of the Web and the vagueness of the most commonly used terms to formulate queries still poses a huge problem to achieves this goal.

In this thesis we propose to explore the user's clicks registered in the search engine logs in order to learn how users search and also in order to design algorithms that could improve the precision of the answers suggested to users. We start by exploring the properties of the user's click data. This exploration allows us to determine the sparse nature of the data providing users behavior models that help us to understand how users search in search engines.

Secondly, we will explore the user's click data to find useful associations among queries registered in the logs. We will focus the efforts on the design of techniques that will allow users to find better queries than the original query. As an application, we will design query reformulation methods that will help users to find more useful terms in order to represent their needs.

On using document terms we will build vectorial representations for queries. By applying clustering techniques we are able to compute clusters of similar queries. Using query clusters, we provide techniques for document and query suggestions which allows us to improve the precision of the answer lists.

Finally we will design query classification techniques that allow us to find concepts semantically related with the original query. In order to do this, we classify the users's queries into a Web directory. As an application, we provide methods for the maintenance of the directory.

Acknowledgements

First of all, I would like to thank the Center of Web Research, specifically the Director of the Center, Ricardo Baeza-Yates, who was also my thesis advisor, and the Director of Yahoo! Research Latin America and Spain. Thanks for your assistance with the conferences and during this work. I thank the members of the committee Mark Levene, Miguel Nussbaum, Gonzalo Navarro and Carlos Hurtado for their comments during the review process. I also thank the members of the Computer Science Department of the University of Chile that were involved in my graduate studies: Claudio Gutierrez, Sergio Ochoa and Jose Pino, among others. I especially thank Carlos Hurtado, for the time he dedicated to this thesis and for his infinite patience. Sincerely, thank you Carlos.

I would like to thank Angelica, Magaly, Francia and Magna for their help. Of course, I am extremely grateful for the friendship of Rodrigo Paredes, Diego Arroyuelo, Felipe Aguilera, Carlos Acosta, Renzo Angles, Hugo Neyem and Gilberto Gutierrez, my partners in the last section of the project, and for Patricio Galdames, Jose Canuman, Cesar Collazos, Luis Guajardo and Claudio Gutierrez S., my partners at the beginning of this adventure.

I would also like to thank my colleagues at the University of Valparaiso Computer Science Department (DECOM!) for their continued support.

Finally I thank my family for their patience. I owe you. Thanks.

Publications related to this thesis

- **[BYHM04a]** R. Baeza-Yates, C. Hurtado and M. Mendoza. Query clustering for boosting web page ranking. In *AWIC 2004, Mexico, May 16-19, volume 3034 of Lecture Notes in Artificial Intelligence*, pages 164-175, Springer, 2004.
- **[BYHM04b]** R. Baeza-Yates, C. Hurtado and M. Mendoza. Query recommendation using query logs in search engines. In *EDBT 2004, Heraklion, Greece, March 14-18, volume 3268 of Lecture Notes in Computer Science*, pages 164-175, Springer, 2004.
- **[BYHM07]** R. Baeza-Yates, C. Hurtado and M. Mendoza. Improving search engines by query clustering. In *Journal of the American Society for Information Systems and Technology*, volume 6, June, 2007.
- **[BYHMD05]** R. Baeza-Yates, C. Hurtado, M. Mendoza and G. Dupret. Modeling user search behavior. In *Proceedings of the 3rd Latin American Web Conference, Buenos Aires, Argentina, October 2005*, pages 242-251, IEEE Computer Society Press, 2005.
- **[CHM06]** A. Cid, C. Hurtado and M. Mendoza. Automatic maintenance of web directories using click-through data. In *Proceedings of the 2nd International Workshop on Challenges in Web Information Retrieval and Integration, Atlanta, USA, April 2006*, IEEE Computer Society Press, 2006.
- **[DM05]** G. Dupret and M. Mendoza. Recommending better queries from click-through data. In *SPIRE 2005, Buenos Aires, Argentina, volume 3772 of Lecture Notes in Computer Science*, pages 41-44, Springer, 2005.
- **[DM06]** G. Dupret and M. Mendoza. Automatic query recommendation using click-through data. In *Symposium on Professional Practice in Artificial Intelligence, 19th IFIP World Computer Congress, Santiago, Chile, August 2006*, pages 303-312, Springer, 2004.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

At the end of the nineties, Bharat and Broder [BB98] estimated the size of the Web to be around 200 million static pages. Recently, Gulli and Signorini [GS05] pointed out that the number of indexable documents in the Web exceeds 11.5 billion. The large size of the Web has impulsed the continuous development of a class of systems that help the users search for documents on the Web. These systems, known as search engines (Google [Goo], Yahoo [Yah], MSN Search [Sea] and Ask Jeeves [Jee], among others), give the users a simple interface by means of which to enter a set of terms known as queries.

Search engines return a list of documents sorted by a relevance measure with regard to a query, known as *answer lists* [BYRN99]. The answer lists can comprise millions of documents due to the size of the Web. This fact implies that many of them may be marginally relevant or may contain poor quality material. The answer lists are usually generated using a *ranking function* that measures the relevance of the documents to the query submitted.

Of course, in answer lists not all the answers are relevant to the users. In fact, only a fraction of the retrieved documents is relevant. This fraction is called the *precision* of the answer lists [BYRN99], usually taken in a fixed number of initial results. Henzinger *et al.* [DH99] points out that one of the most important challenges in Web research is to improve the precision of the answer lists. At the present time, the main search engines agree on

the type of data that they use to compute their answer lists, which mainly involves the *link structure of the Web* and the *document terms*. However, the use of these data sources introduces several problems.

The first ranking functions based on the Vectorial model of Information Retrieval [BYRN99] were based on document terms. However, these methods presented several problems that affected the precision of the answer lists, among them the following:

1. Synonomy: many documents or queries relevant to a topic might not share the same words [Cha03].

2. Polysemic queries: queries with the same descriptive text may have different meanings in the same or in different languages [Cha03].

3. *Spamdexing*[1]: it is common for documents to include artificial repetitions of words, which allow them to appear higher in the answer lists [Nat98].

4. Quality: the text itself does not always reflects the quality of the document [BY04].

In order to improve the precision of the answer lists, search engines have incorporated the Web link structure in ranking functions. It is worth mentioning the link-based algorithms PageRank [BP98] and HITS [Kle99], which improve the performance over the algorithms based on terms. This is mainly due to the fact that the Web link structure helps to identify hubs and authorities in specific topics. However, these algorithms also present problems that affect the precision of the answer lists, such as the following:

1. Visibility: HITS as well as PageRank assume that the content of a document is more relevant as if it is referenced by a larger quantity of relevant documents, a characteristic called *visibility*. However, a visible document does not necessarily have more valuable content than other documents with fewer references pointing to them. For example, new documents cannot be very visible even when their content is high quality.

[1]The term *spamdexing* is a portmanteau of spamming and indexing.

2. Subjective referencing: the links among Web documents are placed by users who publish the documents. These users form a particular group within the universe of users of the Web. Due to the fact that the users who create Web resources can be considered as a group with some degree of expertise, their references are usually biased. Researches on evaluation methods of traditional information retrieval models conclude that the judgements of a group of expert users are usually partial [LS68, FS91].

3. Link spamming: It is common for Web sites to get people giving links to them. This is most easily accomplished using web resources where you can post any kind of content, such as on wikis, in blog comments, and in guest books. Thus, these kind of links do not represent the quality of the site but improve its ranking [GGM05].

Fortunately, the text and the link structure of the Web are not the only relevant data sources for this task. It is also possible to incorporate the user's click data.

Users express their choices by selecting some of the ranked answers suggested by the search engine. This *implicit feedback* from the users to the search engine could allow to determine high-quality document not identifiable by the document terms and link-based methods.

At this point, it is worth asking the following question. Are the user's clicks useful to identify high-quality documents in the Web? The appropiate use of the user's feedback should improve the precision of the answer lists because of the following reasons:

1. Semantics: the users select the documents that have a higher relevance because they know the meaning of their queries [PP02].

2. Quality: the users select those documents which they identify as better quality resources for their queries [ZD02].

3. Freshness: the users select the documents that have better meaning at the moment that they submit their queries [PP02].

4. Non-subjective evaluation: the users form an heterogeneous group, therefore their preferences are not partial [RS67, LS68, FS91].

When the user's feedback is incorporated into the calculation of answer lists, we may describe, as a process, the relationships among the actors involved. Following [BYRN99] this process is known as the *user feedback cycle*. In this process users, queries, documents and search engines are involved. When the user feedback cycle is used to identify relevant resources that other methods cannot identify, we talk about *user relevance feedback*. The final goal of the methods that work with user feedback is to calculate relevance measures that could be useful to determine quality resources.

Two kinds of user feedback methods could be considered at this point. Traditional methods use *explicit user feedback*, such as the specification of keywords or the selection of documents in query time. These kinds of methods force users to do additional tasks beyond their normal searching activities. As a consequence, the effectiveness of these methods can be limited.

In this thesis we study another kind of user feedback methods: *implicit user feedback*. These kinds of techniques obtain information about user needs by watching their natural interactions with search engines. The main advantage of these methods is that such techniques do not require an additional effort from the user and are not obtrusive.

The fundamental question about which observable user behaviors can be used as source of implicit measures of relevance is addressed in some works. For example, Calypool *et al.* [CLWB01] studied user behaviors in order to use them as implicit measures of preferences. Several behaviors were examined: selections, scrolling and time on page, among others. To test the correlation between user preferences and user behaviors, the authors performed an experiment based on user opinions. Users were asked to rate pages regarding their quality to a set of given queries. The authors found that time spent on a page, the amount of scrolling on a page and the combination of time, scrolling and selections had a strong correlation with the explicit ratings. Other works have stated similar conclusions [KT03, MS94].

Now we define the user feedback cycle as a process. The user feedback cycle could be described as follows: users formulate queries in order to express their needs; queries are submitted to search engines; search engines suggest relevant documents – which are recommended to users as answer lists – to every submitted query; users select documents associated to their queries. Relationships among the actors of the user feedback cycle are shown in Figure 1.1.

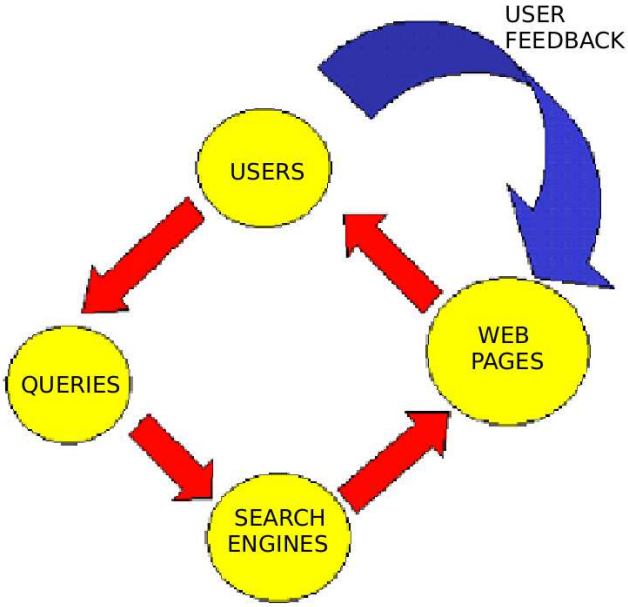


Figure 1.1: The user feedback cycle.

Fortunately, the user feedback cycle is registered by the servers that allocate the search engines. The register is kept in files known as ”system logs”. System log files register all the user’s requests in the search engine, that is to say, submitted queries, selected documents and clicks among answers. There are different formats to store logs, the ”common log format” being the most used. The attributes of every request registered in the common log format are the IP number, date and time of the request, result for the request (with code), transaction size, protocol, request description, referrer (referred resource), browser and operating system employed by the user. Every line of a file represents a user request, as shown in Figure 1.2.

Now, we will introduce some basic definitions that we will use in this thesis. We will define a *query instance* as a single query submitted to a search engine in a defined point of time, followed by zero or more document selections. Following the definition introduced by Silverstein *et al.* [SMHM99], a *query session* consists of a sequence of query instances by a single user made within a small range of time that join with the documents selected in

```
host.analog.cx - - [02/Jan/2000:12:11:12 +0000] "GET /sample2.html HTTP/1.0" 200 1010 "http://google.com/search?q=sample%20search" "Mozilla/4.0 [en] (X11; I; Linux)"
host.analog.cx - - [02/Jan/2000:14:11:12 +0000] "GET /sample2.html HTTP/1.0" 200 1010 "http://google.com/search?q=sample%20search" "Mozilla/4.0 [en] (X11; I; Linux)"
statslab.cam.ac.uk - - [02/Jan/2000:15:11:12 +0000] "GET /sample2.html HTTP/1.0" 200 1010 "http://google.com/search?q=sample%20search" "Mozilla/4.0 [en] (X11; I; Linux)"
statslab.cam.ac.uk - - [02/Jan/2000:17:11:12 +0000] "GET /sample2.html HTTP/1.0" 200 1010 "http://google.com/search?q=sample%20search" "Mozilla/4.0 [en] (X11; I; Linux)"
statslab.cam.ac.uk - - [02/Jan/2000:18:11:12 +0000] "GET /sample2.html HTTP/1.0" 200 1010 "http://google.com/search?q=sample%20search" "Mozilla/4.0 [en] (X11; I; Linux)"
statslab.cam.ac.uk - - [02/Jan/2000:20:11:12 +0000] "GET /sample2.html HTTP/1.0" 200 1010 "http://google.com/search?q=sample%20search" "Mozilla/4.0 [en] (X11; I; Linux)"
statslab.cam.ac.uk - - [03/Jan/2000:10:11:12 +0000] "GET /dir/sample3.html HTTP/1.0" 200 987 "http://google.com/search?q=another%20search" "Mozilla/4.0 [en] (X11; I; Linux)"
statslab.cam.ac.uk - - [03/Jan/2000:10:11:12 +0000] "GET /images/sample3.jpg HTTP/1.0" 200 298 "http://www.statslab.cam.ac.uk/dir/sample3.html" "Mozilla/4.0 [en] (X11; I; Linux)"
statslab.cam.ac.uk - - [03/Jan/2000:12:11:12 +0000] "GET /dir/sample3.html HTTP/1.0" 200 987 "http://google.com/search?q=another%20search" "Mozilla/4.0 [en] (X11; I; Linux)"
statslab.cam.ac.uk - - [03/Jan/2000:10:11:12 +0000] "GET /images/sample3.jpg HTTP/1.0" 200 298 "http://www.statslab.cam.ac.uk/dir/sample3.html" "Mozilla/4.0 [en] (X11; I; Linux)"
statslab.cam.ac.uk - - [04/Jan/2000:11:11:12 +0000] "GET /notfound.html HTTP/1.0" 404 0 "http://badreferrer.com/" "Mozilla/4.0 [en] (X11; I; Linux)"
host.analog.cx - - [04/Jan/2000:11:21:12 +0000] "GET /failed.html HTTP/1.0" 404 0 "http://failedreferrer.com/" "Mozilla/4.0 [en] (X11; I; Linux)"
statslab.cam.ac.uk - - [04/Jan/2000:12:01:12 +0000] "GET /redirected.html HTTP/1.0" 301 0 "http://redirectedref.com/" "Mozilla/4.0 [en] (X11; I; Linux)"
host.analog.cx - - [04/Jan/2000:12:11:12 +0000] "GET /redir.html HTTP/1.0" 301 0 "http://redirref.com/" "Mozilla/4.0 [en] (X11; I; Linux)"
statslab.cam.ac.uk - - [04/Jan/2000:13:11:12 +0000] "GET /sample.html HTTP/1.0" 200 1234 "http://google.com/search?q=sample%20search" "Mozilla/4.0 [en] (X11; I; Linux)"
```

ip | date | time | request | protocol | status | trans. size | referrer | browser | O.S.

Figure 1.2: Log file sample in common log format.

this range of time. As an additional constraint to this definition, we exclude all the query sessions without document selections from the query session set. In what follows, we will refer to these kind of query sessions as *empty query sessions*. The set of requests which belong to non-empty query sessions compound a dataset called *query log data*.

1.2 The query log mining process

In order to observe a query session, it is necessary to pre-process the server log file that identifys the user's request and that belongs to a query session. To do this, it is necessary to identify IP numbers, browsers, operative systems and query instances to associate a given request to a query session.

When query session data is built, it is possible to explore several relationships between queries and documents. We call this process *query log mining*. The query log mining process is defined as the search of non trivial patterns in the query log data. Of course it is defined as a data mining process but it focuses on the exploration of patterns in the query log data.

We will propose a query log mining process in order to identify useful patterns in the query log data that allow us to improve the quality of the answer lists given by a search engine. In order to explore the data we can use standard data mining techniques such as association rules, clustering and classification.

Now, we will propose a query log mining process for this thesis. First of all, we will focus on identify associations among queries. Associations allow us to identify generalization/specialization relationships among queries, or similarity relations among them. When relationships among queries are identified, it will be possible to propose reformulation schemes in order to expand the original query or to propose alternative queries that are more related to a specific concept.

Second, we will work on the construction of vectorial representations of query sessions using the terms that compound queries and documents. Several term weight schemes can be used at this point based on the patterns discovered in the query log mining process. By defining distance functions between queries and standard data mining techniques, such as clustering, it will be possible also to identify groups of similar queries. Finally, by using the query clusters it will be possible to identify recommendable queries or documents to the users who formulate queries that belong to any cluster.

Frequently, search engines provide Web directories to help users in their searches. Web directories are hierarchical structures of nodes organized as trees. The hierarchy represents a taxonomy and the relationships among nodes represent generalization/specialization relationships among the concepts behind the structure. Each node of the Web directory shows a list of documents related to a concept.

Following the query log mining process, we will build vectorial representations for the nodes in the directories using the terms of the documents that compound each document list. By defining distance functions between queries, documents and directory nodes, it will be possible to classify queries and documents in a Web directory. We will also show that the classification of queries into a Web directory is helpful to maintain the structure. The main activities in the proposed query log mining process are described in figure 1.3.

In light of the above, the query log mining process will be based on the following ideas:

1. Query specialization and/or generalization. This will allow us to refine the initial query, guiding the user toward a search with more precise results.

2. Determination of similar query groups. It will allow us to group queries having a

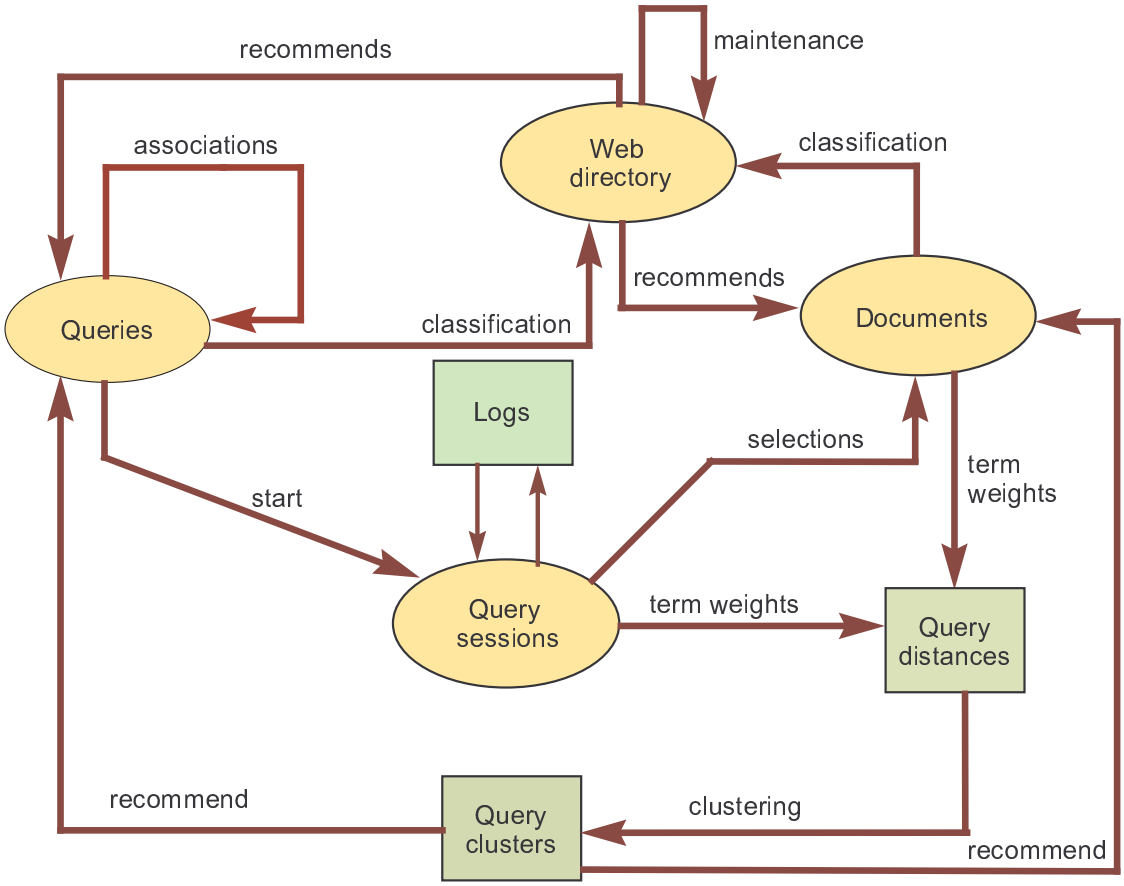


Figure 1.3: The proposed query log mining process

large number of preferences, with similar queries having a small number of preferences. Then, queries and documents may be recommended according to the popularity of each group.

3. Query classification into directories. Given a query, the nearest node in a given directory can be determined. Each directory node may show documents and/or queries which are relevant to the topic of the node. Finally, it will also be possible to specialize or generalize the query by navigating the directory.

At this point, it is relevant to specify which kind of applications will be considered as output of the query log mining process. We will work on three kind of applications:

1. Query recommendation: focusing on the reformulation of the original query, these kind of applications aim at identifying relationships between the original queries and alternative queries, such as generalization/specialization relationships.

2. Document recommendation: these kind of applications will identify relevant documents to the original query.

3. Query classification: these kind of applications will identify relevant queries and documents for each node in the directory, enriching their descriptions.

There is a similar goal behind each application: to identify relevant documents to the users. This is the main objective of a search engine and finally all the search engine applications are oriented to acomplish this goal. But they work in different manners:

1. A query recommendation application searches for a better way to formulate the original query.

2. Document recommendation applications focus on the identification of documents that are relevant to the users who have formulated similar queries in the past.

3. A query classification application searches for a better description of a node in a taxonomy, adding new queries and documents to them. When the original query is classified into a directory node, a document list is recommended to the user. Also it is possible to navigate into the directory, specializing /generalizing the original query.

A global view of the relationships between data mining techniques used in this thesis and the applications generated from the use of these techniques over the query log data is given in Figure 1.4.

Of course, the applications will follow a set of design criteria. The desirable properties of the applications will be the following:

Maintainability : The costs associated to the system maintenance must be low.

Variability : The user's preferences change in the course of time. This user click feature is due to two main factors: changes in the user's interests and elaboration of new

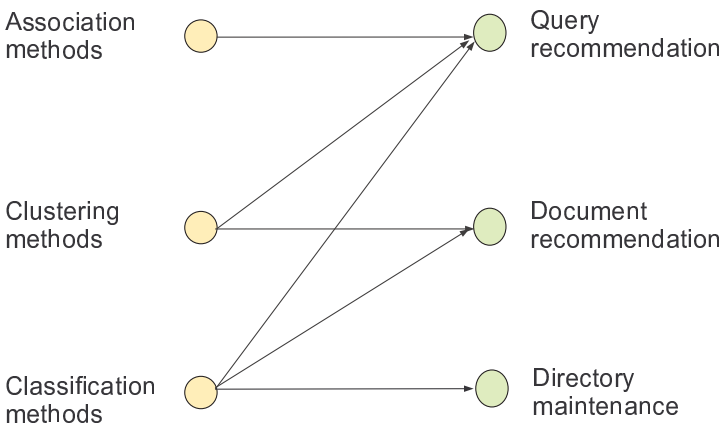


Figure 1.4: Relations between the data mining techniques used in this thesis and the applications generated

documents with contents that can be interesting to the users. The applications must reflect the changes in the user's interests in order to produce quality answer lists.

Scalability : The applications designed must be able to manage huge document collections, efficiently.

1.3 Contributions

At this point, it is necessary to state why this thesis is important and what the specific contributions would be. This thesis can give hints concerning the appropriate use of a data source which could be useful for the recommendation of quality documents. The appropiate use of the user's click data should be able to improve the quality of the answer lists. The study of the user's click data properties could also have significant implications on Web Information Retrieval. It will also state new facts in this area.

The main contribution of this thesis is scientific. We want to discuss novel ideas about the appropiate use of the user's click data in search engines. The focus will be on the use of data mining techniques that will allow us to discover useful patterns in the user's click data. We address the problem using several data mining techniques from a comprehensive

point of view, relating results obtained from different approaches in a coherent corpus of knowledge.

We will explain the specific contribution of each chapter. Each chapter is defined with regard to the exploration of the user's click data using a specific data mining technique. We will use four data mining approaches: descriptive analysis and modeling, associations searching, clustering and classification. Each one is considered as an iteration over the query log mining process. We have outlined each iteration as follows:

User's click data analysis: In this chapter, we will perform an analysis of the data. Also, the collection of queries will be characterized considering the vocabulary. In addition, we will study distributions of user's click data over several variables involved in the user feedback cycle. Finally, we will provide models of user search behavior. This chapter does not work over specific applications. The deployment of this phase is the set of models. The results of this chapter were presented in the Third Latin American Web Conference [BYHMD05], held in Buenos Aires, Argentina in 2005. The proceedings were published by IEEE CS Press.

Associations searching: In this chapter, we will search associations between queries. Interesting associations will allow us to define query specialization or generalization, providing alternative queries in order to formulate an original query. We will focus also on the identification of when a query is better than others. The models will be represented as graphs of queries, where nodes represent queries and arcs represent relations between queries. Applications will be focused on the design of query recommendation algorithms. Main results of this chapter were presented in the 12th String Processing and Information Retrieval Conference [DM05], held in Buenos Aires, Argentina in 2005. The proceedings were published by Springer-Verlag in the Lecture Notes in Computer Science Series, volume 3772. An extended version of this work was presented in the 19th IFIP World Computer Congress [DM06], held in Santiago de Chile in 2006. The proceedings were published by Springer in the IFIP series.

Clustering: This chapter will be focused on the identification of groups of similar queries. The similarity notion is defined using query session data. In this chapter, we will compare query vectorial representations based on different information sources, determining the most suitable representation for clustering. The quality of the clusters will also

be determined. We will provide query and document recommendation algorithms as applications. Main results of this chapter were presented in the Extended Database Technology Conference [BYHM04b] held in Heraklion, Crete, Greece in 2004. The proceedings were published by Springer in the Lecture Notes in Computer Science series, volume 3268 and indexed by ISI. Results related to ranked answer list production were presented in the 2nd Atlantic Web Intelligence Conference [BYHM04a] held in Cancun, Mexico, in 2004. The proceedings were published by Springer in the Lecture Notes in Artificial Intelligence series, volume 3034 and indexed by ISI. Recently an updated version of this work was accepted by the Journal of the American Society for Information Systems and Technology [BYHM07] (a journal indexed by ISI) and will appear in the 56th volume of the journal, in June 2007.

Classification: This chapter will be focused on the classification of the queries into Web directories in order to identify topics behind queries, adding new terms to the original description of the user need. To do this, we will use the document list of the node as a description of the nodes of a Web directory. By using similarity functions among queries and nodes, it will be possible to classify queries into the directory. Applications will be focused on the automatic maintenance of Web directories, identifying relations among documents, queries, and nodes of the directory, enriching their description. Main results of this chapter were presented in the 2nd International Workshop on Challenges in Web Information Retrieval [CHM06], held in Atlanta, USA, in 2006, in conjuction with ICDE 2006. The proceedings were published by IEEE CS Press.

It is worth mentioning that this thesis is viable since we have query log data, which has been provided by the Development Group of the TodoCL search engine [Tod] and by the Center for Web Research of the *Universidad de Chile* [CIW].

1.4 Thesis Outline

This thesis is organized into 7 chapters. The first chapter is an introduction to the thesis, presenting the research motivation, the proposed query log mining process, the thesis contributions, the thesis activities and the thesis organization. Chapter 2 presents the state-of-the-art, introducing definitions and reviewing topics related to this thesis. The remainder

of this thesis is focused on the query log mining process. Chapter 3 is focused on a descriptive analysis of the query log data presenting user behavior models. Chapter 4 is focused on the discovering of patterns using association searches techniques. Chapter 5 is focused on the use of clustering methods to identify groups of similar queries. In chapter 6 we study query and document classification methods. In chapter 4, 5, and 6 we provide applications based on the results of the query log mining process, thus evaluating the quality of the recommendations made by the proposed applications. Finally, in chapter 7, we give conclusions.

Chapter 2

State of the Art

The use of user feedback has been extensively explored in information retrieval. Typically, user feedback is used to make operations over the queries, reformulating them. For example, Bartell *et al.* [BCB94] propose a criterion function, based on user feedback, which determines a relation of partial nature on the documents. The criterion function that they define reaches an optimum value when the order created by the ranking function fits with the order defined by the users. The authors show that this criterion function is highly correlated with the average precision of the system.

Piwowarski [Piw00] builds a probabilistic model of information retrieval which incorporates the effect of the user feedback in the representation of the documents. For this, each document is represented by a table that indicates two values for each word: the number of queries containing the word for which the document has been selected and the number of queries containing the word for which the document has not been selected. The similarity feature intends to minimize the expected number of documents that a user has to check before finding all the relevant documents for a given query. In order to evaluate his model, the author uses two data collections. On each data collection, he compares coverage and precision between the proposed probabilistic model (denoted by DIFFn) and the *tf-idf* model. Experimental results show that when user feedback is not used, the performance of the DIFFn model is similar to the performance of the tf-idf model and, on the other hand, the bigger the feedback effect is, the better the DIFFn model performance is.

As the works described above, traditional relevance feedback methods require explicit

feedback from the users, for example, selecting documents or answering questions about their interests. This kind of feedback shows good results which improve the precision of the answer lists but also force the users to do additional activities. As the benefits are not always apparent for the users, the effectiveness of explicit techniques is limited to the cooperation of the users.

In this thesis we explore techniques based on implicit feedback. As was defined in the first chapter, the user's click data register the queries and the selections of the users. This kind of information describes the natural behavior of the users with search engine results. As a consequence, the methods based on implicit feedback are not limited to the cooperation of the users. On the other hand, it is well known that implicit measures have the following drawback: they are less accurate than explicit feedback measures thus it is neccesary to consider large quantities of user's click data to have similar results.

Some of the most studied user behaviors as sources of implicit feedback are *reading time*, *scrolling* and *selecting*. There are several works regarding the utility of these behaviors to infer implicit feedback measures. In [KT03], Kelly and Teevan propose a categorization of the papers related to implicit feedback crossing two variables: *behavior category* and *minimun scope*. The behavior category is related to the purpose of the observed behavior, for example: examine, retain, reference, annotate and create. The second dimension, scope, refers to the smallest scope of the item being acted upon, for example: segment, object or class. In the context of this thesis, the implicit feedback interaction registered in the user's click data is categorized regarding minimum scope in *segment* or *object* categories and regarding behavior category in the *examine* category. The other interactions are not registered in the user's click data so they are out of the scope of this thesis.

A relevant paper on implicit feedback measures using user's click data is given by Claypool *et al.* [CLWB01]. The authors provide a categorization of different user behaviors indicating which of them are useful as implicit feedback measures. Several behaviors were examinated: mouse clicks, scrolling and reading time, among others. The authors found that scrolling, reading time and the combination of both had a strong correlation with explicit feedback measures. However, the isolated use of mouse clicks or scrolling was found to have a weak correlation with explicit feedback measures. This result encourages the use of combinations of measures based on user's click data instead of the isolated use of a

simple measure.

As was pointed out in the works described above, implicit feedback could be useful to identify quality resources that other methods couldn't detect. As was defined in the first chapter, the aim of this thesis is to set a framework using this information source to effectively infer implicit feedback measures. The remainder of this chapter will organize the related work following the four approaches used in this thesis to study the user's click data. First, in section 2.1, we review related work on the *analysis* of user's click data, focused on the construction of user behavior models. In section 2.2 we discuss the state of the art on query associations searching methods. Section 2.3 reviews the main results on query clustering methods. In section 2.4, we review relevant results on query classification methods that focus on the use of Web directories to do this task. Finally, we give conclusions for this chapter in section 2.5.

2.1 User's click data analysis

The analysis of the user's click data is known as Web usage mining. Web usage mining literature focuses on techniques that could predict user behavior patterns in order to improve the success of Web sites in general, modifying for example, their interfaces. Some studies are also focused on applying soft computing techniques to discover interesting patterns in user's click data [Lal98, CP03], working with vagueness and imprecision in the data. Other kind of works are focused on the identification of hidden relations in the data. Typical problems related are user sessions and robot sessions identification [CMS99, XZC+02].

Web usage mining studies could be classified into two categories: those based on the server side and those based on the client side. Studies based on the client side retrieve data from the user using cookies or other methods, such as ad-hoc logging browser plugins. For example, Otsuka *et al.* [OTHK04] proposes mining techniques to process data retrieved from the client side using panel logs. Panel logs, such as TV audience ratings, cover a broad URL space from the client side. As a main contribution, they could study global behavior in Web communities.

Web usage mining studies based on server log data present statistical analysis in order to discover rules and non trivial patterns in the data. The main problem is the discovery

of navigation patterns. For example, Chen *et al.* [CLL+02] assumes that users choose the following document determined by the last few documents visited, concluding how well Markov models predict user's clicks. Similar studies consider longer sequences of requests to predict user behavior [PSC+02] or the use of hidden Markov models [YH03] introducing complex models of user click prediction. In [DK01] Deshpande and Karypis propose to select different parts of different order Markov models to reduce the state complexity and improve the prediction precision.

In [SMHM99], Silverstein *et al.* presents an analysis of the Alta Vista search engine log, focused on individual queries, duplicate queries and the correlation between query terms. As a main result, authors show that users type short queries and select very few documents in their sessions. Similar studies analyze other commercial search engine logs, such as Excite [XS00] and Ask Jeeves [SG01]. These works address the analysis from an aggregated point of view, i.e., present global distributions over the data. Other approaches include novel points of view for the analysis of the data. For example, Beitzel *et al.* [BJC+04] introduces hourly analysis. As a main result, authors show that query traffic differ from one topic to another considering hourly analysis, these results being useful for query disambiguation and routing. Finally, Jansen and Spink [JS05] address the analysis using geographic information about users. The main results of this study show that queries are short, search topics are broadening and approximately 50% of the Web documents viewed are topically relevant.

The literature also shows some works related with the following purpose: to determine the need behind the query. To do that, Broder [Bro02] analyzed a query log data determining three types of needs: informational, navigational and transactional. Informational queries are related to specific contents in documents. Navigational queries are used to find a specific site in order to browse their pages. Transactional queries allow users to perform a procedure using web resources such as commercial operations or downloads. In [RL04] the last two categories were refined into ten more classes. In recent years, a few papers have tried to determine the intention behind the query, following the categorization introduced above. In [ZHC+04, LLC05], simple attributes were used to predict the need behind the query. In [BYCG06], Baeza-Yates *et al.* introduce a framework to identify the intention behind the query using user's click data. Following a combination between supervised and

unsupervised methods, the authors show that it is possible to identify concepts related to the query and to classify the query in a given user's needs categorization.

As we can see in the works described above, the user's click data has been extensively studied in scientific literature. Despite this fact, we have little understanding about how search engines affect their own searching process and how users interact with search engine results. Our contribution in this topic will be focused in this manner: to illustrate the user-search engine interaction as a bidirectional feedback relation. To do this we should understand the searching process as a complex phenomenon constituted by multiple factors such as the relative position of the pages in answer lists, the semantic connections between query words and the document reading time, among other variables. Finally, we propose user search behavior models that involve the factors enumerated. The study closest in coverage to ours is by Silverstein *et al.* [SMHM99]. The main advantage over this work is the use of several variables involved in the searching process to produce models about how users search and how users use the search engine results.

In table 2.2 we summarize the main results in Web Usage Mining described in this section.

Reference	Focus	Application
[Lal98, CP03]	Soft Computing	User profile construction
[CMS99] [XZC+02]	Soft Computing	Robot Session identification
[OTHK04]	Client side Processing	Global behavior study of Web communities
[CLL+02]	Markov Models	Discovery of Navigation patters
{[PSC+02], [YH03], [DK01]}	Hidden Markov Models	Discovery of Navigation patters
{[SMHM99], [XS00], [SG01]}	Descriptive Analysis	Search Engine Log Characterization
[BJC+04]	Hourly Analysis	User search behavior model
[JS05]	Geographic Analysis	User search behavior model
{[ZHC+04], [LLC05, BYCG06]}	Query Categorization	User need identification

Table 2.1: Summary about Related Work in Web Usage Mining

2.2 Query association methods

It is well known that most of the queries formulated to search engines are vague and short [SMHM99]. As a consequence, search engine results are not relevant to users. To face this problem, a class of methods has been proposed in the literature. These methods allow us to reformulate the initial query adding new terms to the original query and reweighting the terms. This approach is known as *query reformulation*.

A well-known class of query reformulation strategies is based on the use of explicit relevance feedback [BYRN99]. The main idea is the following: the search engine presents to a user a list of documents and after examining them, the user selects those that are relevant to this query. Selecting descriptive terms or phrases attached to each selected document, the original query is expanded reweighting the terms that are more relevant to the selected documents. These kinds of methods show good performance in improving the precision of the retrieved documents but have one main drawback: explicit feedback needs an additional effort from the user and many of them are reluctanct to do this.

Recently, other technique have faced this problem. Implicit feedback techniques based on user's clicks have shown that it is possible to identify useful associations among queries in order to reformulate the original query. For example, in [BSWZ03], large query logs are used to build a surrogate for each document consisting of the queries that were a close match to that document. It is found that the queries that match a document are a fair description of its content. They also investigate whether query associations can play a role in query expansion. In this sense, in [SW02], a somewhat similar approach of summarizing documents with queries is proposed: Query association is based on the notion that a query that is highly similar to a document is a good descriptor of that document.

One of the main differences between the methods that we want to explore and the works described above is the following: our methods are focused on the identification of semantic relations among queries more than in the improvement of the answer lists. From our point of view, when users are given more choices in order to refine their queries, it is more probable that they can finally find useful documents. Moreover, we will focus our work on the identification of asymmetric relations among queries in the sense of hyper/hiponymy relationships. We will propose the relation *better than* trying to identify better expressions

of the original queries. We understand a better expression as an unambiguous formulation of the query. The identification of alternate queries will help it in this sense.

Following this idea, the closest work in spirit to our proposal is based on association rules. Fonseca et al. [FGDMZ03] presents a method to discover associations among queries representing them as traditional items in the association rules context. The method gives good results, however two problems arise. First, it is difficult to determine sessions of successive queries that belong to the same search process. On the other hand, the most interesting related queries, those submitted by different users, cannot be discovered. This is because the support of a rule increases only if its queries appear in the same query session and thus they are constrained to the set of queries submitted by the same user. The method we propose in this thesis aims at discovering alternate queries that could be useful in this situation because we don't need to work with this constraint.

2.3 Query clustering methods

Query clusters could be useful to identify alternate queries to an original one. The literature shows also that query clusters are useful to identify concepts behind queries and relationships among them. A seminal paper in this area was written by Beeferman and Berger [BB00]. They proposed clustering similar queries and documents modeling user preference data as a bipartite graph and applying to it an agglomerative clustering technique. They applied the technique using the real user's click data in the *Lycos* search engine [LYC] to recommend queries. Some drawbacks of their approach are the limitations of the method for large log files due to the complexity of the clustering technique. Another limitation is the orthogonality with methods based on contents being sensible to the sparseness feature of the data.

Wen *et al.* [WNZ02, WNZ01] proposed 4 notions of query distance to cluster similar queries: (1) notions based on keywords or phrases of the query, (2) notions based on string matching in the query term space, (3) notions based on cross-references between queries and documents and (4) notions based on combination among notions (1), (2) and (3). They applied the proposed techniques over real user preference data in the Encarta Encyclopedia [ENC] for Frequently Asked Questions (FAQ's) identification. Experimental results show

that it is recommendable to use combinations of the proposed distance notions but the paper does not provide a clear framework to integrate them. Another drawback is the noise effect due to the absence of a query disambiguation method.

Zaiane and Strilets [ZS02] present a method to recommend queries based on seven different notions of query similarity. Three of them are mild variations of notions (1) and (3). The remainder notions consider the content and title of the URL's in the result of a query. Their approach is intended for a meta-search engine and thus none of their similarity measures consider user's clicks in the form of clicks stored in query logs.

A hierarchical clustering approach is proposed by Chuang and Chien [CC02]. They propose to cluster queries into query taxonomies where each branch represents a well defined query topic and each query is represented by a term vector. Each component of the vector is calculated using a standard `tf-idf` schema, where the collection term for each query is retrieved from the top-N documents selected. To calculate the query taxonomy they propose to use a hierarchical agglomerative clustering technique (HAC) combined with a partitioning technique to generate a multi-way-tree cluster hierarchy. They proposed to apply the technique to FAQ identification and query filtering. Experiments show auspicious results. However, the utility of a hierarchical approach is still unclear. For example, hierarchies that have been created are too vast to navigate and it is not easy to classify new queries because the taxonomy is static and user needs are dynamic. Finally, each node represents isolated queries and due to the sparseness feature of the user's click data, it is difficult to define recommendation methods.

Xue *et al.* [XZC+04] propose to combine clicks and co-citation methods in an integrated clustering framework. They model user's click data as a bipartite graph in the same sense proposed by Beeferman and Berger [BB00]. Relations among queries and documents are calculated by the number of user preferences and added to the documents metadata. An iterative clustering algorithm groups similar queries and documents. Finally, the method recommends documents combining similarity based on document content and document metadata. The work has two main drawbacks: the method does not provide a clear integration framework between both data sources and is limited to the performance of the iterative clustering algorithm, which introduces high computational costs.

Sahami and Heilman [SH06] propose to measure the similarity between queries using

text snippets[1]. They treat each snippet as a query, formulating it to a web search engine and finding the set of documents that contain the terms in the original snippet. The retrieved documents are used to create a context vector for the original snippet. The similarity between text snippets is determined using the cosine distance calculated over the set of context vectors. The main advantage of the method is that it is capable to determine semantic relationships among queries which do not share common terms. Unfortunately the paper lacks extensive experimentation in order to assess the quality of the suggested queries. Also, the method is biased towards the search engine ranking function.

Zhang and Dong [ZD02] propose a document recommendation algorithm based on query clustering. Given a query q, the system determines the user group U who have submitted the query recently. Then, the cluster of queries Q submitted by the group of users U is retrieved. Later, the group of selected documents by the group of user U associated to the cluster Q are determined. The ranking functions of the query are calculated on this cluster of documents, considering contents criteria. The proposed algorithm is applied to an images search engine on the Web. Unfortunately, the authors do not compare the performance of the proposed models with other ones.

In this thesis we present a new framework for clustering queries. The clustering process is based on a term-weight vector representation of queries, obtained by aggregating the term-weight vectors of the selected URLs for the query. The vectorial representation leads to a similarity measure in which the degree of similarity of two queries is given by the fraction of common terms in the URLs clicked in the answers of the queries. This notion allows us to capture semantic connections between queries having different query terms.

Because the vectorial representation of a query is obtained by aggregating term-weight vectors of documents, our framework avoids the problems of comparing and clustering sparse collection of vectors, a problem that appears in the works described above. The central idea in our vectorial representation of queries, is to allow manipulating and processing queries in the same fashion as documents are handled in traditional IR systems, therefore allowing a fully symmetric treatment of queries and documents. In this form, query clustering turns into a problem similar to document clustering.

We present two applications of our clustering framework: query recommendation and

[1] Snippets are excerpts that describe the content of a document and its relation with a given query.

answer ranking. The use of query clustering for query recommendation has been suggested by Beeferman and Berger [BB00], however as far as we know there is no formal study of the problem. Regarding answer ranking, we are not aware of formal research on the application of query clustering to this problem. For both applications, we provide a criterion to rank the suggested URLs and queries that combine the *similarity* of the query (resp. URL) to the input query and the *support* of the query (resp., URL) in the cluster. The support measures how much the recommended query (resp. URL) has attracted the attention of users. The rank estimates the *interest* of the query (resp., URL) to the user that submitted the input query. It is important to include a measure of *support* for the recommended queries (resp., URL's), because queries (URL's) that are useful to many users are worth being recommended in our context.

Table 2.2 shows a summary about related work in query clustering.

Reference	Criterion	Applications
[BB00]	Agglomerative Clustering Technique over a Bipartite Graph	Query Recommendation
[WNZ01, WNZ02]	Four notions of distance: Keywords or phrases of the query String matching in the query-space Co-citation Combinations of previous notions	FAQs identification
[ZS02]	Seven notions of distance: mild variations over notions introduced in [WNZ02]	Query recommendation
[CC02]	Hierarchical Agglomerative Clustering (HAC) for Query Taxonomies Construction	FAQs identification Query filtering Web Usage Mining
[ZD02]	Query clustering infered from user groups	Document Recommendation Image Recommendation
[XZC+04]	Combination of clicks and co-citation over a Bipartite Graph	Document Recommendation
[SH06]	Query Similarity function based on Web snippets	Query Recommendation

Table 2.2: Summary about Related Work in Query Clustering

2.4 Query classification methods

Web directories are subject taxonomies where each category is described by a set of terms that consign the concept behind the category and a list of documents related to the concept. To classify queries into directories could be useful to discover relationships among queries and the concepts behind each query. In this sense there has been substantial work. For example, in [Cha03] Chakrabarti proposes a Bayesian approach to classify queries into subject taxonomies. Based on the construction of training data, the queries are classified following a breadth first search strategy starting from the root category and descending one level on each iteration. The main limitation of the method is the following: queries are always classified into leaves because leaves maximize the probability of the path *root - leaf*.

In the *KDDCUP'05*, the proposed competition was focused on the classification of queries into a given subject taxonomy. To do that, the competition organizers provided a small training data set composed by a list of queries and their category labels. Most of the papers were based on classifiers which learn under supervised techniques. The winning paper was written by Shen *et al.* [SPS+05] and applies a two phase framework to classify a set of queries into a subject taxonomy. Using a machine learning approach, they collected data from the web for training synonym based classifiers that map a query to each related category. In the second phase, the queries were formulated to a search engine. Using the labels and the text of the retrieved pages, the queries were enriched in their descriptions. Finally, the queries were classified into the subject taxonomy using the classifiers through a consensus function. The main limitations of the proposed method are the dependency of the classification to the quality of the training data, the human effort involved in the training data construction and the semi-automatic nature of the approach which limits the scale of the method applications.

Vogel *et al.* [VBH+05] classify queries into subject taxonomies using a semi-automatic approach. First they post the query to the Google directory [Goo] which scans the *Open Directory* [DMO] for occurrences of the query within the Open Directory categories. Then the top-100 documents are retrieved from Google formulating the query to the search engine. Using this document collection an ordered list of those categories that the 100 retrieved

documents are classified into is built. Using a semi-automatic category mapping between the web categories and a subject category the method identifies a set of the closest topics to each query. Unfortunately, the method is limited to the quality of the Google classifier that identifies the closest categories in the Open directory. Also, it is limited to the quality of the answer lists retrieved from Google. Finally, the semi-automatic nature of the approach limits the scale of the method.

In this thesis, we explore the idea of processing the user's click data to classify queries into a Web directory. To do this, we model a query session as an instance of a single query followed by a list of clicks to URLs in the answer of the query. Then, we can build a vectorial representation of a query session using a similar approach as in the query clustering chapter. Using a simple nearest neighborhood classifier based on distances, we classify queries into categories. Finally, using relationships between queries and documents we can also classify documents into a category using the utility of the document to describe the concept behind the query.

Our work shows several advantages compared with the related work described above. First, it doesn't need to use training data to classify queries into categories. Thus, it is not limited to the quality of the training data and is not biased towards the user experts that built the training data. Second, it's an automatic method. Most of the methods described above are semi-automatic, so in some step of the procedure it is neccesary to have human supervision. Finally, it provides a framework to automatically maintain the web directory. By determining the utility of a document to a category, it is possible to enrich the description of the whole directory.

2.5 Conclusions

Relevance feedback is an extensively studied area of information retrieval. From the point of view of this thesis, the closest works are those which are based on implicit feedback. There are several works related to the four approaches used in this thesis. Despite this fact, there are not conclusive results on how to use user's click data to infer implicit feedback measures. It is also not clear also from the literature how to use this data and how to use implicit feedback to improve the performance of search engines.

This thesis presents the main contributions in the following direction: to provide a clear framework to work with user's click data. As a conclusion we can assure that this thesis, would actually constitute a contribution to the state of the art.

Chapter 3

User's Click Data Analysis

In this chapter we will characterize the user's click data performing the data analysis phase of this thesis. The aim is to show distributions of the data identifying user search patterns. We formalize this showing models about how users search and how users use search engine results.

A relevant portion of this chapter has been published in [BYHMD05].

3.1 User's click data pre-processing

A log file is a list of all the requests formulated to a search engine in a period of time. Search engine requests include query formulations, document selections and navigation clicks. For the subject of this chapter, we only retrieve from the log files query formulations and document selection requests. In the remainder of this chapter we will call this *query log data*, as was defined in the previous section.

Using query log data, we identify query sessions using a query session detection algorithm. Firstly, we identify user sessions using IP and browser data retrieved from each request. Then, a query session detection algorithm determine when a query instance starts a new query session considering the time gap between a document selection and the following query instance. As a threshold value, we consider 15 minutes. Thus, if a user make a query 15 minutes after the last click, he starts a new query session.

We organized the query log data in a relational schema. Main relations on the data

model are *query session*, *click*, *query*, *url*, *popuq* (popularity), *queryterm* and *keyword*. Several relations has been formulated in order to answer specific queries to this thesis, but they are basically views over the previous relations, thus they are not included in the data model. Figure 3.1 shows the data model of the query log database.

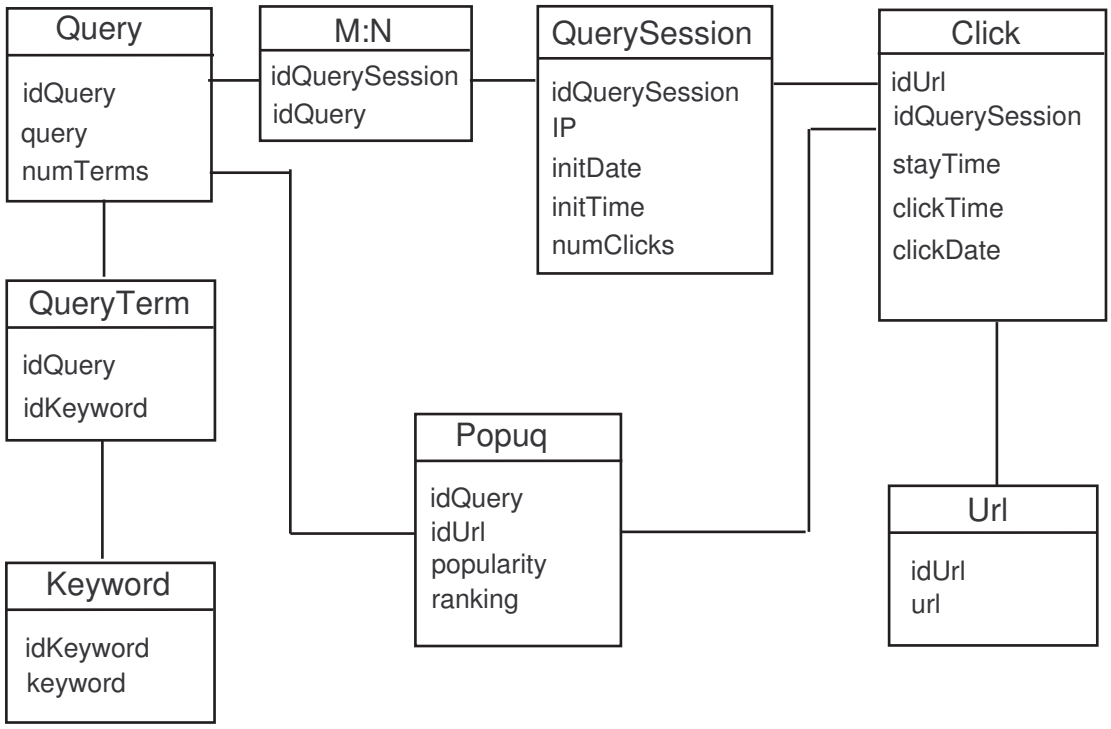


Figure 3.1: Data model used in the query log database.

We processed and stored the query log database in an Athlon-XP at 2.26 GHz, with 1GB of RAM memory and 80 GB of hard disk space, using *MySQL 5.0* as a database engine.

In order to work with query and document collections in our analysis, we need to introduce a text pre-processing step. The main goal of this process is to transform the collections from a full text expresion to a set of index terms. The process is compound by three operations: first, the elimination of accents, hyphens, digits and punctuation marks. Second, the case of letters. Finally, the elimination of stopwords.

In document or query full texts, there are several characters or words that are not related to their meaning. For example, accents, hyphens, digits and punctuation marks in general are usually not good index terms because they are vague. So, tipically this words or characters are removed from the collection. There are some special cases where hyphens or digits contain meaning. For example, the term *mp3* refers to a specific audio file format but removing the digit it has no sense and the retrieval of relevant documents for this example will be difficult. For these cases, a list of exceptions is prepared in order to avoid the elimination of relevant index terms which contains hyphens or digits. Frequently, the case of letters is not important for the ellaboration of index term sets. In our study we convert all the text to lower case.

Document and query collections have words that are very frequent. As a consequence they are not good discriminators. Such words are known as *stopwords* and are normally eliminated from the collections. Tipically examples of stopwords are articles, prepositions and conjunctions, among others. To remove stopwords from our collections we use a list of 670 stopwords that includes frequent verbs, articles and conjuntions, among others words.

Figure 3.2 shows the text pre-processing phases used in this thesis.

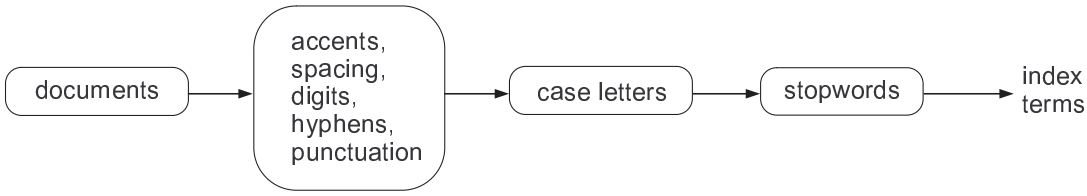


Figure 3.2: The phases of the text pre-processing used in this thesis.

3.2 DataSets

We work over the data generated by a Chilean search engine called TodoCL [Tod]. TodoCL mainly covers the .CL domain and some pages included in the .COM and .NET domain that are hosted by Chilean ISP providers. TodoCL collects over three million Chilean Web pages, and has more than 50,000 requests per day being the most important Chilean owned

	L_1	L_3	L_6
Distinct Queries	6.042	65.282	127.642
Distinct selected URLs	18.527	122.184	238.457
Sessions	10.363	102.865	245.170

Table 3.1: Characteristics of L_1, L_3 and L_6.

Successful requests	4,920,463
Average successful requests per day	53,483
Successful request for pages	1,898,981
Average successful requests for pages per day	20,641
Redirected requests	380,922
Data transferred	66.96 gigabytes
Average data transferred per day	745.29 megabytes

Table 3.2: Statistics that summarize the contents of L_3.

search engine.

In this thesis we work over three periods of logs of approximately 15 days, 3 and 6 months. We denote the 15 days log by L_1, the three months log by L_3 and the six months log by L_6. In table 3.1 we show some characteristics of the logs.

The log file denoted by L_1 was extracted from a 15-day query-log. The log contains 6,042 queries having clicks in their answers. There are 22,190 selections registered in the log, and these selections are over 18,527 different URL's. Thus in average users clicked 3.67 URL's per query.

The log file L_3 gathered 20,563,567 requests, most of them with no selections: Meta search engines issue queries and re-use the answer of `TodoCL` but do not return information on user selections.

The log file denoted by L_6 was collected from a 6 months log file. The 6-month log file contains 127,642 queries over 245,170 sessions. There are 617,796 selections registered in the log and these selections are over 238,457 different URL's. Thus in average users clicked 4.84 URL's per query.

We use in this chapter the query log file denoted by L_3 collected from 20 April 2004 to 20 July 2004. The table 3.2 summarize some log file statistics.

3.3 Sessions and Queries

Following previous definitions, L_3 register 1,524,843 query instances, 1,480,098 query sessions, 102,865 non empty query sessions and 65,282 different queries with at least 1 related click. Also, the logs register 232,613 clicks over 122,184 different documents. The average number of queries for all sessions is 1,037 and 1,435 if we count only non empty sessions. Figure 3.3(A) shows the number of new queries registered in the logs.

Another relevant feature of the data is the occurrence distribution of the queries in the log. Figure 3.3(B) shows the log plot of the number of queries per number of occurrences. Figure 3.3(B) shows that over the 80% of the queries are formulated only once. They are similar to most frequent queries in other search engines.

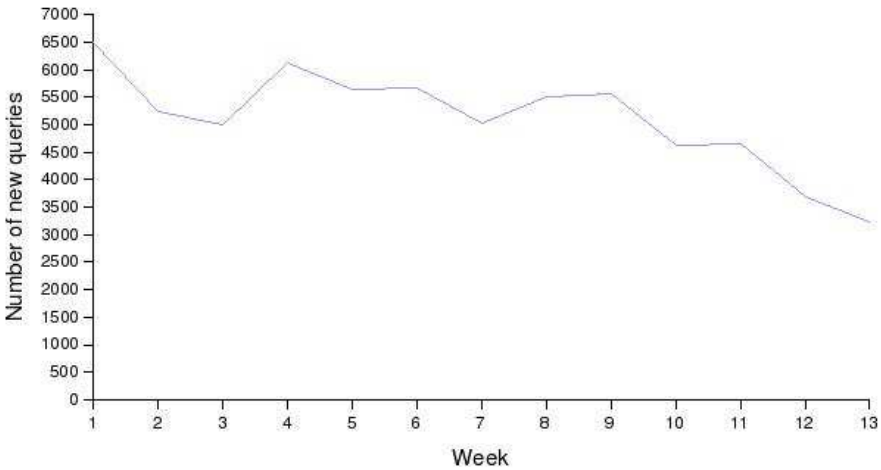
Finally, in table 3.3(A) are shown the most common queries.

3.4 Keyword Analysis

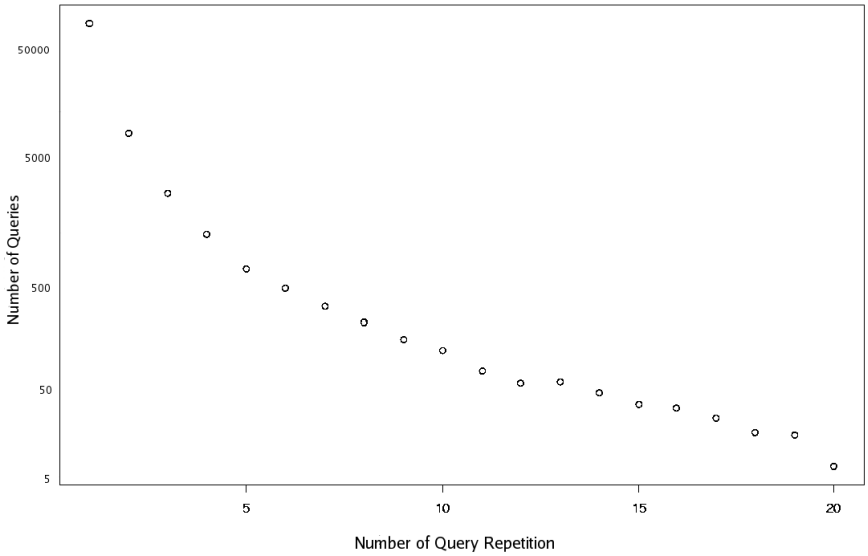
An important issue related to this chapter is to determine some properties over the keyword collection. For example the top query terms used to make queries.

The table 3.3(B) summarize number of occurrences and normalized frequencies for the top 10 query terms in the log file. As we can see in the table, some keywords are related to universal topics such as sales, photos or rents. Other keywords are related to domain topics (.CL) such as *Chile*, *Santiago* or *Chilean*, and are equivalent to the 40% of the keyword occurrences over the query log. As shown in table 3.3(A), the most frequent queries do not share some popular keywords. Thus keywords as *Chile*, *Santiago* or *Chilean* appear in many queries, but these queries are not so popular individually. This means that people are trying to get geographicaly aware results, such as "*cars in Santiago*" or "*rent an apartment in Santiago*".

On the other hand, some specific keywords that individually are not so popular (thus, they are not included in the previous table) are popular as queries (for example, chat, Google and Yahoo queries). Finally, some keywords appear in both tables such as *cars* and *house*. These kind of keywords are good descriptors of common need information clusters, i.e., they can be used to pose similar queries (for example, *used car, rent car, old*



(A)



(B)

Figure 3.3: (A) Number of new queries registered in the logs. (B) Log plot of number of queries v/s number of occurrences.

car and *luxury car*) and, at the same time, have an important meaning as individual queries.

In the query term space, keywords appearing only once represent around 60% of the queries. It is important to see that the query term data follows a Zipf distribution as the

query	occurrences
google	682
sex	600
hotmail	479
emule	342
chat	324
free sex	270
cars	261
yahoo	259
games	235
kazaa	232

(A)

term	occurrences	frequency
chile	63460	0,34
sale	23657	0,13
cars	10192	0,05
santiago	9139	0,05
radios	7004	0,04
used	6994	0,04
house	6073	0,03
photos	5935	0,03
rent	5352	0,03
Chilean	5113	0,03

(B)

Table 3.3: (A) The top 10 queries. (B) List of the top 10 query terms sorted by the number of query occurrences in the log file. Queries were written originally in Spanish.

log-plot of the data in figure 3.4(A) shows. Let X be the number of occurrences of query terms in the whole query log database. Then the number of queries expected for a known number of occurrences in the log is given by:

$$f(X = x) = \frac{1}{x^b}, \quad b > 0 \,. \tag{3.1}$$

Fitting a straight line to the main body of log-plot of the data we can estimate the value of the parameter b that is equal to the slope of the line. Our data shows that this value is 1.079.

It is important to see the data distribution of the terms that not belong to the intersection. The collection formed by text terms that not belong to the query vocabulary is shown in the log plot of figure 3.4(B). On the other hand, the collection formed by query terms that not belong to the text vocabulary is shown in the log plot of figure 3.4(C). Both collections show a Zipf distribution, with b values 1.231 and 1.643, respectively. For sake of completeness we include in 3.4(D) the distribution of all the terms in the text, which has $b = 1.408$.

Finally, another relevant subject of this chapter is to show the correlation between query terms and document terms. The query vocabulary has 27,766 terms and the text vocabulary

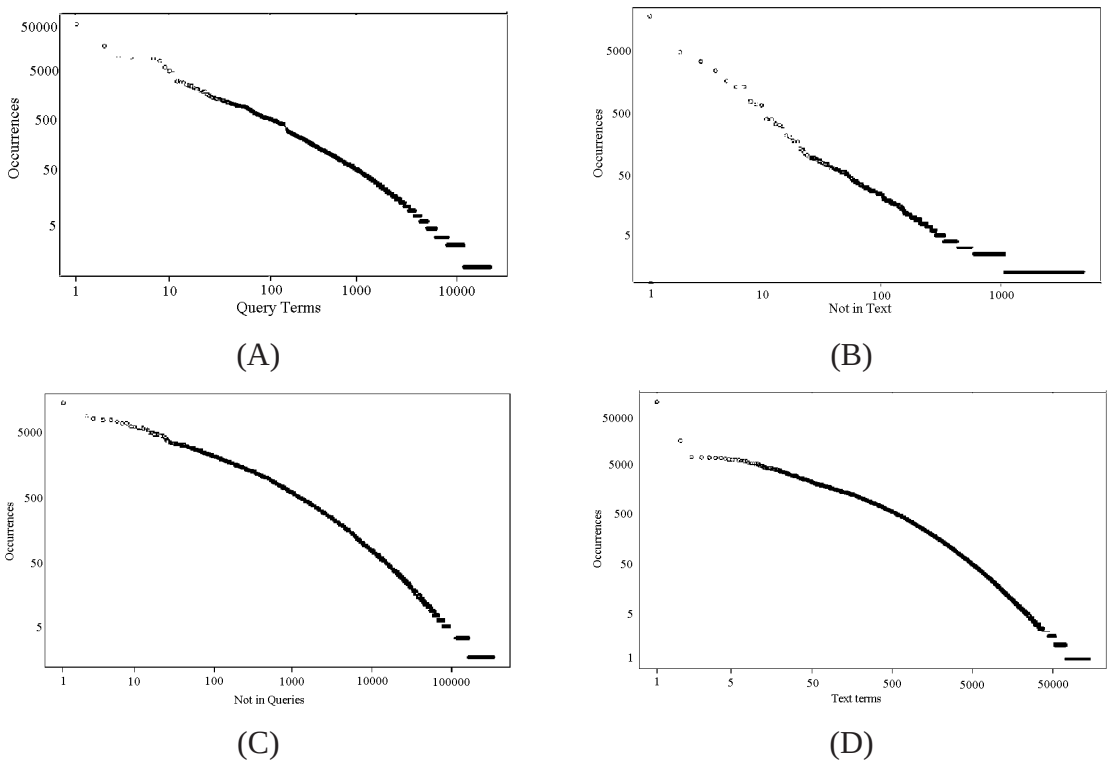


Figure 3.4: (A) Log-plot of the query term collection. (B) Log-plot of the query terms that do not appear in the text collection. (C) Log-plot of the text terms that do not appear in the query collection. (D) Log-plot of the overall text term collection.

has 359,056 terms. Common terms between both collections are only 22,692. The figure 3.5 shows an scattering plot of the query term and the text term collection.

The plot was generated over the intersection of both collections and comparing the relative frequencies of each term calculated over each collection. As the plot shows, the Pearson correlation between both collections is 0.625.

3.5 Click Analysis

An important information source that can be useful to describe user behavior in relation with their queries and their searches is the click data. Click data could be useful in order to determine popular documents associated to particular queries. Also could be useful to

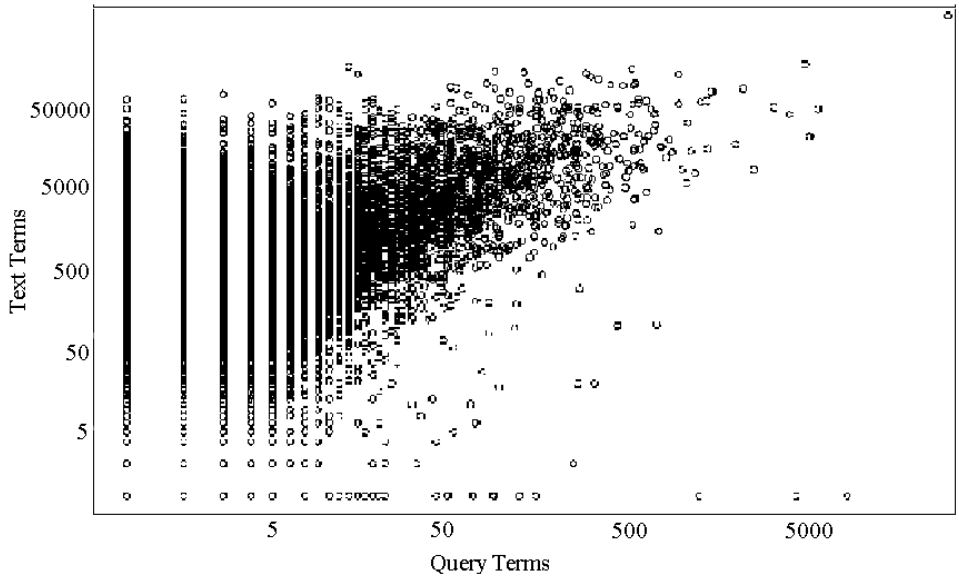


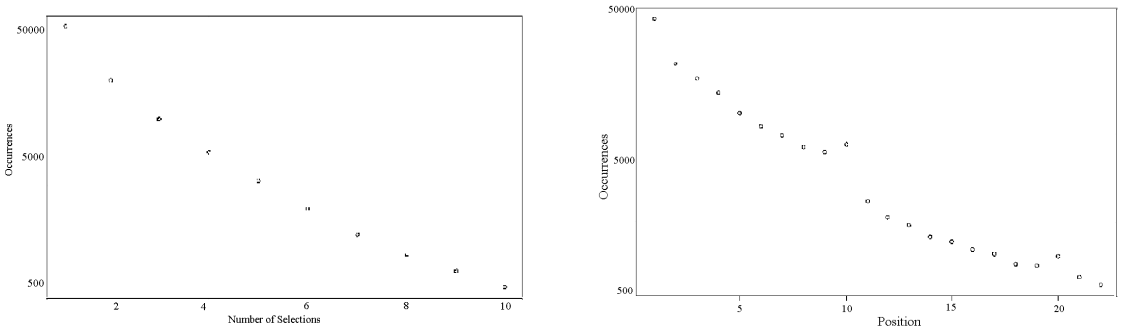
Figure 3.5: Scatter plot of the intersection between query term and text term collections.

determine the effect of the order relation between documents (ranking) and user's clicks.

Firstly, we show the number of selections per query session in the log plot in figure 3.6(A). Our data shows that over the 50% of the query sessions have only one click in their answers. On the other hand, only the 10% of the users check over five answers. The average number of clicks over all queries is 0,1525 and 1,57 without including empty sessions. The data follows a Zipf distribution where $b = 1.027$.

One important aspect for this chapter is to show the effect of the ranking over the user selections. Intuitively, we expect that pages that are shown in better ranking places have more clicks than pages with less score. The position effect is based on the following fact: the search engine shows their results ordered by the relevance to a query, thus, users are influenced by the position at which documents are presented (ranking bias). This phenomenon could be observed in figure 3.6(B). As we can see in the figure, the data follows a Zipf distribution where $b = 1.4$.

The data shows a discontinuity in positions number ten and twenty. Our data shows that this discontinuity appears also in positions 30 and 40 and, in general, in positions that



(A) (B)

Figure 3.6: (A) Log plot of number of selections per query session. (B) Log plot of number of selections per position.

are multiple of 10. TodoCL shows ten results per page result, thus, this fact cause the discontinuity (interface bias). Finally, the 79.72% of the pages selected are shown in the first result page.

Another interesting variable involved in this process is the visit time per selected page. Intuitively, this variable is important because the time spent in a selected page indicates a preference. From our data we can see that over the 75% of the visit times per page are less than 2 minutes and a half. On the other hand, the others pages show higher visit times. This fact indicate that these pages were very relevant to the their queries and obviously, this information could be used in order to improve their rankings. Besides this, an important proportion of the pages show visit times between 2 and 20 seconds. This represents the 20% of the selected pages, approximately.

3.6 Query Session Models

3.6.1 Aggregated query session model

One of the main purposes of this thesis is to describe user behavior patterns when a query session is started. We consider only non empty query sessions in this chapter, thus query instances with at least one page selected in their answers. We have focused this analysis considering two independent variables: number of queries formulated and number of pages

selected in a query session.

In a first approach, we calculated a predictive model for the number of clicks in a query session when the total number of queries formulated is known. Let $X = x$ be the random variable that models the event of formulating x queries in a query session and let $Y = y$ be the random variable that models the event of selecting y pages in a query session. This first approach models the probability of selecting y pages given that the user has formulated x queries using conditional probability $P(Y = y \mid X = x) = \frac{P(Y=y,X=x)}{P(X=x)}$. In order to do that, we consider the total number of occurrences of the events in the query session log files. Thus, the data is depicted in an aggregated way, i.e. this first approach considers a predictive model for the total number of queries and clicks in a query session. Figure 3.7 shows the first model.

As we can see, if a user formulates only one query, in general he will select only one page. Probably, this fact is caused by a precise query, i.e., the user has the need defined at the begin of the query session. As a consequence of the quality of the search engine, when the query is very precise, the user finds the page in few trials. We will say that this kind of queries are of good quality, because they enable users to find their pages quickly. If the session has two, three or four queries, probably the session will have many clicks. In general, this kind of sessions are associated to users that do a more exhaustive search pattern. This kind of users can be categorized as *browser people* because they want to look at multiple pages on one topic. Finally, if the session has more than four queries, probably users will select only one result. In general, these sessions show less specific queries at the begin than at the end. So, the user had the need to refine the query in order to find the page that he finally selects. This kind of sessions are related to users that do not formulate good quality queries.

Figure 3.8 shows the contingency table and the class distribution for the joint events, number of queries done, and number of pages selected. At first sight, the surface follows a bidimensional Zipf distribution. In order to prove our intuition, we fit a bidimensional Zipf function to the data and then we calculated the error of the approximation.

In order to adjust a bidimensional Zipf, we calculate the log - log curves from the data shown in figure 3.8. A linear surface was obtained, showing the Zipf behaviour of the data. Doing the assumption of the independence of the marginal distributions $f(X)$, where X is

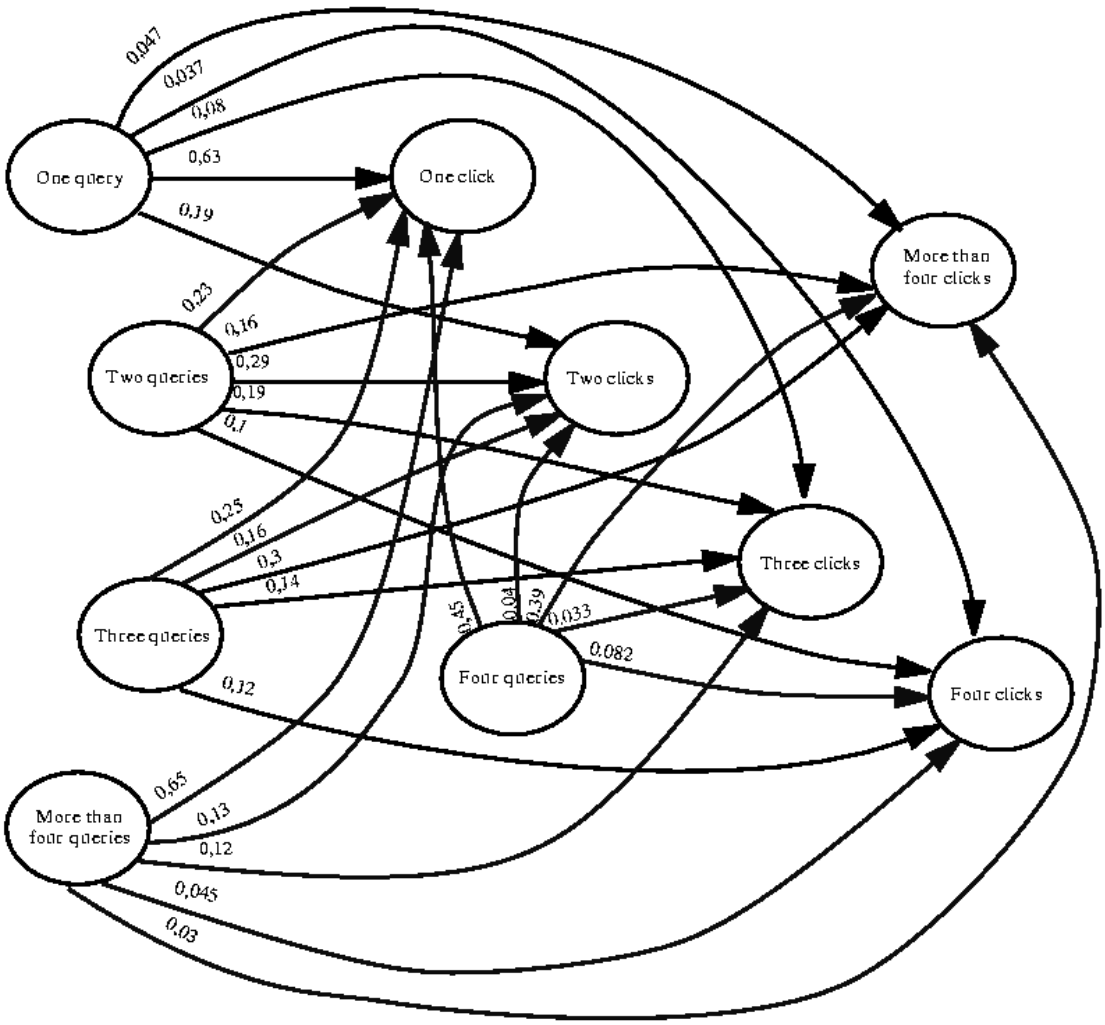
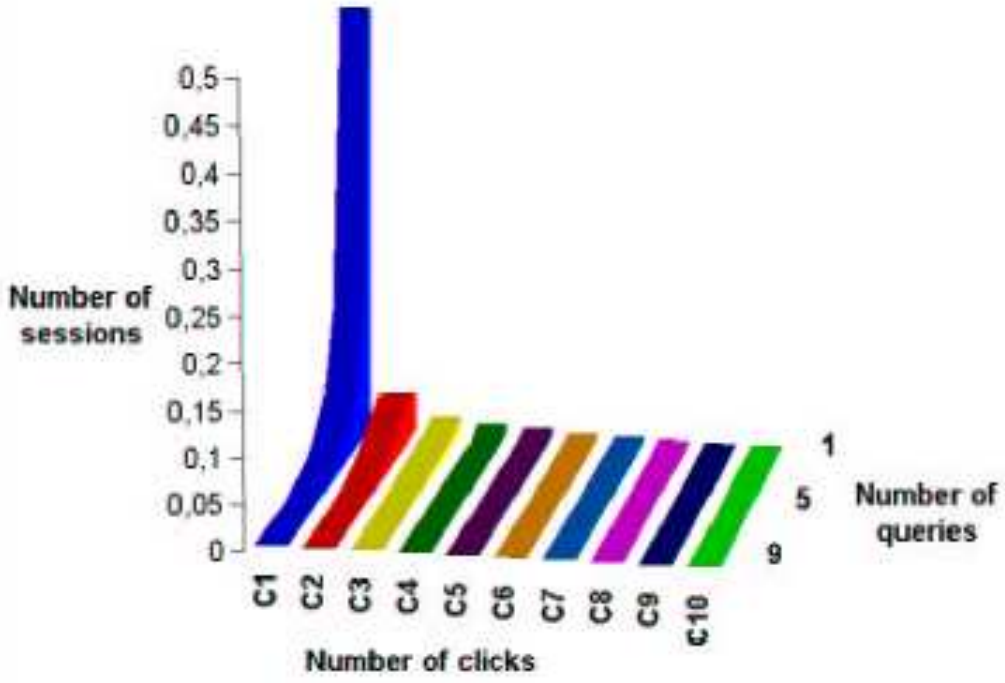


Figure 3.7: Predictive model for the number of clicks in a query session with a known number of queries formulated.

the random variable that represent the number of queries, and $f(Y)$ where Y is the random variable that represent the number of clicks, we adjusted straight lines to each surface projection into the dimensions X and Y, respectively. As a result, we build a bidimen-



	1	2	3	4	5	6	7	8	9	10
Serie1	0,464	0,141	0,059	0,027	0,015	0,008	0,004	0,003	0,002	0,001
Serie2	0,038	0,049	0,033	0,018	0,01	0,006	0,004	0,002	0,002	9E-04
Serie3	0,014	0,009	0,008	0,007	0,006	0,004	0,002	0,002	0,001	8E-04
Serie4	0,009	8E-04	7E-04	0,002	0,002	0,002	0,002	1E-03	8E-04	6E-04
Serie5	0,007	8E-04	5E-04	0,002	4E-04	7E-04	5E-04	6E-04	5E-04	4E-04
Serie6	0,003	7E-04	2E-04	4E-04	1E-04	1E-04	2E-04	3E-04	2E-04	2E-04
Serie7	0,002	5E-04	1E-04	3E-04	5E-05	6E-05	4E-05	5E-05	1E-04	1E-04
Serie8	9E-04	4E-04	2E-05	3E-05	4E-05	4E-05	1E-05	2E-05	1E-05	6E-05
Serie9	2E-04	3E-04	1E-05	1E-05	2E-05	2E-05	1E-05	1E-05	2E-05	1E-05
Serie10	2E-05	5E-05	0	0	0	0	0	1E-05	0	0

Figure 3.8: Contingency table and class distribution for the joint events number of queries made and number of pages selected.

sional Zipf doing the multiplication of the marginal distributions as follows:

$$f(X = x, Y = y) = C\frac{1}{x^a y^b},$$

where the values of the parameters were $a = 3.074$ and $b = 1.448$.

In order to prove the independence assumption, we calculate a Kolmogorov-Smirnov goodness fit test for the theoretical function and the real data. The value of the Kolmogorov-Smirnov statistic was 0.035705. Thus the hypothesis was accepted with a confidence of 99%.

3.6.2 Markovian query session model

Focused in transitions between frequent states in a query session, in this second approach we build a finite state model. Using the query log data, we have calculated probabilities between states making a Markovian query session model. In order to do that, we have considered the following states:

- *M-th query formulation*: m-th query formulated in the query session.
- *N-th document selection*: n-th document selected (n-th click) for the m-th query formulated in the query session.

States labeled as n-th document selection are related to a given query formulated. States labeled as n-th query formulation are related to a previous document selection, except the first query that starts the query session. Let X_i be the random variable that model the number of queries formulated after i transitions in a query session and let Y_i be the random variable that model the number of document selected for the last query formulated in the session after i transitions. Thus, states in a query session model could be modeled as a pair $(X_i = x, Y_i = y)$ representing that the user has selected y documents after formulating the x-th query.

In order to calculate it, we consider the data in the query log files in a non aggregated way, as the opposite of the first approach. Thus, the events are distributed over the time. As a consequence, the second model is a discrete-state, continuous-time Markov Model, being

the probability of be over a defined state $(X_i = x, Y_i = y)$ determined by the Markov chain of the $i-1$ previous states.

We have considered the same events as in the first approach but focused on two kind of transitions: the first one considers the probability of doing the m-th query given that in the $m-1$-th query users has selected n pages. The second one considers the probability of doing the n-th click given that the user has formulated the m-th query. We calculate these as follows:

$$P(X_i = m \mid X_{i-1} = m-1, Y_{i-1} = n) = \frac{P(X_i=m,Y_i=n)}{P(X_{i-1}=m-1,Y_{i-1}=n)},$$

$$P(Y_i = n \mid X_{i-1} = m, Y_{i-1} = n-1) = \frac{P(X_i=m,Y_i=n)}{P(X_{i-1}=m,Y_{i-1}=n-1)}.$$

In figure 3.9 we show the second model. Final states are depicted with concentric circles.

The initial event in query sessions is to formulate the first query and to make the first click. After k steps, the probability of being over a determined state could be calculated using the Markovian transition matrix, M, of k-th order, M^k, and the initial distribution vector $P^{(0)}$ formed by the probabilities of being at determined states at the beginning of the process. The set of probabilities of be over a state in the chain compounds the probability vector of k order given by $P^{(k)} = M^k \times P^{(0)}$.

After a large number of transitions, the $P^{(k)}$ vector converges to the $\vec{v}$ fixed point vector of the process, the stationary probabilities of the chain, that satisfy the fixed point property $\vec{v} = M \times \vec{v}$. Of course, the fixed point vector has positive values only for final states.

Using our model, we verify that after 10 steps the $P^{(k)}$ vector converge to the fixed point vector. As an example of the convergence, in the following matrix the i-th row represents the probabilities after $i+1$ steps. Each column represents final states, considering only states from 1 to 8, because the probability of be over all the other states is close to zero. We show only the first five steps and we start from $k = 2$ (for $k = 1$ the vector is null and only the transitive state $(X_1 = 1, Y_1 = 1)$ is attainable):

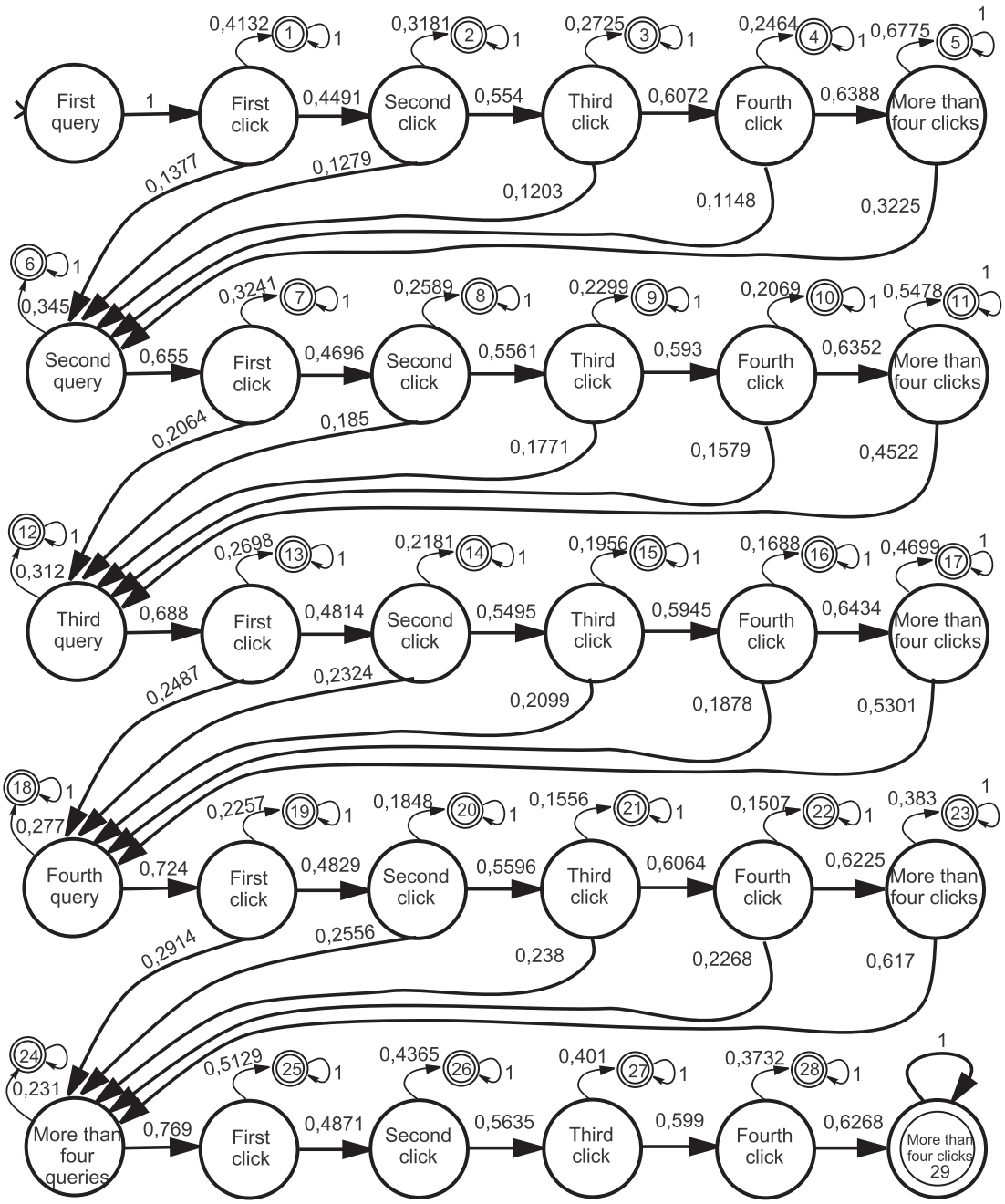


Figure 3.9: Markovian transition query session model.

$$\begin{pmatrix} 0,41 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0,41 & 0,14 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0,41 & 0,14 & 0,07 & 0,05 & 0,04 & 0,07 & 0,03 & 0 \\ 0,41 & 0,14 & 0,07 & 0,04 & 0,05 & 0,08 & 0,04 & 0,01 \\ 0,41 & 0,14 & 0,07 & 0,04 & 0,05 & 0,08 & 0,05 & 0,02 \end{pmatrix}$$

As we can see, at the begin, only the first states are possible but after few steps, probabilities are propagated over the complete chain. Finally, the fixed point vector for all the final states is:

$$\begin{aligned} \vec{v} \;=\; & (0,41; 0,14; 0,07; 0,04; 0,05; \\ & 0,08; 0,05; 0,02; 0,01; 0,005; 0,01 \\ & 0,02; 0,01; 0,005; 0,002; 0,002; 0,005 \\ & 0,015; 0,007; 0,006; 0,003; 0,001; 0,004 \\ & 0,007; 0,005; 0,001; 0,001; 0; 0,01). \end{aligned}$$

Now we calculate the probability that, starting at a given state, the chain goes into a state and never comes out. This kind of probabilities are known as *absortions*. We calculate the absortion for the first and second query instances, because they represent the most frequent transitions in real situations.

Absortion probabilities for the first query instance are given by:

$$\begin{aligned} P(\text{absortion in state 1} \mid X_0 = \text{first query}) &= 0.41 \\ P(\text{absortion in state 2} \mid X_0 = \text{first query}) &= 0.14 \\ P(\text{absortion in state 3} \mid X_0 = \text{first query}) &= 0.07 \end{aligned}$$

$$P(\text{absortion in state 4} \mid X_0 = \text{first query}) = 0.04$$
$$P(\text{absortion in state 5} \mid X_0 = \text{first query}) = 0.05$$
$$P(\text{absortion in state 6} \mid X_0 = \text{first query}) = 0.11$$
$$P(\text{absortion in other states} \mid X_0 = \text{first query}) = 0.19$$

Off course, absortion probabilities of states 1 - 5 coincide with the stationary probabilities calculated from the fixed point vector of the transition matrix. For the second query instance the absortion probabilites are the follows:

$$P(\text{absortion in state 6} \mid X_0 = \text{second query}) = 0.34$$
$$P(\text{absortion in state 7} \mid X_0 = \text{second query}) = 0.21$$
$$P(\text{absortion in state 8} \mid X_0 = \text{second query}) = 0.07$$
$$P(\text{absortion in state 9} \mid X_0 = \text{second query}) = 0.03$$
$$P(\text{absortion in state 10} \mid X_0 = \text{second query}) = 0.02$$
$$P(\text{absortion in state 11} \mid X_0 = \text{second query}) = 0.03$$
$$P(\text{absortion in state 12} \mid X_0 = \text{second query}) = 0.08$$
$$P(\text{absortion in other states} \mid X_0 = \text{second query}) = 0.18$$

3.6.3 Time distribution query session model

In a third model we address the problem of determine the time distribution between state transitions. Each transition time follows a distribution that could be determined using the log data. In order to do that, we measure the time between events in the query logs considering two kinds of transitions: the time distribution in query sessions between the query formulation and the following document selections and the time distribution between the clicks and the following query formulation.

Intuitively, a query formulation time is distributed around an expected value and for

higher times the probability density follows an exponential distribution. To calculate the time distribution we use a two parameter Weibull density function. Let t be the continuous random variable that models the time involved in a query formulation. We state that t follows a Weibull distributions if its density function is given by:

$$f(t;\alpha,\theta) = \frac{\alpha}{\theta^{\alpha}} t^{\alpha-1} e^{-(\frac{t}{\theta})^{\alpha}}, \quad t > 0, \alpha, \theta > 0.$$

The θ parameter scales the function along the horizontal axis, and the α parameter defines the shape of the function. Our data shows that the measurement errors (the difference between the density function and the data) do not have equal variance, but rather, their variance is proportional to the height of the mean curve. The Weibull distribution is often used as a model when the response variable is nonnegative. Ordinary least squares curve fitting is appropriate when the experimental errors are additive and can be considered as independent draws from a symmetric distribution, with constant variance. Off course we are modeling a time variable and the resulting distribution will be asymmetric (we consider nonnegative values for t). Thus we assume multiplicative errors, symmetric on the log scale. We can fit a Weibull curve to the data under this assumption using a nonlinear least squares method to fit the curve. In figure 3.10 we show the estimation of α, θ parameters for each transition.

Transitions between queries and clicks are highly correlated with the search engine interface effect. As we saw in section 3.5, the search engine interface produces a bias in the searching process caused by the relative position of the documents in the ranking. Intuitively, we can guess that times involved in the searching process are also correlated with the ranking. As we can see in the data, this assumption is true for the first click and these kind of transitions follows Zipf distributions. In figure 3.10 we show the b parameter value for each transition considered. We can see that values are independent of the query order. However, for transitions to second or higher order clicks, time distribution follows a Weibull as in the previous case. Intuitively this is a consequence of the searching time involved that changes the expected value from zero to higher values. As is well known, the expected value of a Weibull random variable is given by $\theta \times \Gamma\left(1 + \frac{1}{\alpha}\right)$. The expected values for each transition are given in table 3.4.

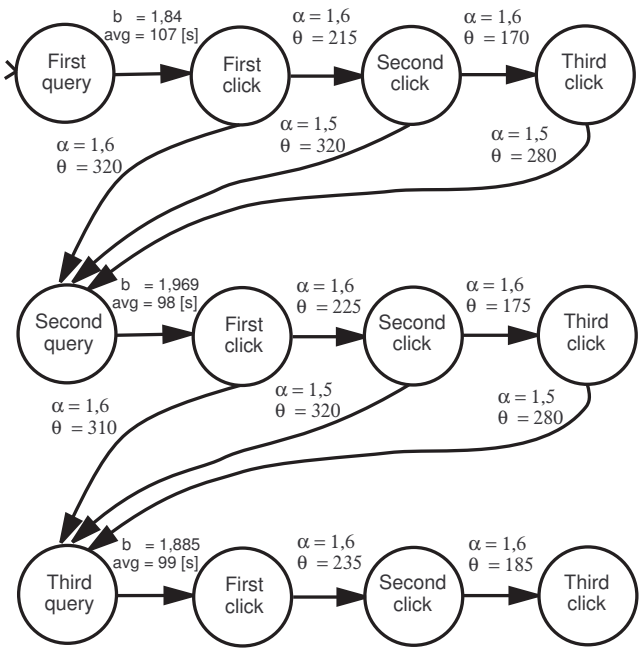


Figure 3.10: Time distribution query session model.

Times involved in query formulation are higher if the preceding states are document selections of low order. It is important to see that all these times are biased to the previous viewing time involved. In order to unbias the results, we must subtract the expected values for the viewing document times to the expected values of query formulations. However expected values for Zipf distributions are close to zero, thus the subtraction do not affect final results.

In order to test the goodness fit of the proposed model, we determine for each pair of

To - From	First click	Second click	Third click
Second click	192	201	210
Third click	151	156	165
Second query	286	288	252
Third query	277	288	252

Table 3.4: Expected values (in seconds) for the Weibull distributions involved in the query session model.

To - From	First click	Second click	Third click
Second query	0.1317	0.1439	0.1741
Third query	0.1404	0.1149	0.2055

Table 3.5: Value of the Kolmogorov statistic D_n for each transition.

events the time transitions between them, sorted as an increasing time series $x_1, x_2, \ldots, x_n$, and we calculate the Kolmogorov-Smirnov statistic as follows:

$$D_n = max_x \mid S_n(x) - F_0(x) \mid, \tag{3.2}$$

where $S_n(x)$ is the accumulative sampling distribution given by:

$$S_n(x) = \begin{cases} 0 & x < x_1 \\ \frac{k}{n} & x_k \leq x < x_{k+1} \\ 1 & x \geq x_n. \end{cases}$$

and $F_0(x)$ is the theoretical probability distribution. Then, given a confidence value α for the hypothesis test, if the value of the Kolmogorov-Smirnov statistic go over the values tabulated in [Mas51], the fitting hypothesis is rejected. Otherwise, it is accepted.

For our tests, we use a confidence of 99% and 18 classes (50 seconds each one). The threshold value for the acceptance of the test and for 18 samples is 0.371. Table 3.5 shows the results of the tests for each transition. From the table we can see that all the hypotheses were accepted.

3.7 Conclusion

In this chapter we have proposed models to describe user behavior in query sessions using query log data. The models proposed are simple and provides enough evidence to be applied to Web search systems. Our results show that Web users formulate short queries, select few pages and an important proportion of them refine their initial query in order to retrieve more relevant documents. Query log data, in general, show Zipf distributions such

as for example clicks over ranking positions, query frequencies, query term frequencies and number of selections per query session. Moreover query space is very sparse showing for example that in our data the 80% of the queries are formulated only once.

The evidence shows that query reformulation methods are useful for many users. As was shown in the model depicted in figure 4.12, users reformulate their queries in order to define adequately their needs. The exploration of query reformulation methods based on click through data will be the main goal of the chapter 4.

Based on the sparse nature of the data, any recommender system could be work with aggregated versions of the data. A significant proportion of the queries have very short sessions, with very few selected documents. Thus recommender systems based on user preferences are viable only if they work over query clusters or if they use data structures that redirect the searching process to well defined user needs where the quantity and quality of the click through data could be enough to formulate statistically significatively recommendations. The use of the data to calculate query clusters and to use them to formulate recommendations will be the objective of the chapter 5. The use of directories to redirect the searching process to well defined user needs will be the goal of the chapter 6.

Chapter 4

Query Association Methods

In this chapter we will introduce a method to identify relations between queries. The basic idea is to identify groups of related queries in order to replace an original query with a better one. The application proposed is query recommendation. Firstly in section 4.1 we will explore a naive approach: two queries are similar if they share their terms. We will show that this idea allow to identify similar queries but with a limitation: the query term space is sparse as was shown in the previous chapter, so the method is useful only for a few queries. In section 4.2 we will explore an alternative idea: we order the documents selected during past sessions of a query according to the ranking of other past queries. If the resulting ranking is better than the original one, we recommend the associated query. To show the effectiveness of this method we set experiments with real user's click data evaluating the precision of the recommended queries.

This chapter has been partially published in [DM05, DM06].

4.1 The query term based method

4.1.1 The method

Queries have the goal of describing needs of users. However, in many cases, the query submit by a user is just a first approximation to a better description of the information being searcher. A better description requires the user to have knowledge about the information

being searched, such as technical terminology and proper nouns. It may even require an empirical understanding of the behavior of the search engine.

In order to allow the user to reformulate the original query, we propose to identify similar queries. Then the user may move to a more general/specific query. Then she/he can select a query and then can follow a suggested document.

To identify similar queries we will explore firstly a naive idea: two queries are similar if they have common terms. Intuitively, if two or more queries share terms, they are semantically connected because the terms are related to a common meaning.

In this approach we represent each query by a list of terms. To calculate the similarity between two queries we will use a widely studied similarity function: the quotient between the intersection and the union of both sets:

$$Sim(q_a, q_b) = \frac{L_a \bigcap L_b}{L_a \bigcup L_b}, \tag{4.1}$$

where L_a and L_b represent the list of query terms of the queries q_a and q_b, respectively. Intuitively, when two queries share several terms, the value of $Sim(q_a, q_b)$ is close to 1 and when two queries do not share terms, the value of $Sim(q_a, q_b)$ is close to 0.

Using the similarity function given by 4.1 we can identify groups of related queries. In order to identify relations in the group we use the user's click data. Based on this information source we can identify how many users re-formulated a query after the formulation of an original one. The reformulation will be more representative of the common user needs when its relevance, calculated as the proportion of the users that reformulate the original query with the new query over the total number of users who formulates the original query, will be greater than other reformulation relevances.

When a user formulates a query and then she/he specialize or generalize the original query selecting a suggested query, he is given a positive vote for the relation. Using the user's click data it is possible to label the arcs in the graphs, where each label represents the number of users which use the relation to reformulate the original query.

4.1.2 Experimental results

To test the method we used the log file denoted by L_3 in section 3.2. We represent each query by a node in a directed graph. The reformulation processes are identifying by arcs in the graph, labeled with the number of users that specialize/generalize the original query using the suggested query. The label value represents the relevance of the relation.

With this method we can build graphs related to general concepts such as `computation`. If the query terms are specific the graph will represent a well defined meaning. It is the case of the *computation* topic. Computation terminology is very specific so using query terms it is possible to identify group of queries that are related to the same meaning. As we show in figure 4.1, we can identify several queries related to computation that allow to specify/generalize the original queries.

We show into boxes the main concepts related to computation that are used as queries in order to reformulate the original queries. The label of the arcs show the number of users who reformulate the query using the query which is pointed by the arc.

The graph shown in 4.1 represents a good example of the utility of the method based on query terms. This is an effect of the terminology used in the area, because related terms do not have meaning related to other topics (they are unambiguous). Thus, the terms consign the meaning of the group. Unfortunately, there are query terms that have several meanings. We will call this kind of terms *polysemic terms*.

When users formulate their queries using polysemic terms it is more difficult to identify the meaning behind the query. We show in figure 4.2 queries related to the concept law. Into the box we show the most relevant query for this group. Users that formulate queries related to the concept *law* reformulate their original queries formulating the query *penal procedural code*. The label of each arc shows the number of users that reformulate the query.

We can see that the queries share the term `code` (except the query `penal procedures` which share the term `penal` with the main query). The term `code` is the term that join this group, but not always consign the meaning of the group. For example, the query `bar code` is not related to the concept *law* but it belongs to the group. This kind of phenomenon it could be interpreting as *noise*.

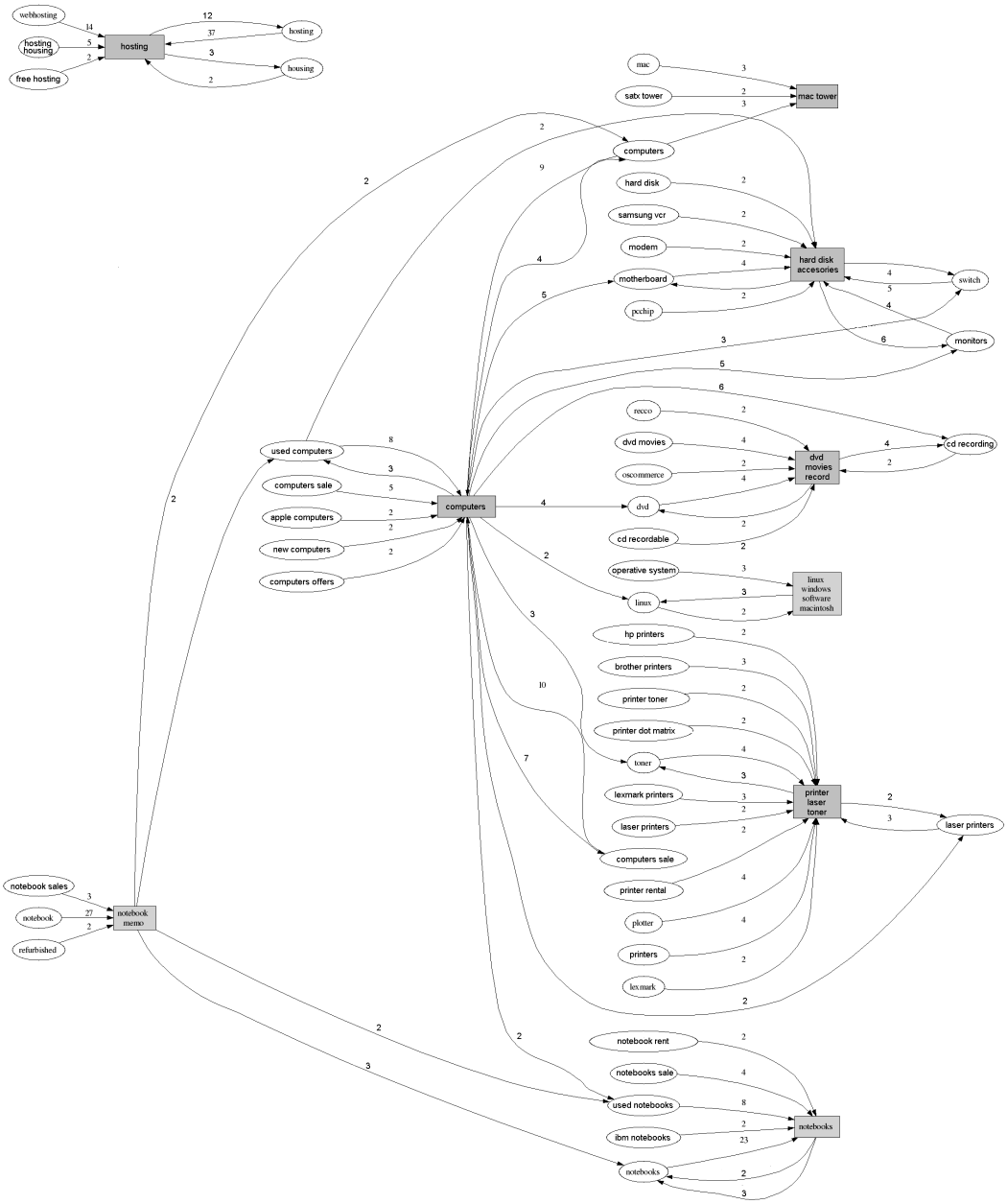


Figure 4.1: Recommendations based on query terms associated to the concept *computation*.

The noise effect is introducing because the term `code` is polysemic. As an effect of the use of polysemic words, two queries who share terms do not always have the same meanings. The example in figure 4.2 shows clearly this. In general, we can easily see that the query terms not always consign the need behind the query.

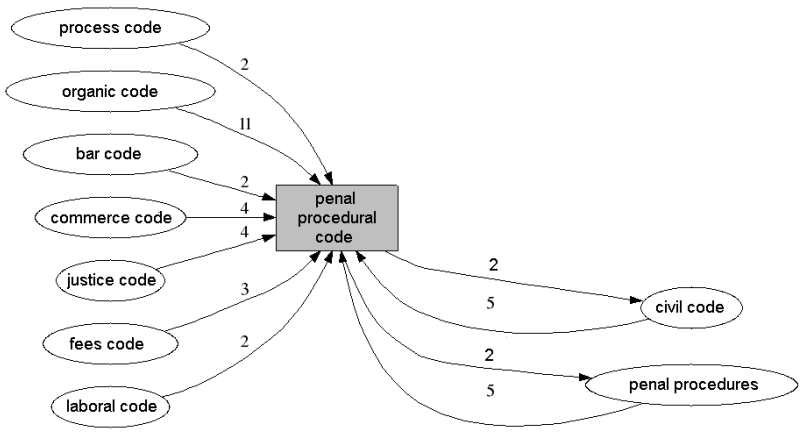


Figure 4.2: Recommendations based on query terms associated to the concept *law*.

We can stay that recommendations based on query terms are successful when the terminology of the concept is well defined and the terms are not polysemic. The main problem of this approach is the following: the method assumes that terms are independent in their occurrences. Thus the method does not identify relations between different query terms which share their meanings. In order to change this assumption we will explore an alternative idea in the following section.

4.2 The co-citation method

4.2.1 The method

The simple method we propose in this section aims at discovering alternate queries that improve the search engine ranking of documents: We order the documents selected during past sessions of a query according to the ranking of other past queries. If the resulting ranking is better than the original one, we recommend the associated query.

In order to formalize our idea, we will start defining a notion of consistency between a query and a document. A document is `consistent` with a query if it has been selected a significant number of times during the sessions of the query. Consistency ensures that a query and a document bear a natural relation in the opinion of users and discards documents that have been selected by mistake once or a few time. Similarly, we say that a query and a set of documents are consistent if each document in the set is consistent with the query.

Many identical queries can represent different user information needs. Depending on the topic the user has in mind, he will tend to select a particular sub-group of documents. Consequently, the set of selections in a session reflects a sub-topic of the original query. We might attempt to assess the existing correlations among the documents selected during sessions of a same query, create clusters and identify queries relevant to each cluster, but we prefer a simpler, more direct method where clustering is done at the level of query sessions.

Let $D(s_q)$ be the set of documents selected during a session s_q of a query q. If we make the assumption that $D(s_q)$ represents the information need behind q, we might wonder if other queries are consistent with $D(s_q)$ and better rank the documents of $D(s_q)$. If these queries exist, they are potential query recommendations. We then repeat the procedure for each session of the original query, select the potentially recommendable queries that appear in a significant number of sessions and propose them as recommendations to the user interested in q.

We need to introduce a criteria in order to compare the ranking of a set of documents for two different queries. Firstly, we define the `rank` of a document in a query: The rank of document u in query q, denoted $r(u, q)$, is the position of document u in the answer list returned by the search engine. We extend this definition to sets of documents: The rank of a set U of documents in a query q is defined as:

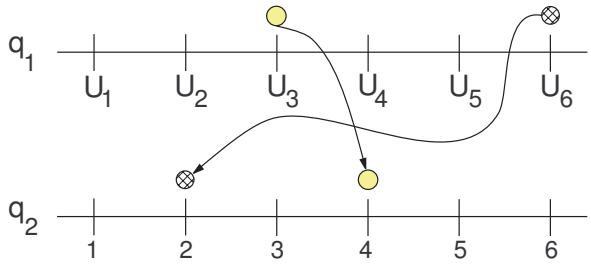


Figure 4.3: Comparison of the ranking of two queries.

$$r(U, q) \quad = \quad \max_{u \in U} r(u, q) \quad .$$

In other words, the document with the worst ranking determines the rank of the set. Intuitively, if a set of documents achieves a better rank in a query q_a than in a query q_b, then we say that q_a ranks the documents better than q_b. We formalize this as follows: Let U be a set of common documents between two queries denoted by q_a and q_b. A query q_a ranks better the set U of documents than a query q_b if:

$$r(U, q_a) < r(U, q_b) \quad .$$

This criteria is illustrated in Fig. 4.3 for a session containing two documents. A session of the original query q_1 contains selections of documents U_3 and U_6 appearing at position 3 and 6 respectively. The rank of this set of document is 6. By contrast, query q_2 achieves rank 4 for the same set of documents and therefore qualifies as a candidate recommendation.

Now, it is possible to recommend queries comparing their rank sets. We can formalize the method as follows: A query q_a is a recommendation for a query q_b if a significant number of sessions of q_a are consistent with q_b and are ranked better by q_a than by q_b.

The recommendation algorithm induces a directed graph among queries. The original query is the root of a tree with the recommendations as leaves. Each branch of the tree

represents a different specialization or sub-topic of the original query. The depth between a root and its leaves is always one, because we require the recommendations to improve the associated document set ranking.

Finally, we observe that nothing prevents two queries from recommending each other. We will stay that two queries are quasi-synonyms when they recommend each other. Thus, if two or more queries are quasi-synonyms, they form a cluster of similar queries. We will see in the following section that this definition leads to queries that are indeed semantically very close. Off course this is not the main goal of our proposal. The literature shows several methods to discover groups of quasi-synonyms queries. From our point of view, its a good sign in order to set the correcteness of our method identifying related queries. The final goal of our method is the following: to recommend better queries more than similar queries.

4.2.2 Experimental Results

In this section we present the evaluation of the method introducing a descriptive analysis of the results and showing a user evaluation experiment. Firstly we will describe the data used for the experiments.

Data

The algorithm was easy to implement using log data organized in our query log relational database. For the experiments of this section we use the three months log file denoted by L_3 in section 3.2.

Descriptive Analysis of the Results

We intend to illustrate with examples that the recommendation method has the ability to identify sub-topics and suggest query refinement. Fig. 4.4 shows the recommendation graph for the query `Valparaiso`. The number inside nodes refer to the number of sessions registered in the logs for the query. The edge numbers count the sessions improved by the pointed query. A given session can recommend various queries. They are reported

only for nodes having children, as they help make sense of the numbers on the edges which count the number of users who would have benefited of the query reformulation.

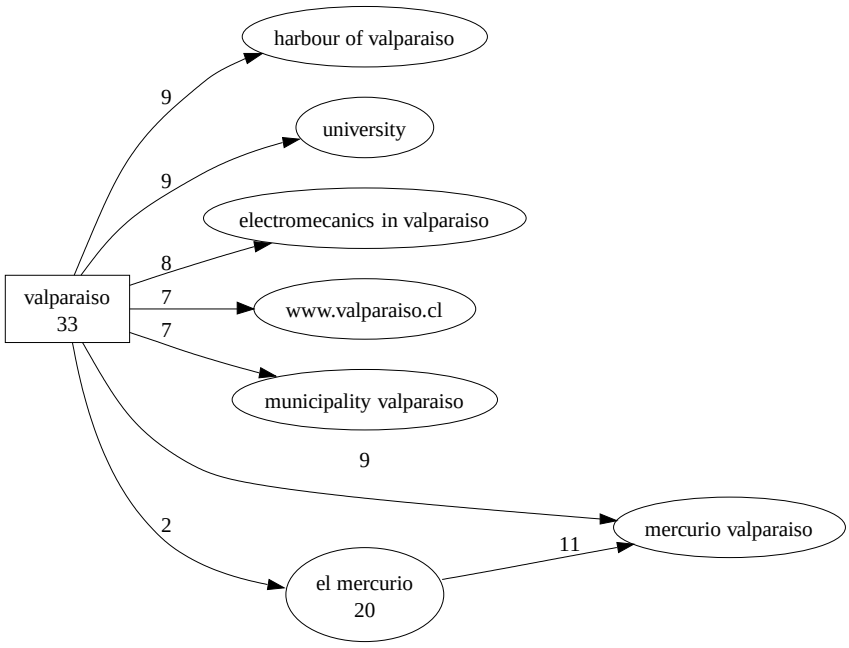


Figure 4.4: Query `Valparaiso` and associated recommendations.

This query requires some contextual explanation. Valparaiso is an important touristic and harbor city, with various universities. It is also the home for the *Mercurio*, the most important Chilean newspaper. Fig. 4.4 captures all these informations. It also recommends some queries that are typical of any city of some importance like city hall, municipality and so on. The more potentially beneficial recommendations have a higher link number. For example $9/33 \simeq 27\%$ of the users would have had access to a better ranking of the documents they selected if they had searched for `university` instead of `Valparaiso`. This also implicitly suggests to the user the query `university Valparaiso`, although we are maybe presuming the user intentions.

The query `maps` illustrated in Fig. 4.5 shows how a particularly vague query can be disambiguated. The `TodoCL` engine user's click data bias naturally the recommendations towards Chilean information.

A third example concerns the `''naval battle in Iquique''` in May 21, 1879 between Peru and Chile. The query graph is represented in Fig. 4.6. The point we want

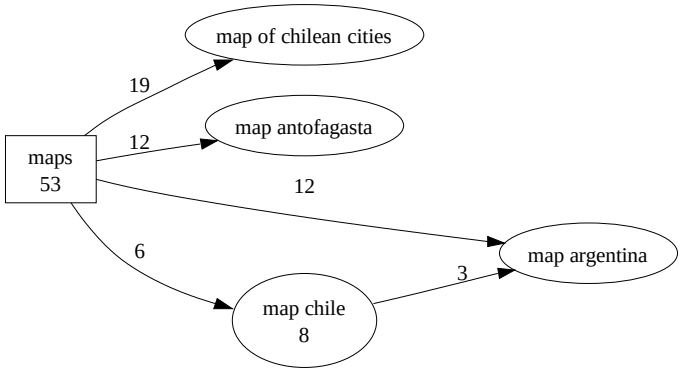


Figure 4.5: Query `maps` and associated recommendations

to illustrate here is the ability of the algorithm to extract from the logs and to suggest to users alternative search strategies. Out of the 47 sessions, 2 sessions were better ranked by `armada of Chile` and `sea of Chile`.

In turn, out of the 5 sessions for `armada of Chile`, 3 would have been better answered by `sea of Chile`. Probably the most interesting recommendations are for `biography of Arturo Pratt`, who was captain of the "Esmeralda", a Chilean ship involved in the battle, for the `Ancn treaty` that was signed afterward between Peru and Chile and for the `Government of Jos Joaqun Prieto`, responsible for the war declaration against the Per-Bolivia Federation.

A more complex recommendation graph is presented in Fig. 4.7. The user who issued the query `fiat` is recommended essentially to specify the car model he is interested in, if he wants spare parts, or if he is interesting in selling or buying a fiat. Note that such a graph also suggests to a user interested in – say – the history or the profitability of the company to issue a query more specific to his needs.

We already observed that two queries can recommend each other. We show in Table 4.1 a list of such query pairs found in the logs. We reported also the number of original query sessions and number of sessions enhanced by the recommendation so as to have an appreciation of the statistical significance of the links. We excluded the mutual recommendation pairs with less than 2 links. For example, in the first row, out of the 13 sessions for `ads`, 3 would have been better satisfied by `advert`, while 10 of the 20 sessions for `advert` would have been better satisfied by `ads`.

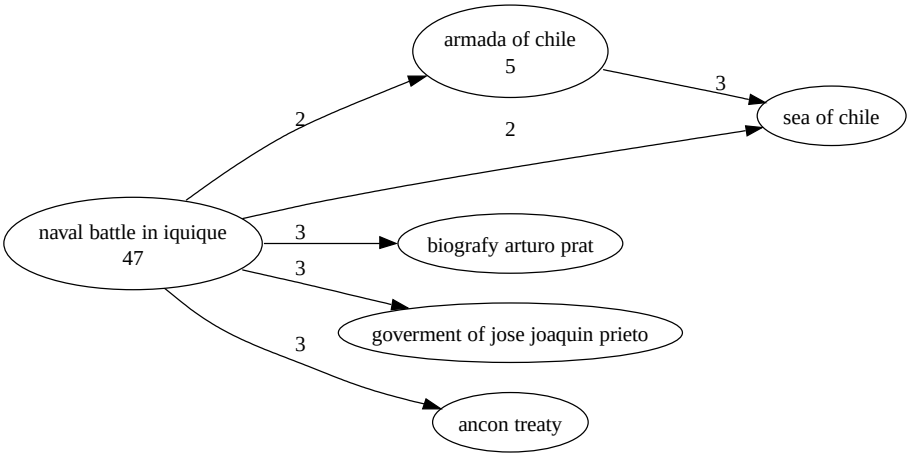


Figure 4.6: Query `Naval battle in Iquique` and associated recommendations.

We can see that the proposed method generates a clustering *a posteriori* where different sessions consisting of sometimes completely different sets of documents end up recommending a same query. This query can then label this group of session.

User Evaluation

We will analyze user evaluations of the quality of query recommendations. We presented to a group of 19 persons of different backgrounds ten recommendation trees similar to Fig. 4.4, selected randomly from all the trees we extracted from the logs. Obviously we discarded queries with a small number of sessions and queries with a too large number of recommendations. We asked two questions to the participants:

1. What percentage of recommendations are relevant to the original query?
2. According to our intuition, what is the percentage of recommendations that will improve a typical user query?

In figure 4.8 we show the distribution of the opinions for the both questions. The figure on the left is related to the first question raised to the participants. It reports on abscissa the percentage of relevant recommendations and in ordinate the number of participant votes. On the right figure, we plotted the results for improved recommendations corresponding to the second question.

→	query	query	←
3/13	ads	advert	10/20
3/105	cars	used cars	2/241
34/284	chat	sports	2/13
2/21	classified ads	advertisement	2/20
4/12	code of penal proceedings	code of penal procedure	2/9
3/10	courses of english	english courses	2/5
2/27	dvd	musical dvd	2/5
2/5	family name	genealogy	2/16
3/9	hotels in santiago	hotels santiago	2/11
5/67	mail in Chile	mail company of Chile	2/3
7/15	penal code	code of penal procedure	2/9
8/43	rent houses	houses to rent	2/14
2/58	van	light truck	3/25

Table 4.1: Examples of "Quasi-synonym" queries recommend each other.

We used two factors analysis of variance with no interaction to test whether there is a large variation between participants opinions and between queries. For the first question concerning the relevance of the recommendations, the $p - values$ of the variation induced by the participants is 0.5671 and by the queries is 0.1504, leading to accepting the hypothesis H_0 that none of these variations is statistically significative. The same conclusion holds for question 2 about whether the recommendations might improve the original query. The $p - values$ in this case are 0.9991 and 0.2130. This shows that no participant was over or under estimating the relevance and the improvement percentages systematically, and that no query is particulary worse or better than the other. The recommendations along with the mean of participants answer can be found in Table 4.2. Average relevance value is shown in column 2. Average improvement value is shown in column 3. Wrong recommendations are in cursive fonts. Trademarks are in bold fonts.

Some queries and recommendations in this table are specific to Chile: "El Mercurio" is an important national newspaper, "Colmena" is a private health insurance company but the term "colmena" in Spanish also means beehive. It seems that people searching for honey producers where fooled by a link to the "Colmena" health insurance company. Some sessions of the query for "health insurance company" contained links to "Colmena" that appear high in the ranking for "honey bees". This problem should disappear if we fix

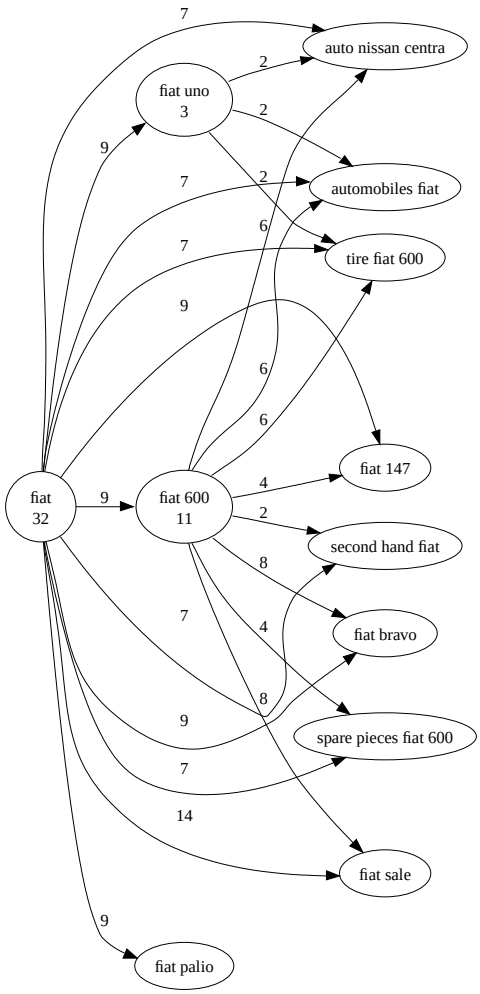


Figure 4.7: Query `fiat` and associated recommendations.

a higher consistency threshold, which would be possible with larger logs. Folklore and Biology are important activities in the “University of Chile” that users might look for. “Wei” is a computer store.

4.3 Conclusion

In this chapter we have introduced one method for query recommendation based on user’s click data that are simple to implement and has low computational costs. The recommendation method we propose here are made only if they are expected to improve the original

Figure 4.8: Distribution of the opinions for the both questions.

query. Moreover, the algorithms do not rely on the particular terms appearing in the documents, making it robust to alternative formulations of an identical information need. Our experiments show that query graphs induced by our methods identify information needs and relate queries that allow to specify the original query.

Among the limitations of the methods, one is inherent to user's click data: Only queries present in the logs can be recommended and can give rise to recommendations. On the other hand, some recommendations are related to the terms of the original query so in these cases its difficult to specify alternative concepts. Thus, the query specification process is limited to the quality of the original query. Finally, queries change in time and query recommendation systems must reflect the dynamic nature of the data. This will be the focus of the next chapters.

Query	Relevance	Improvement	Recommended queries
map of Santiago	81%	68%	map of Chile Santiago city city map of Santiago street map of Santiago
naval battle of Iquique	78%	76%	Arturo Prat biography J. Prieto government treaty of Ancon Chilean navy
computers	74%	61%	*sky Chile* **Wei** motherboards notebook
used trucks	70%	53%	cars offers used cars sales used trucks rentals trucks spare parts
El Mercurio	62%	52%	**El Mercurio** of December *currency converter* **El Mercurio de Valparaiso** **El Mercurio de Antofagasta**
wedding dresses	61%	55%	dress rentals wedding dress rentals party dresses bachelor party
health insurance companies	58%	55%	**Banmedica** **Colmena** *honey bees* contralory of health services
people finders	52%	45%	Chilean finder Argentinian finder OLE search Finder of Chilean people
dictionary	41%	34%	English dictionary dictionary for technology look up words in a dictionary tutorials
Universidad de Chile	41%	25%	**Universidad Catolica** university folklore biology

Table 4.2: Queries used for the experiments and recommendations with strong levels of consistency, sorted by relevance.

Chapter 5

Query Clustering Methods

In this chapter we will introduce methods to identify similar queries based on vectorial representations of query sessions using clustering algorithms. Firstly we will use a query sessions vectorial representations considering terms of clicked documents pounder by the number of clicks of these documents in the sessions. Secondly we will propose to use snippet terms. Applications proposed are document and query recommendation. To show the effectiveness of our methods we set experiments with real user's click data evaluating the precision of the recommended items.

This chapter has been partially published in [BYHM04b], [BYHM04a] and [BYHM07].

5.1 The method based on document terms

We will identify similar queries representing them in a term vectorial space. Then we can cluster the queries using traditional techniques of data mining. Two design factors are involved in this step: the vectorial query representation and the clustering technique. In this chapter we will focus the first problem: to define an appropriate query session representation. The second element is address adopting a simple and popular clustering technique: k-direct clustering. Basically, k-direct clustering methods need to define as the input parameter the number of cluster. In a first approach, we do not know how many clusters are in the query space. Thus we will define arbitrarily the number of cluster. Then

we will repeat the clustering experiment with a different number of cluster comparing the quality of both solutions. This will be done for several number of cluster values choosing the solution with the best performance. In this chapter we will use the most popular k-direct algorithm: k-means.

Thus basically our problem is reduced to find a good vectorial representation for the queries. A first approach is to represent each query by their query terms but as we know from chapter 3, the query term space is sparse. But considering the user's click data we can represent each of them by the document terms selected en each query session. As a consequence the term space is expanded from the query term space to the document term space overcoming the problem of sparseness.

In this method, we will use a simple notion of query session similar to the notion introduced by Wen *et al.* [WNZ01] which consists of a query, along with the URLs clicked in its answer.

$$\texttt{QuerySession} := (\texttt{query}, (\texttt{clickedURL})^*)$$

A more detailed notion of query session may consider the rank of each clicked URL and the answer page in which the URL appears, among other data that can be considered for further versions of the algorithm we present in this thesis. Our representation considers only queries that appear in the query-log.

Our vocabulary is the set of all different words in the clicked URLs. *Stopwords* (frequent words) are eliminated from the vocabulary considered. Each term is weighted according to the number of occurrences and the number of clicks of the documents in which the term appears.

Given a query q, and a URL u, let $\texttt{Pop}(q, u)$ be the popularity of u (fraction of clicks) in the answers of q. Let $\texttt{Tf}(t, u)$ be the number of occurrences of term t in URL u. We define a vector representation for q, $\vec{q}$, where $\vec{q}[i]$ is the i-th component of the vector associated to the i-th term of the vocabulary (all different words), as follows:

$$\vec{q}[i] = \sum_{URLu} \frac{\texttt{Pop}(q, u) \times \texttt{Tf}(t_i, u)}{\max_t \texttt{Tf}(t, u)} \tag{5.1}$$

where the sum ranges over all clicked URLs. Note that our representation changes

the inverse document frequency by click popularity in the classical `tf-idf` weighting scheme.

Intuitively, each component of the query term vector represent the relevance of the term for the query representation. This is calculated as follows: for each document we calculate the product between the absolute frequency of the term in the document (the number of occurrences of the term in the document) and the popularity of the document. As $\texttt{Pop}(q, u)$ is defined as the fraction of clicks of d in the answers of q, it takes values in the $[0, 1]$ range. Thus, in order to normalize each component, we divide by the number of occurrences of the most frequent term $\max_t \texttt{Tf}(t, u)$. As we normalize each document term weight with the popularity, the sum over all the selected documents takes values in the $[0, 1]$ range. As a consequence, the query vectorial representation generates a document term space in $[0, 1]^n$, where n is the size of the vocabulary.

Different notions of vector similarity (e.g., cosine function or Pearson correlation) can be applied over the proposed vectorial representation of queries. In this chapter we will use the cosine function, which considers two documents similar if they have similar proportions of occurrences of words (but could have different length or word occurrence ordering).

The notion of query similarity we propose has several advantages: (1) it is simple and easy to compute; (2) it allows to relate queries that happen to be worded differently but stem from the same information need; (3) it leads to similarity matrices that are much less sparse than matrices based on previous notions of query similarity; (4) the vectorial representation of queries we propose yields intuitive and useful feature characterizations of clusters, as we will show in the section of experiments.

5.1.1 Application: Query Recommendations

The query recommender algorithm operates in the following steps:

1. Queries along with the text of their clicked URL's extracted from the Web log are clustered as was explained in the previous section. This is a preprocessing phase of the algorithm that can be conducted at periodical and regular intervals.

2. Given an *input query* (i.e., a query submitted to the search engine) we first find the cluster to which the input query belongs. Then we compute a rank score for each

query in the cluster. The method for computing the rank score is presented next in this section.

3. Finally, the related queries are returned ordered according to their rank score.

The rank score of a related query measures its interest and is obtained by combining the following notions: (1) **Similarity of the query**. The similarity of the query to the input query. It is measured using the notion of similarity introduced previously. (2) **Support of the query**. This is a measure of how relevant is the query in the cluster.

One may consider the number of times the query has been submitted as the support of a query. However, by analyzing the logs in our experiments we found popular queries whose answers are of little interest to users. In order to avoid this problem we define the support of a query as the fraction of clicks in answers of the query. It is estimated from the query log as well. As an example, the query *free advertisement* has a relatively low popularity (5.76%) in its cluster, but users in the cluster found this query very effective, as its support in Figure 5.3 shows.

The similarity and support of a query can be normalized, and then linearly combined, yielding the rank score of the query. Another approach may consider to output a list of suggestions showing the two measures to users, and to let them tune the weight of each measure for the final rank.

5.1.2 Application: Document Recommendations

The document recommender algorithm operates in the following way:

1. Queries and clicked URLs are extracted from the Web logs and clustered as was explained previously.

2. For each cluster C_i, compute and store the following; a list Q_i containing queries in the cluster; and a list U_i containing the k-most popular URLs in C_i, along with their popularity. Otherwise, do nothing.

This ranking can then be used to boost the original ranking algorithm using:

$$\texttt{NewRank}(u) = \beta\texttt{OrigRank}(u) + (1-\beta)\texttt{Rank}(u). \tag{5.2}$$

In this expression, `OrigRank(u)` is the current ranking returned by the search engine and `Rank(u)` is given by the order of the URLs in the cluster.

5.1.3 Experimental results

We compute the clusters by successive calls to a k-means algorithm, using the CLUTO software package[1]. We chose an implementation of a k-means algorithm for the simplicity and low computational cost of this approach, compared with other clustering algorithms. In addition, the k-means implementation chosen has shown good quality performance for document clustering. We refer the reader to [ZK04] for details. Since queries in our approach are similar to vectors of Web documents, the requirements for clustering queries in our approach are similar to those for clustering documents.

The quality of the resulting clusters will be measure using a criterion function, adopted by common implementations of a k-means algorithm [ZK02]. The function measures the total sum of the similarities between the vectors and the centroids of the cluster that are assigned to. Let C_r be a cluster found in a k-way clustering process ($r \in 1 \ldots k$), and let c_r be the centroid of the rth cluster. The criterion function I is defined as:

$$I = \frac{1}{n}\sum_{r=1}^{k}\sum_{v_i \in C_r} sim(v_i, c_r), \tag{5.3}$$

where the centroid c_r of a cluster C_r is defined as $\frac{(\sum_{v_i \in C_r} v_i)}{|C_r|}$, and n is the number of queries.

Since in a single run of a k-means algorithm the number of clusters k is fixed, we determine the final number of clusters by performing successive runs of the algorithm. Using real log data we show in figure 5.7 the quality of the clusters found for different values of k. The curve below shows the incremental gain of the overall quality of the

[1]CLUTO is a software package developed at the University of Minnesota that provides a portfolio of algorithms for clustering collections of documents in high-dimensional vectorial representations. For further information see http://www-users.cs.umn.edu/ karypis/cluto/.

clusters.

We use the heuristic of determining a value of k for which the increase of the solution quality flattens markedly. We selected $k = 600$. We ran the clustering algorithm on a Pentium IV computer, with CPU clock rate of 2.4 GHz, 512MB RAM, and running Windows XP. The algorithm took 64 minutes to compute the 600 clusters.

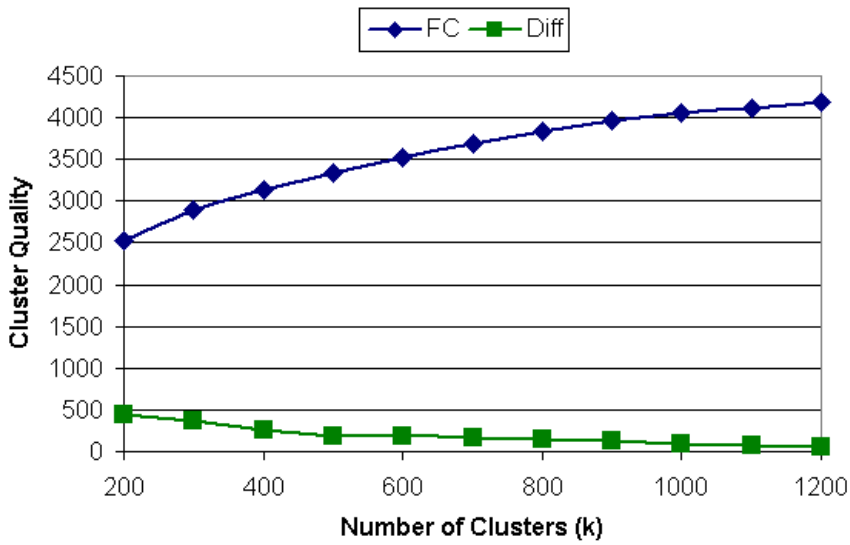


Figure 5.1: Cluster quality vs. number of clusters.

We will feature a precision experiment in order to evaluate the quality of the recommended items. Firstly we will process a query log file to obtain a significant number of query sessions to represent queries according to our proposed vector. Second we will select ten queries and their clusters for the experiments. Then we will set a experiment based on user opinions about the quality of the recommended items considering both applications: documents and queries. We will also describe the quality of the clusters obtained showing features and descriptive words.

In these first experiments we will work with a small query log file denoted by L_1 in section 3.2. In our experiments we consider ten queries: (1) *theater* (*teatro*); (2) *rental apartments viña del mar* (*arriendo de departamentos en viña del mar*); (3) *chile rentals*

(*arriendos chile*); (4) *recipes* (*recetas*); (5) *roads of chile* (*rutas de chile*); (6) *fiat*; (7) *maps of chile* (*mapas de chile*); (8) *resorts of chile* (*resorts de chile*); (9) *newspapers* (*diarios*) and (10) *tourism tenth region* (*turismo décima región*). The ten queries were selected following the probability distribution of the 6042 queries of the log. The original queries along with the results shown in this section are translated from Spanish to English. Figure 5.2 shows the clusters to which the queries belongs. Besides we show the cluster rank, the quality of each cluster (average internal similarity), the cluster size (number of queries that belongs to each cluster) and the set of feature terms that best describe each cluster. Right next to each feature term, there is a number that is the percentage of the within cluster similarity that this particular feature can explain.

Figure 5.3 shows the ranking suggested to Query 3 (*chile rentals*). The second column shows the popularity of the queries in the cluster. The third column shows the support, and the last column depicts the similarity of the queries. The queries are ordered according to their similarity to the input query.

For a non-expert user the keyword *lehmann* may be unfamiliar for searching rental adds. However, this term refers to a rental agency having a significant presence in Web directories and ads of rentals in Chile. Notice that our algorithm found relation between queries without related terms.

Query Recommendation Evaluation In order to asses the quality of the queries recommended, we follow a similar approach to Fonseca *et al* [FGDMZ03]. The relevance of each query to the input query were judged by members of our department. They analyzed the answers of the queries and determined the URL's in the answers that are of interest to the input query. Our results are given in graphs showing precision vs. number of recommended queries. Figure 5.4 shows the average precision for the queries considered in the experiments.

The figure shows the precision of a ranking obtained using the similarity, support, and popularity of the queries. The graphs show that using the support measure, in average, we obtain a precision of 80% for the first 3 recommended queries. For both popularity and similarity, the precision decreases, however the popularity rank has better precision than the the other methods.

Q	Cluster Rank	ISim	Size	Query Selected	Descriptive Keywords
q_1	15	0,98	8	theater	productions $(18,4\%)$ *Campbell* productions $(7,7\%)$ dance $(4,5\%)$
q_2	81	0,709	15	rental apartments *Viña del Mar*	real estate $(21,7\%)$ property $(17,0\%)$ used $(11,1\%)$
q_3	124	0,618	9	*Chile* rentals	storehouse $(5,3\%)$ warehouses $(4,6\%)$ office $(3,0\%)$
q_4	136	0,588	7	recipes	food $(28,4\%)$ soft drinks $(9,4\%)$ eggs $(2,2\%)$
q_5	147	0,581	14	roads of *Chile*	maps $(10,8\%)$ springs $(4,2\%)$ ski $(4,0\%)$
q_6	182	0,519	8	*Fiat*	spare parts $(28,2\%)$ shock absorber $(3,9\%)$ mechanic $(3,1\%)$
q_7	220	0,481	7	maps of *Chile*	maps $(50,3\%)$ geological $(1,1\%)$ *Mapcity* $(1,0\%)$
q_8	306	0,420	11	resorts of *Chile*	hotels $(69,2\%)$ region $(1,4\%)$ bay $(0,5\%)$
q_9	421	0,347	7	newspapers	journal $(25,6\%)$ *el mercurio* $(18,1\%)$ *estrategia* $(1,9\%)$
q_{10}	597	0,264	7	tourism tenth region	*Montt* $(17,9\%)$ *Osorno* $(5,5\%)$ *Chaitén* $(3,7\%)$

Figure 5.2: Clusters for the experiment, sorted by their cluster rank.

Query	Pop. (%)	Support (%)	Similarity
rentals	23,74	0,24	0,998
real estate	1,44	0,1	0,9852
lehmann properties	0,72	0,1	0,963
properties	56,83	0,19	0,7203
parcel purchase	3,6	0,1	0,7089
rentals offices	2,52	0,19	0,655
free advertisement	5,76	0,29	0,602
rentals apartments	3,6	0,24	0,396

Figure 5.3: Ranking of queries recommended to the query *Chile rentals*.

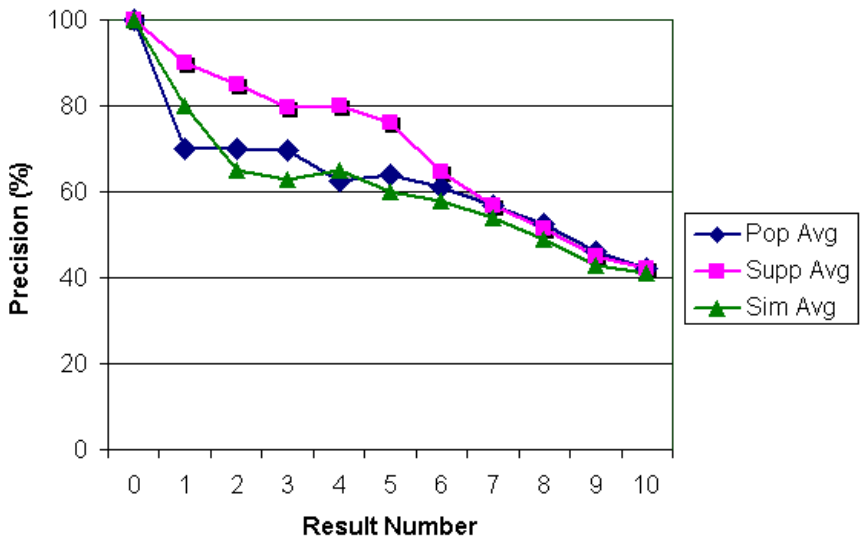


Figure 5.4: Average retrieval precision for the queries considered in the experiments.

Document Recommendation Evaluation Figure 5.5 illustrates the average retrieval precision of TodoCL and the proposed ranking algorithm, for the top-10 answers of each of the ten queries studied. In order to evaluate the quality of the ranking approach based on query-logs we use $\beta = 0$ for the linear combination. The judgements of the relevance of the answers to each query were performed also by people from our computer science

department. The figure shows that the proposed algorithm can significantly boost the average precision of TodoCL. For the top-2 answers the two algorithms behave similarly since TodoCL in general returns good answer at the top and users tend to click them. The improvement in the retrieval quality of the original algorithm becomes notorious when we consider answer sets of more than three documents, where the information carried by the clicks is very useful to discard documents that are of little interest to users.

For six of the ten queries studied, the proposed algorithm clearly outperformed the retrieval precision of TodoCL. In two of them the original algorithm performed slightly better. A possible explanation for this is that some relevant URLs returned by TodoCL is the first places of these queries had text descriptors that were not attractive to users.

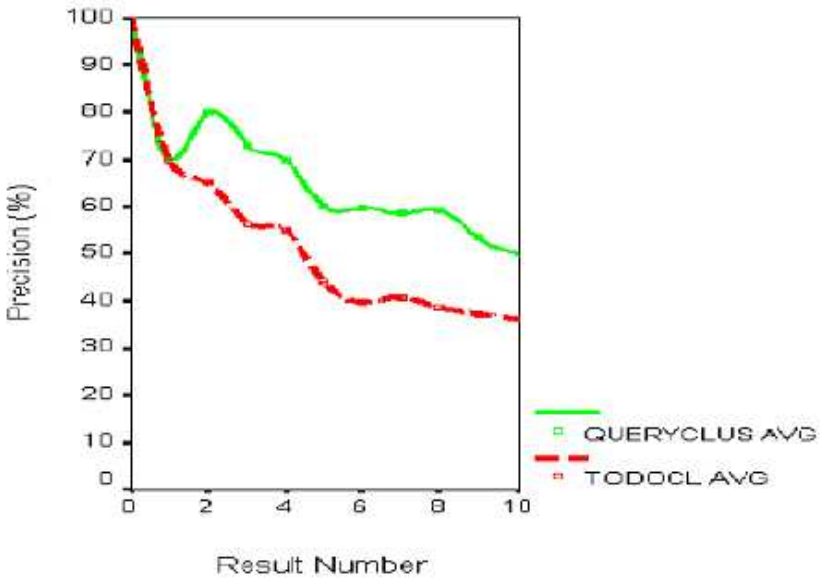


Figure 5.5: Average retrieval precision for the documents considered in the experiments.

In this section we have presented a method for suggesting related queries based on a clustering process over data extracted from the query log. The method can be identify related queries who not share common terms in the query space. The method could be easily used for query and document recommendation. The method proposed is simple and has reasonable computational cost for periodical preprocessing. Our experiments show that

it achieves good results in improving the retrieval precision of the popular search engine for the Chilean Web, TodoCL [Tod]. However four main problems arise: the evaluation function is proportional to the number of clusters so the evaluation is biased to the k value. Also we do not know the number of clusters. The second problem is a limitation of the clustering technique. K-means is sensitive to noise effects. Noise could be introduced by terms who are not related to the meaning of the documents, for example caused by *spamdexing* (see chapter 1). A third problem is related with the efficient of the clustering algorithm: k-means in general can be extremely slow. Finally another problem inherent to the use of query logs is that the clicked URLs are biased to the ranking algorithm of the search engine.

In the following section we will preform experiments with larger logs and also probing a new query session representation based on snippet terms. Finally we will work in how to reduce the bias induced by the position of the documents in the search engine interface.

5.2 The method based on snippets

In this section we present a query clustering process based on a term-weight vector representation of queries, obtained by aggregating the term-weight vectors of the snippet associated to the selected URL's for the query. The construction of the vectors includes a technique to reduce and adjust the bias of the preferences to URL's in answers.

Because the vectorial representation of a queries is obtained by aggregating term-weight vectors of the snippet of the documents, our method avoids the problems of comparing and clustering sparse collection of vectors, a problem that appears in chapter 3. The central idea in our vectorial representation of queries, is to allow manipulating and processing queries in the same fashion as document are handle in traditional IR systems, therefore allowing a fully symmetric treatment of queries and documents. In this form, query clustering turns into a similar problem to document clustering. The proposed method is basically an improvement over the method presented in the previous section, performing more extensive experimentation over the quality of the recommendations and defining a method to reduce de bias introducing in the user's clicks by the position effect,

We present two applications of our clustering method: query recommendation and answer ranking. The use of query clustering for query recommendation has been suggested by Beeferman and Berger [BB00], however as far as we know there is no formal study of the problem. Regarding answer ranking, we are not aware of formal research on the application of query clustering to this problem. For both applications we provide a criterion to rank the suggested URL's and queries that combines the *similarity* of the query (resp. URL) to the input query and the *support* of the query (resp., URL) in the cluster. The support measures how much the recommended query (resp. URL) has attracted the attention of users. The rank estimate the *interest* of the query (resp., URL) to the user that submitted the input query. It is important to include a measure of *support* for the recommended queries (resp., URL's), because queries (URL's) that are useful to many users are worth to be recommended in our context.

We next present the term-weight vector model for a query. Each snippet term is weighted according to the number of occurrences and the number of clicks of the documents in which the term appears.

Given a query q, and a URL u, let $\texttt{Pop}(u, q)$ be the number of clicks to u in the sessions related to query q. Let $\texttt{Tf}(t, u)$ be, as usual, the number of occurrences of term t in the snippet of the URL u. In this method we reduce the vocabulary to the terms presented in the snippet of the document because they are reviewing for the user before the document selection. Thus, this terms are closely related to the meaning behind the original query. Intuitively, we expect that this reduced set of terms could be avoid the noise effect of the method presented in the previous section.

Stopwords are eliminated from the vocabulary considered. We define the *vectorial representation of a query*, $\vec{q}$, in a similar way as was defining for the previous method, as follows:

$$\vec{q}[i] = \sum_{URLu} \frac{\texttt{Pop}(u, q) \times \texttt{Tf}(t_i, u)}{\max_t \texttt{Tf}(t, u)} \tag{5.4}$$

That is, Pop plays the role of *Idf* in the well-known tf-idf weighting scheme for the vector model.

5.2.1 The bias reduction method

Now we will present the method to reduce de bias introducing by the position effect. Assume that users submit queries and have plenty time to choose, among the whole set of URLs in the Web, the URLs that they like as answers to a query. In such a process we can easily estimate the relevance of a page. However, we need to estimate relevances from the query log. The main problem in doing so is in the fact that URLs are returned by search engines in some ordering, and their preferences, as registered in the log, are biased against such ordering. In the remainder of the section we present a method to estimate the relevance of a page to a query from the clicks of the page obtained from the logs.

First we define the following random process. When a user submit a query q, the URL u is returned at a position r in the answer ranking. Then the user with some probability denoted $P_{visit}(X \geq r)$ skips the process at some URL greater then r. If the user reaches the position r, then he/she selects the URL u with probability $P_{likes}(u|q)$, otherwise he/she does not select the URL u.

Given a query q and URL u, we denote by $\texttt{Rank}(u, q)$ the current rank of u in the answers of q. Let us denote by $P_{rank}(u|q)$ the probability a user selects u, given that she/he has submitted the query q, considering that u appeared in position $\texttt{Rank}(u, q)$ in the answer ranking. We have:

$$P_{logs}(u|q) = P_{visit}(X \geq \texttt{Rank}(u, q))P_{likes}(u|q).$$

Thus, in order to estimate $P_{likes}(u|q)$, we only need to estimate $P_{visit}(X > r)$, for each r. We posit that each successive position in the rank is less likely to be reached by users, with a power-law decay. We use a power-law because this is the distribution that appears in incoming-links [BKM+00] as well as in term frequency. In addition, the distribution of ranking of clicks in our data follows a power-law, although there are discontinuities due to the 10 answer-per-page output. More precisely, since $P_{visit}(x)$ gives the probability a user visit's is greater than or equal to x, it can be modeled using the following Pareto distribution:

$$P_{visit}(X \geq x) = \frac{1}{x^b} \tag{5.5}$$

Thus, we have:

$$P_{likes}(u|q) = P_{rank}(u|q)(\texttt{Rank}(u,q))^b.$$

The CDF of the Pareto distribution is $P_{visit}(X < r) = 1 - (1/r)^b$, and its PDF is $p_{visit}(X = r) = br^{-(b+1)}$, which is a power-law distribution. The latter represents the probability a user exits the process exactly at the position x. It remains to estimate b. We do so as follows. Given a query session we assume that the user exits at the largest position in the answer list of a clicked URL. Then we fit a power-law distribution to the histogram for the frequencies of estimated exit positions. Figure 5.6 shows a plot of the logarithm for the rank of last URLs clicked in query sessions versus the logarithm for the number of users. We fitted a power-law distribution over this data, which leads to $b = 1.725$.

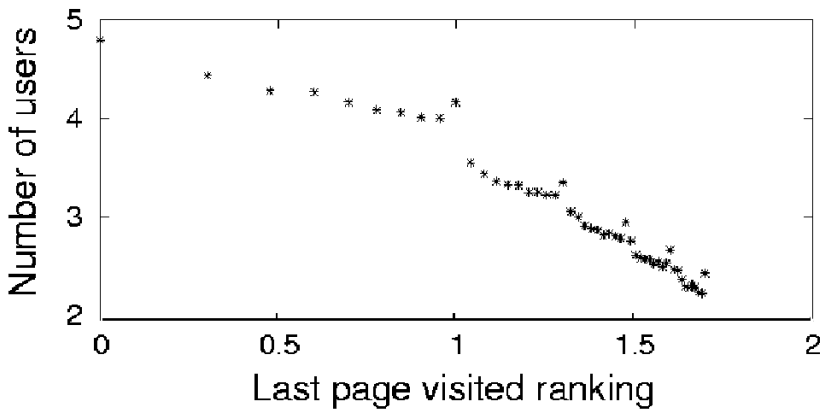


Figure 5.6: Plot of the logarithm for the Rank of last URLs clicked in query sessions versus the logarithm for the number of users.

Finally, the adjusted popularity is as follows:

$$\texttt{AdjPop}(u,q) = \texttt{Pop}(u,q)(\texttt{Rank}(u,q))^b \tag{5.6}$$

In the unbiased similarity function, we replace $\texttt{Pop}(u, q)$ by $\texttt{AdjPop}(u, q)$ in the vectorial representation given in Formula 5.4.

5.2.2 Experimental Results

In order to evaluate our method, we performed experiments over the L_6 log file. We selected the 30.363 queries with more the one session, which yield 147.891 sessions. This sessions have 431.432 clicks in total, which are over 131.924 documents.

Section 5.2.3 shows experiments for the clustering process for queries and their vector representation extracted from L_6. Section 5.2.4 shows experiments for the comparison of the distance functions proposed with other distance functions proposed in the literature. The selected clustering presented in section 5.2.5 is used to evaluate the answer ranking and query recommendation applications using 30 selected queries.

5.2.3 Clustering Process

In this section, we study the clustering process for the vectorial representations of queries given in Section 5.1 over a 6-month log. After removing stop-words we reach 98370 terms for the vectorial representation of the queries. We consider the cases with and without bias reduction. Figure 5.7 shows the quality of the clusters found for different values of k (recall that k is the number of clusters found). The quality is calculated as the value of the criterion function for this cluster. The figure display the clustering quality for the similarity function given in Section 5.2 (labeled snippet term) and the similarity function given in Section 5.2.1 (labeled unbiased snippet term). The qualities displayed by the curves of Figure 5.7 are not comparable since they are based on different similarity function. However, the curves show a similar behavior, where a plateau is reached around $k = 600$ (number of clusters). Table 5.1 shows statistics for the clusters found for $k = 600$ for the 6-month log. The table shows an overall cluster quality (Equation 5.3) of 14900, and average internal similarity between queries in each cluster of 0.2596 for the clustering based on snippet terms and an overall cluster quality of 14900 and average internal similarity between queries in each cluster of 0.2614 for the clustering based on unbiased snippet terms.

Figures 5.8 (A) and (B) show histograms for the number of queries per cluster for

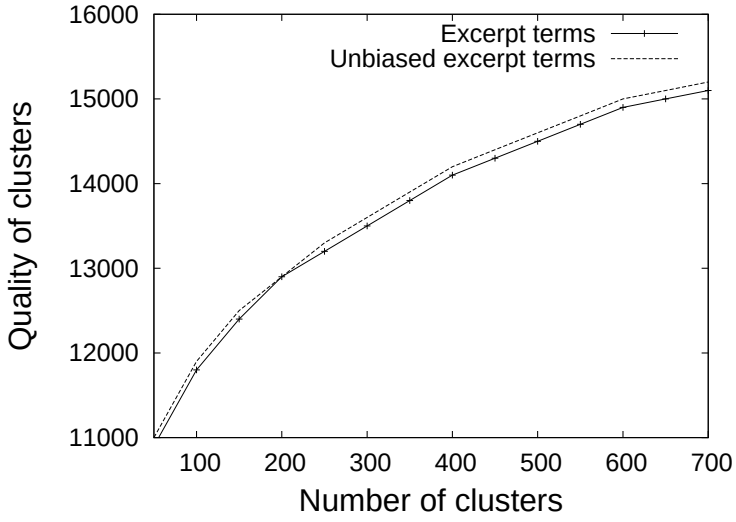


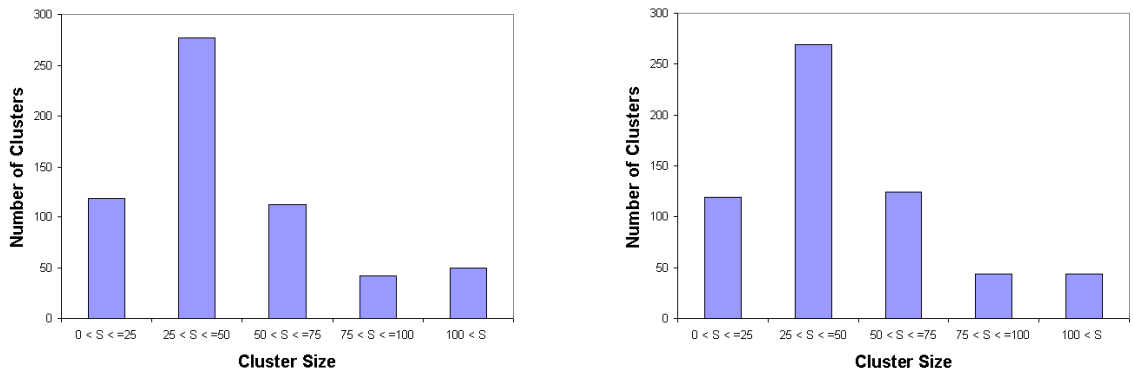
Figure 5.7: Quality of clusters vs. number of clusters.

	Exc. Terms	Unb. Exc. Terms
Overall Cluster Quality	14900	14900
Internal Similarity (Avg)	0.2596	0.2614
Internal Similarity (StDev)	0.0678	0.0678
External Similarity (Avg)	0.0512	0.0514
External Similarity (StDev)	0.0186	0.0188

Table 5.1: Statistics for the clusters found for $k = 600$

$k = 600$. The solution shows an adequate distribution of queries per cluster. The average number of queries per cluster is 51 for both solutions, with standard deviations of 34 and 35. For both solutions (i.e. over the 1200 clusters calculated) there are only 121 cluster with less than 20 queries. Moreover there aren't clusters with less than 10 queries. Thus we will use for the experiments the solution calculated for $k = 600$ because it represents a good tradeoff between the quality of the clusters and the size of the clusters.

We ran the algorithms on a Pentium IV computer, with CPU clock rate of 2.4 GHz, 1024MB RAM, and running Fedora 4. Figure 5.9 shows the running times of the clustering processes for the proposed similarity functions.



(A) (B)

Figure 5.8: Histogram of the number of queries per cluster for the (A) snippet terms based method and the (B) unbiased snippet terms based method.

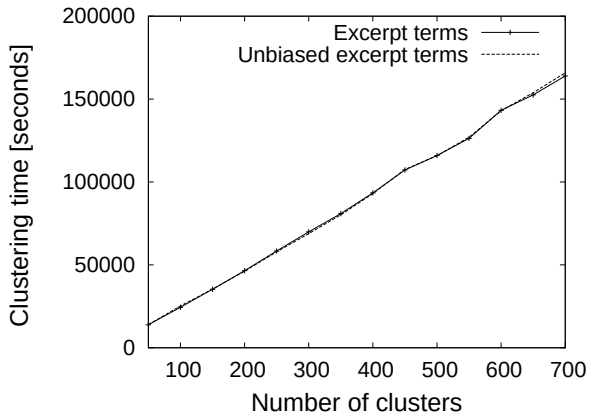


Figure 5.9: Running Time for the Clustering Processes.

5.2.4 Distance functions comparison

To determine the dependence between the proposed distance functions, we calculate distance matrices between 500 queries, selected by their popularity. In order to perform a more extensive comparison, we consider two additional distance functions introduced in [BB00]: a) Queries represented by their terms using a `TF-IDF` weighting schema and b) Queries represented by document co-citation, calculated over the collection of selected documents.

We perform a *Mantel's test* to compare the distance matrices. Let A and B be two distance matrices of $N \times N$ elements. We calculate a statistic given by:

$$Z = \sum_{i=1}^{N} \sum_{j=1}^{N} A_{i,j} B_{i,j}. \tag{5.7}$$

If the two distances are small in the same places, and big in the same places the value of Z will be large indicating a strong link between the distance functions. To test the significance of such a link, a permutation test is performed. Using a Monte Carlo method, the elements of one of the distance matrices is permuted while the other is hold constant. Each time the value of Z is recalculated. Finally a P-value is obtained comparing the test statistic with each recalculated value of Z.

For the experiment we used 1000 permutations. In table 5.2 we show the Z and P values for the experiment. In the third column we also show the Pearson Correlation Coefficient. As the experiment results show, there is a significant dependence between the distance pair based on Query Terms and based on Co-citation and there is a moderated correlation between the pair of proposed functions. This is intuitive: documents are retrieved because they contain query terms so it stands to reason the correlation of co-citation and query terms would be high. Dependencies and correlations between the other pairs are not significant.

	Z Value	P Value	Pearson Correlation
Query terms vs Co-citation	4914	0.000998	0.905293
Query terms vs snippet terms	4558	0.000996	0.365039
Query terms vs Unb. snippet terms	4501	0.000996	0.365535
Co-citation vs snippet terms	4570	0.000999	0.391686
Co-citation vs Unb. snippet terms	4512	0.000999	0.390907
snippet terms vs Unb. snippet terms	4506	0.000998	0.650991

Table 5.2: Distance functions comparison based on a Mantel's test.

In figure 5.10 we show histograms for the distances calculated over the query collection. The histograms for the proposed distance functions generate a strong differentiation between the queries. Figure 5.10(C) shows that the 63% of the distances are less than 0.95

for the representation based on snippet terms. Figure 5.10(D) shows that over the 70% of the distances are less than 0.95 for the representation based un unbiased snippet terms. On the other hand, figures 5.10(A) and 5.10(B) show that for the distance functions based on query terms and co-citation, over the 95% of the distances belong to the majority class, very close to the maximum value.

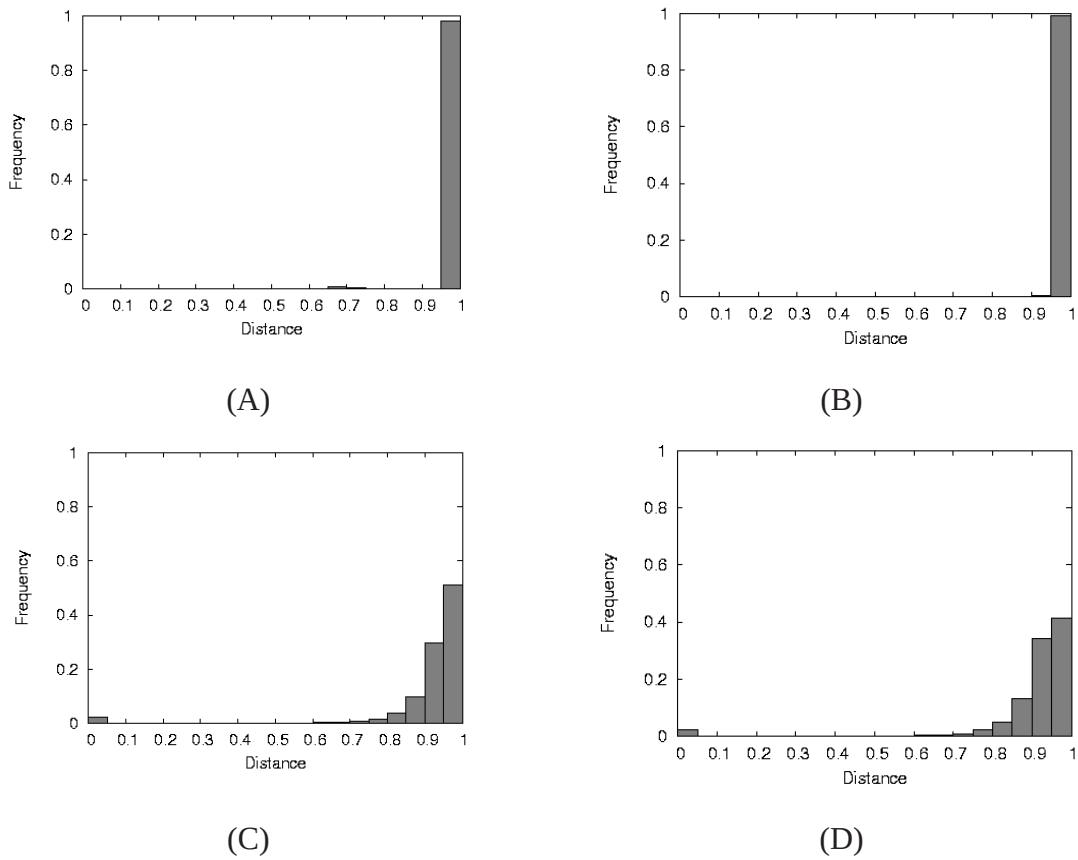


Figure 5.10: (A) Histogram of the distances for the query terms based function. (B) Histogram of the distances for the co-citation based function. (C) Histogram of the distances for the snippet terms based function. (D) Histogram of the distances for the unbiased snippet terms based function.

Finally to compare each proposed distance function we use a set of 600 pairs of quasi-synonym queries. For each pair of queries and for each function we calculate the distance between them. As the pairs of queries are very close semantically, we expect to obtain small

distance values. In table 5.3 we show the results for each distance function. The results are shown by deciles. Each element of the table represent the percentage of the set of queries pairs that belong to the decile. In table 5.4 we show the medians for each decile and for each distance function. In 5.5 we show a table that illustrates the experiment. For 20 selected pairs of queries and for each distance function, we show the distances calculated and the associated percentile. Rows are sorted by distance, in the following order: query terms distance function, co-citation distance function, snippet and unbiased snippet term distance function. Values in bold fonts represent the better results for the query pair comparison. As table 5.5 shows, in 9 of the 20 pairs the better result is achieved with the proposed functions. Instead of in the remaining comparisons the better results are achieved by the distance functions based on query terms or co-citation, the proposed functions achieve good results, having low percentiles in the distance distribution. On the other hand, the first 7 comparisons are very disappointed for the distance functions based on query terms or co-citation. They classify the pairs in the worst percentile of the distribution. The reason why is that the distance distribution based on query terms or co-citation basically achieve a binary separation in the data as the histograms 5.10 show. Thus, most of the pairs have distance 1. In the co-citation distance distribution this quantity is close to the 85% of the pairs and in the query terms distance distribution is close to the 90% (see table 5.3). The remaining pairs are classified into the first percentiles of the distance distributions (showed in the last rows of the table 5.5).

	d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8	d_9	d_{10}
Query terms	20.7	0	0	0	0	0	0	0	0	79.3
Co-citation	53	0	0	0	0	0	0	0	0	47
snippet terms	68.5	9.5	4.7	0.2	9.5	2.8	4.7	0.1	0	0
Unb. snippet terms	69	9.7	8.8	4.5	3.5	4.2	0.3	0	0	0

Table 5.3: Distance distributions for a set of 600 pairs of quasi-synonym queries. Results are shown in percentages.

	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	m_9	m_{10}
Query terms	0.95	1	1	1	1	1	1	1	1	1
Co-citation	0.95	1	1	1	1	1	1	1	1	1
snippet terms	0.575	0.85	0.875	0.9	0.911	0.925	0.95	0.975	1	1
Unb. snippet terms	0.575	0.8	0.825	0.85	0.875	0.9	0.925	0.95	1	1

Table 5.4: Medians for each decile of the distance distributions shown in table 5.3.

5.2.5 Evaluation of recommendation algorithms

We will evaluate the applications proposed in the previous section but using the new query clusterring method. In order to do this we consider a study of a set of 30 randomly selected queries. The 30 queries were selected following the probability distribution of the 30363 queries of the 6-month log. In table 5.6 we show the selected queries for the experiments and some descriptive features of their clusters calculated using the unbiased snippet terms based solution for $k = 600$. The results are sorted by the cluster rank, that indicates the quality of the cluster.

The first column of Table 5.6 shows the selected queries. The second column gives the rank of the clusters, according to their quality. The third column shows the cluster quality (the inner quality value). The fourth column shows the cluster size. The last column depicts the set of feature terms that best describe each one of the clusters. We have translated the terms in the original language of the search engine. Right next to each feature term, there is a number that is the percentage of the within cluster similarity that this particular feature can explain. For example, for the cluster of the query *used notebooks* the feature ”compaq” explains the 41% of the average similarity of the queries in the cluster. Intuitively, these terms represent a subset of dimensions (in the vectorial representation of the query traces) for which a large fraction of objects agree. The cluster objects form a dense subspace for these dimensions. Only the more relevant feature terms are showed in the table.

Our results show that many clusters represent clearly defined information needs of search engine users and reflect semantic connections between queries which do not share

Queries	Query t.	$P\%$	Co-cit.	$P\%$	snippet t.	$P\%$	Unb. Exc. t.	$P\%$
houses - properties	1	100	1	100	0.899	42	**0.874**	**25**
Chilean music - cueca	1	100	1	100	**0.869**	**19**	0.852	23
hospitals - clinics	1	100	1	100	0.913	47	**0.882**	**36**
crashed cars - dismantlers	1	100	1	100	0.958	61	**0.916**	**59**
games - toys	1	100	1	100	0.791	8	**0.735**	**6**
Chilean food recipes - typical food	1	100	1	100	0.861	19	**0.825**	**12**
movies - films	1	100	1	100	0.741	5	**0.642**	**4**
tarot - horoscope	1	100	0.981	12	0.872	20	**0.771**	**7**
automobile dealers - car sales	1	100	0.971	9	0.597	4	**0.549**	**3**
estate agency - construction companies	1	100	0.967	8	**0.449**	**3**	**0.411**	**3**
work - job	1	100	**0.958**	**4**	0.822	9	0.811	11
jokes - humor	1	100	**0.956**	**4**	0.766	8	0.749	7
cars - automobiles	1	100	**0.944**	**1**	0.77	8	0.71	5
pubs - bars	1	100	**0.925**	**1**	0.701	5	0.604	4
tires - wheel rims	1	100	**0.916**	**1**	0.556	3	0.584	4
plans - maps	1	100	**0.874**	**1**	0.44	3	0.51	3
work offers - job offers	**0.591**	**1**	0.966	8	0.388	3	0.323	3
chords - songs with chords	**0.422**	**1**	**0.761**	**1**	0.318	3	0.372	3
travel agencies - tourism agencies	**0.333**	**1**	**0.869**	**1**	0.408	3	0.406	3
mp3 - free mp3	**0.292**	**1**	**0.894**	**1**	0.231	3	0.357	3

Table 5.5: Distance functions comparison based on quasi-synonym queries sorted by distance. The columns labeled with $P\%$ indicate the percentile in the distance function distribution

query words. As an example, the feature term *brakes* of the cluster of query *tyres* reveals that users that search for web sites about tyres, also search for information about brakes. Probably, some sites containing information about tyres contain references to brakes. Another example is the term *restaurant* related to query *food home delivery*, which shows that users searching for *food home delivery* are mainly interested in restaurant information. These examples, and many others found in our results, showed the utility of our framework for discovering information needs related to queries.

5.2.6 Query Recommendation

In order to asses the quality of the query recommendation algorithm for the thirty queries given in Table 5.6, we follow a similar approach to Fonseca *et al* [FGDMZ03]. The relevance of each query to the input query were judged by 20 members of ours Computer Science Departments. They analyzed if the answers of the queries are of interest to the input query. Each query was evaluated through 5 levels of relevance. Then, the relevance of each item was calculated as the average among all the judgements given by the expert group. Every expert evaluated all the queries.

Our results are given in graphs showing precision vs. numbers of recommended queries. Figure 5.11 shows the average precision for the queries considered in the experiments. For the methods based on co-citation and query terms the scores are calculated using the popularity of the queries. In average, with the proposed methods we obtain a precision of 75% for the first ten recommended queries. Therefore, the suggested queries are relevant to users that submitted the original queries. Our results also show that the rank schemas proposed are better than the scores obtained by considering only the popularity of the queries in the cluster that are recommended using co-citation or query terms. Finally, the method based on unbiased snippet terms is the best of the methods considered in the experiments.

5.2.7 Answer Ranking

We compared our ranking algorithm with the algorithm provided by the search engine for the thirty queries given in Table 5.6. The proposed ranking algorithm was performed using $\beta = 0$. The ranking algorithm of the search engine is based on a belief network which

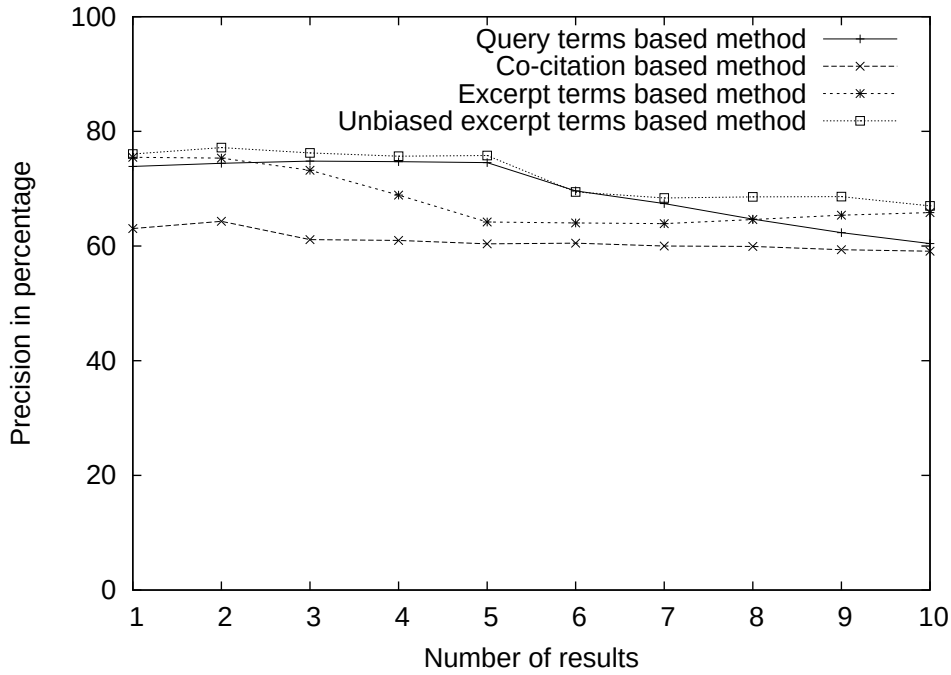


Figure 5.11: Average retrieval precision of the query recommendation algorithms.

is trained with links and content of Web pages, and does not consider logs. The top-10 answers of the queries studied were considered in the experiment. The judgments of the relevance of the answers to each query were performed by people from ours Computer Science Departments.

Figure 5.12 shows the average retrieval precision of the search engine and the proposed algorithms for answer ranking using both methods. The graph shows that our algorithm can significantly boost the average precision of the search engine. For all the queries studied in the experiment our algorithm outperforms the ranking of the search engine. Over the top-10 results the average precision of the proposed method is approximately 65%. The original rank has only an average precision of 50%. Over the top-5 results the difference is more significant. Our methods are close to a precision value of 75% while the original rank is close to the 60%. Finally the figure shows that the method based on unbiased snippet terms is better than the other two methods.

For the 300 documents evaluated by users, we show in figure 5.13 a scatter plot of

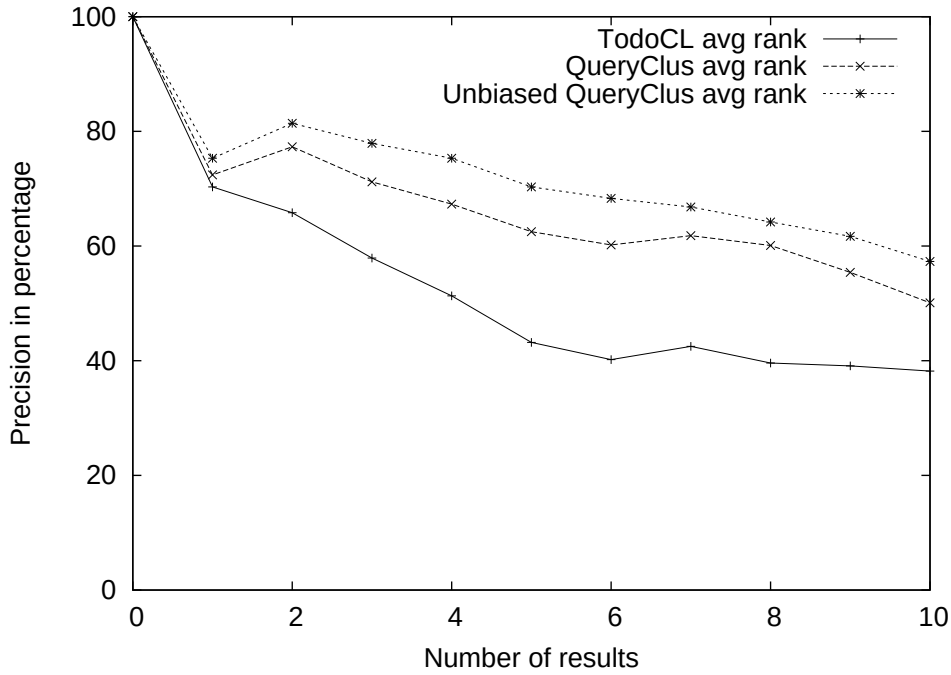


Figure 5.12: Average retrieval precision of the proposed ranking algorithm.

the ranking based on unbiased snippet terms and the original rank. As the plot shows, our method has a low degree of dependence with the original method. In fact the Pearson coefficient of both rankings is only 0.3639. The graph could be interpreted as follows: for the first recommended documents by our method, the original rank is in the 1 - 6 range. Some of the first recommendations have original positions in the last 20 or 30 places. This is a direct consequence of the unbiased method.

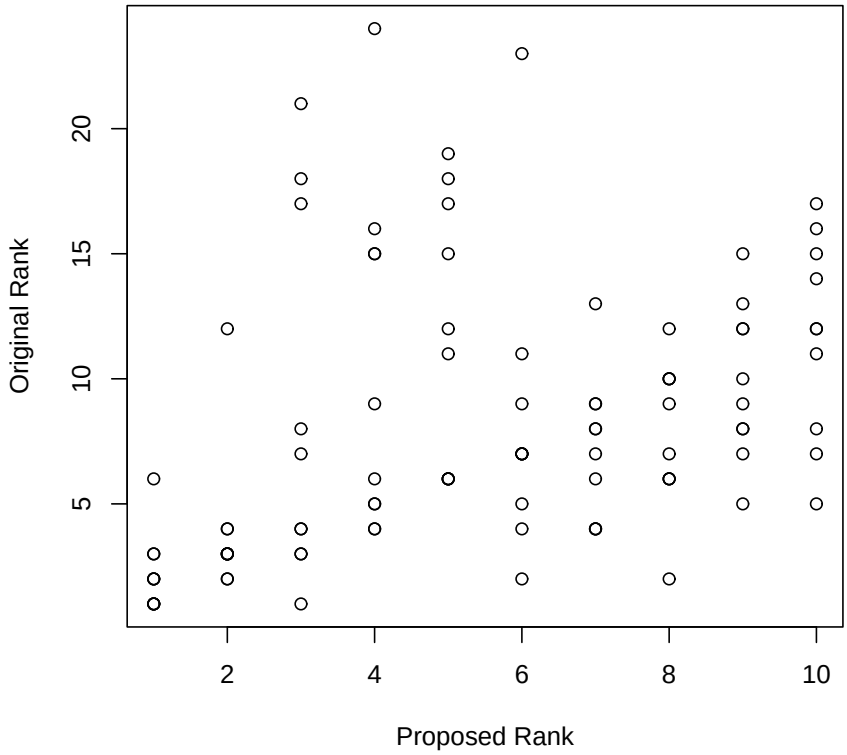


Figure 5.13: Scatter plot of the original ranking versus the proposed ranking for the 300 documents considered in the experiments.

5.3 Conclusion

We have proposed two methods that allow us to find groups of semantically related queries. Our experiments show that the bias reduction technique proposed improves the quality of the clusters found. It is possible to conclude also that the method based on snippet terms reduce the noise associated to broad and imprecise vocabularies as the used in the formulations of documents. The results also provide enough evidence that our ranking algorithm improves the retrieval precision of the search engine, and that our query recommender algorithm has good precision in the sense that it returns relevant queries to the input query.

The notions of query similarity we propose has several advantages: they are simple and easy to compute; they allow to relate queries that happen to be worded differently but stem from the same user need; they lead to similarity matrices that are much less sparse than matrices based on previous notions of query similarity; finally, the vectorial representation of queries we propose yields intuitive and useful feature characterizations of clusters.

Query	Cluster Rank	Cluster Quality	Size	Descriptive Terms
bikinis	1	0.863	11	swimsuits (2%)
Chilean typical food	30	0.315	45	typical (9%) south (4%)
Kino	46	0.278	33	*Polla* (24%) *Iman* (19%) *Loteria* (5%)
embassy of *Canada*	67	0.261	27	embassies (42%) consulates (18%)
divorce law	68	0.260	49	marriage (40%) law (12%) married couple (5%)
Chilean chats	73	0.248	43	nick (2%) chat rooms (2%)
swimming pools	76	0.244	45	swimming pool (60%) Jacuzzi (2%)
plots	94	0.241	74	sales (21%) rentals (17%) properties (12%)
second hand laptops	95	0.228	34	*Compaq* (41%) *Toshiba* (7%)
used cars Chile	100	0.226	93	cars (41%) automobiles (9%) vehicles (4%)
lemon pie	117	0.221	57	catering service (4%) cakes (6%)
rental	119	0.219	56	properties (47%) apartments (10%)
rentals in *Iquique*	119	0.219	56	properties (47%) apartments (6%)
prefabricated homes	126	0.217	30	intermodal (16%) houses (11%)
Yellowpages	135	0.209	50	*Publiguias* (38%) pages (17%)
applications	155	0.206	55	judicial (19%) appeal (18%)
ministries	164	0.2	166	law (35%) incise (11%) ministry (3%)
Harry Potter	188	0.190	86	*Papelucho* (38%) literature (2%)
V Region cabins	205	0.180	37	*Quisco* (32%) *Algarrobo* (12%)
Via del Mar	226	0.177	76	*Valparaiso* (75%) guest house (2%)
tires	228	0.173	55	brakes (41%) batteries (6%)
spare pieces	247	0.170	74	accesories (48%) auto body shop (2%)
INE	269	0.164	28	*IPC* (25%) statistics (11%)
florist shop	304	0.153	34	flower (21%) roses (17%) lower bouquets (14%)
emotional Intelligence	351	0.148	51	*Reiki* (13%) *Yoga* (8%)
bars	390	0.142	81	panoramas (12%) gastronomy (5%)
steam cycle	443	0.139	70	weather (12%) mountain chain (7%)
food home delivery	481	0.120	92	food (41%) dishes (12%) restaurants (8%)
software	543	0.102	82	software (42%) *Windows* (18%)
Chile post office	578	0.095	130	companies (7%) market (5%)

Table 5.6: Queries selected for the experiments and the cluster to which they belong for the solution obtained using the unbiased snippet terms based method for $k = 600$. Results are sorted by their cluster rank. Proper nouns are showed in italics.

Chapter 6

Query Classification Methods

In this chapter we study how to classify a query in a given taxonomy. The motivation is to identify well defined concepts behind Web queries. To do this, we use Web directories, classifying the original query in a category of the directory. After this, we finalize the chapter, introducing a method to keep directories updated. To evaluate the success of our approaches, we perform experiments using user's click data.

This chapter has been partially published in [CHM06].

6.1 The classification method

6.1.1 Preliminaries

Directories are hierarchies of classes which group documents covering related topics [BYRN99]. Directories are compounded by nodes. Each node represents a category where Web resources are classified. Queries and documents are traditionally considered as Web resources. The main advantage of the use of directories is that if we find the appropiate category, the related resources will be useful in most of cases [BYRN99].

The structure of a directory is as follows: the root category represents the *all* node. The all node means the complete corpus of human knowledge. Thus, all queries and documents are relevant to the root category. Each category shows a list of documents related to the category subject. Traditionally, documents are manually classified by human editors.

The categories are organized in a child/parent relationship. The child/parent relation can be viewed as an ”IS-A” relationship. Thus, the child/parent relation represents a generalization/specialization relation of the concepts represented by the categories.

It is possible to add links among categories without following the hierarchical structure of the directory. These kind of links represent relationships among subjects of different categories in the directory which share descriptive terms but have different meanings. For example, in the TODOCL directory [Tod], the category *sports/aerial/aviation* has a link to the category *business & economy/transport/aerial* because they share the term *aerial* that has two meanings: in one category the term is related to sports and in the other to transportation.

A relevant property of web directories is inheritance. According to Chakrabarti [Cha03] we will use the following definition of *inheritance*: If a category c_0 is the parent of a category c_1, any web item that belongs to c_1 also belongs to c_0.

If we understand a parent/child relation as an ”IS-A” relationhip, any web item that belongs to a descendant of a given category represents a *specialization* of its meaning. Conversely, any web item is related to the meaning of the parent categories and the relationship among them represents a *generalization*.

A hierarchical classification method should consider in its design a minimal consistency between the classification rules and the taxonomy structure. We state the following principle from the consistency of a hierarchical classification: Let c be a category in a taxonomy τ and q be a query semantically related with τ. If q is classified into c, the classification is consistent with τ only if q was classified in all the parents of c.

We can generalize the principle introduced above as a generic hierarchical classifier. Let Λ be a hierarchical classifier and τ be a taxonomy. Λ is consistent with τ only if for each query q classified into τ under Λ, the classification satisfies the consistency principle in hierarchical classification.

As it was presented in the state of the art chapter, frequently a classification schema is worked out following a flat approach. A flat approach classifies the Web resource into the nearest category using a distance function, without any constraint related to the hierarchical structure of the taxonomy. The main problem of flat models is the violation of the consistency principle. In general a flat model does not guarantee the consistency principle

because it follows a simple rule of minimum distance.

In the following section, we present a method based on hierarchical classification according to the Consistency Principle in Hierarchical Classifiers.

6.1.2 The classification method

Search engines show their recommended documents to queries as lists of items where each item is formed by the document URL, the title and a snippet. The snippet expresses the section of the document which is closer to the query. Intuitively, if the snippet is semantically related to the user query, then the user will select the document.

As in the previous chapter, we will consider the terms of the snippets to build a term-weighted vectorial representation. In order to do this, we measure the relevance of each snippet considering another variant useful for our representation: the time spent in each document visit. We will use the following assumption: the more relevant the document is, the longer the user will spend visiting it. Finally, we will also include query terms. For our representation the query is another selected document where its meaning is expressed by the query terms.

Now we will formalize the representation. Given a query session s, let q be the query associated to s, and let U_s be the set of documents selected in s. For a document u in U_s, let $Tf_{t,q}$ and $Tf_{t,u}$ be the number of occurrences of the term t in the query q and in the document snippet of u, respectively. We built our representation from the document snippets in U_s and the query q considering a $tf - idf$ scheme proportional to the time t_u spent in each document selected, and normalized by the total time t_s in the session s.

Let v_s be the term vector for a query session s, where $v_s[i]$ is the *i*-th component of a vector associated to the *i*-th term of the vocabulary. The *i*-th component $v_s[i]$ of the vector is defined as follows:

$$\begin{aligned} v_s[i] &= \left(0,5 - 0,5\frac{Tf_{i,q}}{max\,Tf_q}\right) \times \log\frac{N_Q}{n_{i,Q}} \times \frac{1}{\mid U_s \mid} \\ &+ \sum_{u \in U_s} \frac{1}{\mid U_S \mid} \times \frac{Tf_{i,u}}{max\,Tf_u} \times \log\frac{N_U}{n_{i,u}} \times \frac{t_u}{t_s}, \end{aligned} \tag{6.1}$$

where Q is the query collection (the set of queries formulated and registered in the logs), N_Q is the number of queries in Q, $n_{i,Q}$ is the relative frequency of the *i*-th term in Q (the number of queries in Q where the *i*-th term appears), U is the document collection (the set of selected documents registered in the logs), N_U is the number of documents in U, and finally $n_{i,u}$ is the relative frequency of the *i*-th term in the document set U (the number of documents in U where the *i*-th term appears).

The first half of the equation represents the weight of the *i*-th term for the user query. We use the schema proposed by Salton and Buckley in order to avoid a sparse term vector. The second half of the equation is a sum of the weights of the *i*-th term for each selected document in the session. Each weight is normalized with the time spent by the user in the visit.

To calculate this representation, we retrieve the snippet for each pair query-document. Snippet terms are processed in order to eliminate *stopwords*. Visit times are also calculated from the user's click data.

Similarly, for a category c, we obtain a term vector representation v_c, by aggregating the text of every snippet that appears in the category. This text comprises the descriptive text of the category and the snippets of the documents listed in the category.

Each query will be classified following a top-down approach. First, we will determine the closest centroid to the query considering all the centroids at the first level of depth in the concept taxonomy. Then we repeat the process for each level of the taxonomy while the distance between the query and the closest centroid will be less than the distance at the previous level. The top-down approach is used to avoid the noise effects introduced by document and query terms. From our point of view, the term relevance decreases from general topics to sub-topics. In figure 6.1 we illustrate the classification schema.

Now, we formalize the method. Let q be a query in a collection C of queries registered in the logs and $c_{i,j}$ be the i-th category in the j-th level of a web taxonomy τ. For the root of the web taxonomy, the nearest category to the query is determined. Let $c_{*,0}$ be the closest category to q at the root level of τ and $\Gamma(c_{*,0})$ be the children set of $c_{*,0}$. In the next iteration of the method the classifier calculates the distances between q and each category in $\Gamma(c_{*,0})$. Then the nearest category in $\Gamma(c_{*,0})$ is determined. Let $d_{min}(q, c_{*,1})$ be the distance between q and the closest category in $\Gamma(c_{*,0})$ and $d_{min}(q, c_{*,0})$ be the distance between q an

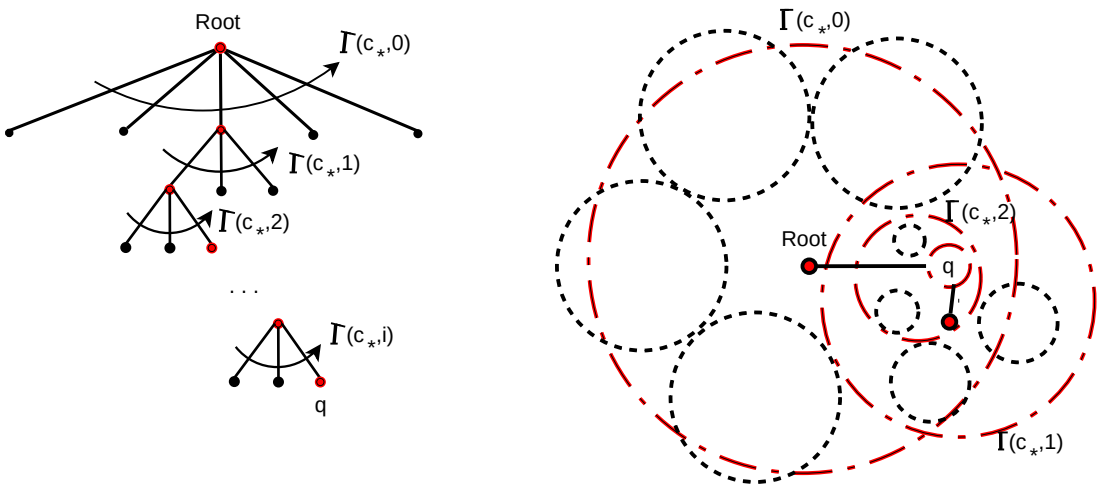


Figure 6.1: The hierarchical classification schema proposed

the closest category at the root level of τ. If $d_{min}(q, c_{*,1}) < d_{min}(q, c_{*,0})$ then q is classified in the closest category of the first level of τ. Then, in the next iteration of the method, the classifier calculates the distances between q and each category in $\Gamma(c_{*,1})$. Otherwise, the method stops.

6.1.3 Experimental results

In order to evaluate the proposed method, the following experiments will be carried out. First of all, we classify 33,163 queries in the taxonomy of TodoCL [Tod], using the L_6 log file from the TodoCL search engine, the same data set that was used for the experiments on the chapter of query clustering.

We intend to illustrate with examples that the clasification method has the ability to identify concepts related to the query. To do this, we randomly select 30 queries from the total. On the 30 queries, we will carry out experiments that allow to evaluate the appropriateness of the classification and its usefulness for the user. Table 6.1 shows the 30 queries considered in our experiments, the taxonomy nodes where they are classified and the distance between the vectorial representations of the initial query and the node.

The hyphen indicates specialization in the taxonomy. For example, the *art- music-midi* node shows that the query about Chilean music history was classified in the art topic, music

section and in the midi topic within music. As we can see, each selected query is related to a well defined concept that describes one possible meaning. None of the selected queries are false positive regarding the classification goal.

As a second experiment, we perfom an expert evaluation of the hierarchical classifier compared with a flat classifier. We will use for this comparison: a flat classifier based on minimum distance. In order to evaluate the quality of the classification approaches, we will compare both classifiers using the same distance function. To do this, we use the function proposed in the equation 6.1.

A group of experts evaluated the relevance of the node where the query is classified according to its meaning. We presented to 19 persons of different backgrounds the thirty queries and their categories. We asked one question to the participants: Is the category relevant to the query? The answer could be expressed using relevance degrees varying from 0 - 4, from lower to higher relevance. Subsequently, we calculated the average relevance of every query on every opinion expressed by the users. The distribution of the opinions over the 5 possible values is shown in Figure 6.2.

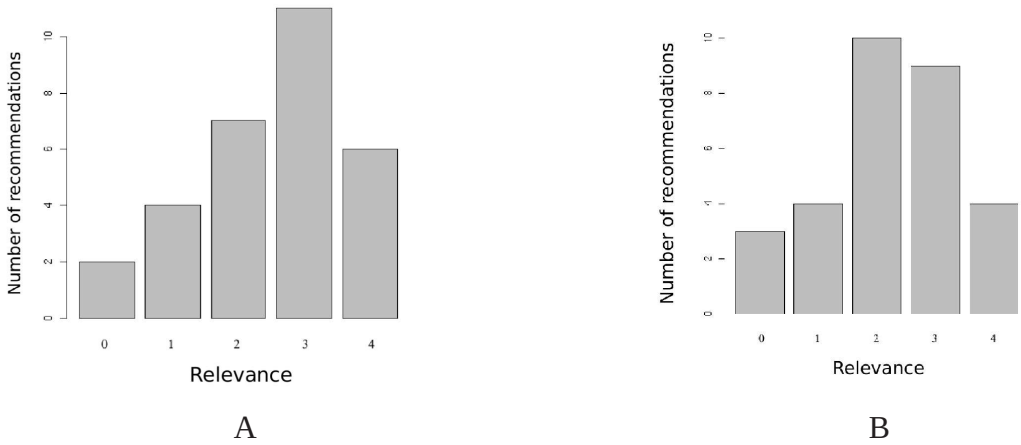


Figure 6.2: User opinions for the experiments based on query taxonomy classification. Figure A) shows the results for the hierarchical classifier and Figure B) shows the results for the flat classifier.

Figure 6.2-A shows that over the 70% of the recommendations receive a good evaluation from the users (relevance is greater than or equal to 2). Moreover, almost 50% of the recommendations (relevances with values 3 or 4) improve the initial query. For this method,

Query	taxonomy node	distance
Romane ratings	travels tourism	0,99
interactive museum Mirador	education	0,988
yoga	health	0,978
work environment complaints	government	0,974
Patricio Del Canto	arts + museums and cultural centers	0,973
Sinergia	arts + music + bands artists	0,973
Francisco Moya	arts + galleries	0,969
Metalcon foundation blocks	economy and businesses + industries + forests	0,967
jewelry lessons	arts	0,963
signs publication	arts + graphic arts	0,952
rolls of grass	home + gardening	0,948
clothing projects	econony and businesses + textiles + clothes	0,947
southern road	tourism travels	0,942
financial concepts	economy and businesses + finances	0,92
law 14,908	society + family	0,917
Metal furniture	home	0,895
shows	arts and entertainment + audiovisual production	0,877
X region companies	economy and businesses	0,871
educational evaluation issues	education	0,87
houses for sale in Iquique	regions + geographic zones	0,868
companies in Chile	guides directories	0,855
transpersonal psychology	health + psychology	0,85
hosting	guides directories + portals	0,822
furniture sales	home	0,818
Antofagasta Clinic	health + clinics and hospitals	0,815
satellite telephony	economy and businesses + telecommunications	0,815
PSU results	education + university selection test	0,814
Chilean music history	arts + music + midi	0,796
properties	economy and businesses + estate agencies market	0,761
sanitary engineering	economy and businesses + environment	0,756

Table 6.1: Queries selected for the evaluation of the method of query classification in directories sorted by distance.

the majority class is associated to the value 3. Figure 6.2-B shows that expert opinions are less expressive compared with the results obtained by the hierarchical classifier. Only 40% of the recommendations (relevances with values 3 and 4) improve the initial query. For this method, the majority class corresponds to the value 2.

A third experiment consisted of evaluating the quality of the answer lists retrieved by the search engine ranking method, the method based on the hierarchical classifier, and the method based on the flat classifier. In order to do this, we have considered the first 10 documents recommended by the search engine for each one of the 30 queries, the first ten documents recommended by the web directory when the closest category is determined using the flat classifier, and the first 10 documents recommended by the web directory when the query is classified using the hierarchical method. The document quality has been evaluated by a group of experts using the same relevance criteria as in the previous experiment (0-4, from lower to higher relevance). The precision for every ranking and every query is obtained, according to the position. Finally, the average precision is calculated over the total of documents recommended by position. Results are shown in Figure 6.3.

In figure 6.3 we can observe that all the evaluated methods are good quality rankings, especially for the first 5 recommended documents. The recommendation methods based on hierarchical classification and flat classification perform in a better way for the first 5 positions than the TodoCL ranking. This means that the classification in the taxonomy is a good quality one, the same as shown in the previous experiment. However, the ranking loses precision compared to the one of TodoCL, if we consider the last 5 positions. This is due to the fact that many of the evaluated queries are classified in taxonomy nodes where less than 10 documents are recommended. In these cases, since there is no recommendation, the associated precision equals 0, which severely disqualifies the methods based on classifiers. Fortunately, none of the queries is classified in a node with less than 5 recommended documents. Therefore, a fair comparison of the methods should be limited to the first 5 positions where, as we have seen, the hierarchical method is favorably compared with the original ranking and with the flat classifier. Due to the fact that in general the coverage of directories is low, i.e. there are few documents recommended in each node of the directory, it is necessary to design a maintenance method in order to classify documents

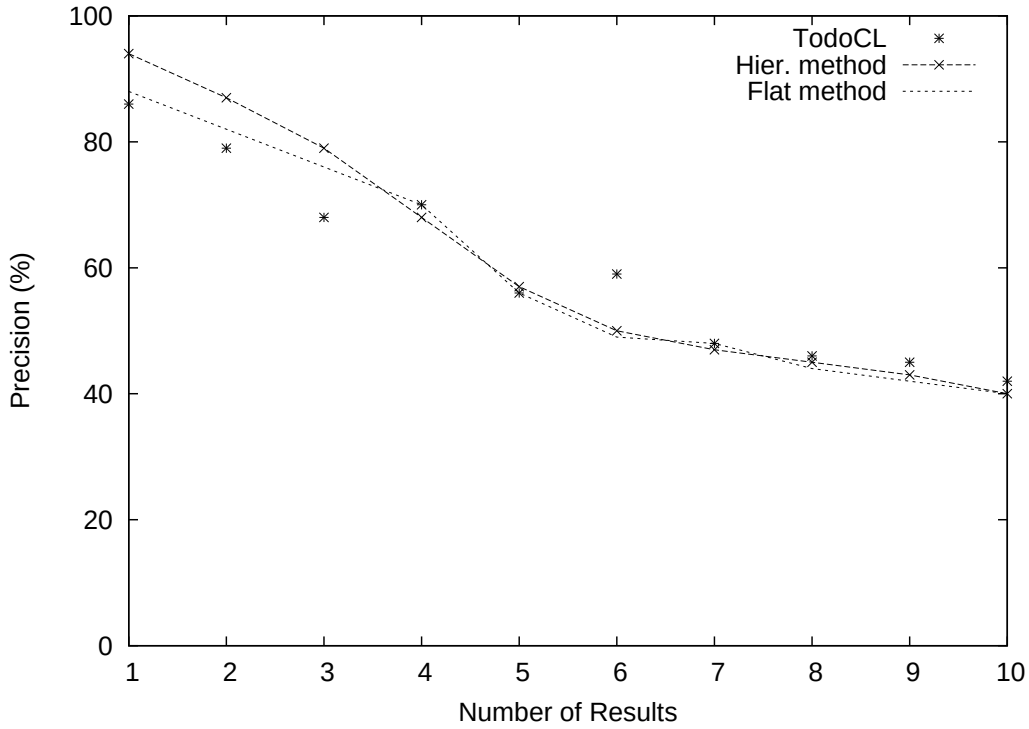


Figure 6.3: Average precision of the retrieved documents for the methods based on classifiers and for the search engine ranking.

into nodes enriching their descriptions and improving the coverage of the directory. Solving this problem we will state that the hierarchical method improves the precision of the answer lists compared with the flat classifier and with the original ranking.

6.1.4 Conclusion

We can conclude that our query classification method allows us to identify concepts associated to the queries. The taxonomy would permit us to specialize the query in order to provide final accurate recommendations. The proposed method also allows us to improve the precision of the retrieved documents over the first 5 positions of the answer lists. Compared with the search engine ranking method and compared with the method based on a flat classifier, the hierarchical classifier method provides better results regarding the precision of the answer lists.

One of the biggest constraints of the proposed method lies in the fact that it depends

strongly on the taxonomy quality. In the third experiment, we can notice that an important proportion of the nodes contain an insufficient quantity of recommended documents, which prevents a favorable comparison to the TodoCL ranking beyond the first 5 recommendations. Another constraint is that the taxonomy does not always allow us to identify the need behind the query. Both constraints are related to the fact that since the directory is manually maintained, it is limited in enlargement and freshness because of the frequency of the editor's evaluations and updates.

In the next section, we will present a method based on the proposed classification schema which permits us to maintain automatically the directories, by adding new queries and documents to each directory node.

6.2 The Web directory maintenance method

6.2.1 The method

Some Web directories are manually maintained by editorial staffs, while in other cases, they are updated by networks of volunteer editors. The manual maintenance of a Web directory is an extremely difficult and costly task due to the huge amount of documents and categories handled. As an example, the Open Directory project [DMO], the largest human-edited directory in the Web, comprises approximately 5 million Web pages, which have been classified into 590.000 categories by 70.000 editors. In addition, the dynamic nature of the Web makes it difficult to manually maintain a Web directory. As the Web changes and evolves, periodically several sites become obsolete, while many new relevant sites arise.

A Web directory should account for the quality and relevance of documents to the categories of the directory. Therefore, ideally, human editors should not only add and delete documents, but also change the ranks of documents that appear at each node. This task would require even more extra input and time for the editors. Furthermore, human editors may not necessarily represent the interest of common users to whose requirements the directory must be targeted at.

In this section, we explore the idea of processing the user's click data registered in the

logs of a search engine for the automatic maintenance of a Web directory. The method we present in this section uses the hierarchical classification method and is based on the vectorial representations of queries and categories given in the previous section.

The maintenance method operates in the following steps. First, we classify query sessions into categories using the method proposed in the previous section. Considering a category c, we compute queries that are related to c based on a measure we refer to as the utility of the query to the category. Intuitively, this measure estimates the ability of any arbitrary query to retrieve, in the first positions of its answer, documents that are relevant to the category. We expect that false positives achieve low values of utility. Using this measure and a threshold value, it would be possible to identify false positives eliminating them from the node.

After the classification process, each category can be viewed as a set of query sessions, which themselves have associated clicks made by users. Then we estimate the relevance of documents to the category based on these clicks. Documents are ranked in each category based on the estimated relevance.

We extract from the query sessions of a category c a sample to estimate the relevance of each document d to c. We assume that in a query session the user views d if she/he selects a document in a lower position than d's position. If this is the case, the query session can be accounted as an event for the sample. By standard statistic, we could estimate the true success rate from the sample. The error in the estimation depends on the number of events in the sample. We call it the *support* of d for c. This support would help us to discard documents that are considered by few users, even though these few users have clicked the document.

Consider a category c, which has associated a set N_c of query-sessions. Given a query session s, we denote by $\texttt{last}(s)$ the last position reached in s by the user, i.e., the position where the user skipped the search. We assume that this position corresponds to the last document clicked in the query session. We define the document support as follows: given a document u, we estimate the number of query-sessions in which the document was seen by users according to:

$$S(u, c) = \texttt{COUNT}(\{s \in N_c \,|R(u, s) < \texttt{last}(s)\}),$$

where $R(u, s)$ is the position of u in the answer list of s. $S(u, c)$ is called the *support* of the document in the category. The support $S(u, c)$ is the number of query sessions in the category c where u appeared before the last seen document. The relevance of a document d to a category c will be estimated as the success rate in the sample, which is the standard way of estimating the true success probability p of a Bernoulli process.

In order to define relevance, we will count the number of selections (successes) of the document u in the query sessions of the category c as follows:

$$C(u, c) = COUNT(\{s \in N_c \mid u \text{ was clicked in } s\}).$$

Then we define the estimated relevance of a document. The relevance of a document u to the category c is estimated as:

$$\texttt{relevance}(u, c) = \frac{C(u, c)}{S(u, c)}$$

Given a category c we rank each document u to c according to the estimated relevance $\texttt{relevance}(u, c)$. However, we only consider in the ranking documents u such that the support $S(u, c)$ is above a minimum threshold `MinSupp`.

Now we will explain how to obtain a ranking of recommended queries for each category of the directory. The idea is to order the queries according to their ability to retrieve documents which are relevant to the category. Moreover, we want to rank at first positions queries that return more relevant documents at first positions of their answers. In order to do this, we propose a measure called the *utility* of the query to the category.

Given a category c, we assume we have already computed the support and estimated relevance of the documents to c. The utility of a query is a measure of how useful the query is in returning relevant documents to a given category. Intuitively, a more useful query will return more relevant documents. However, we must carefully aggregate the relevances of the documents in order to consider the positions at which the documents are returned. A query will have more utility if the most relevant documents appear in higher positions of the ranking returned by the query.

We use a probabilistic approach to aggregate the relevances of the documents of a query.

For a fix query and position i of its result rank, let $V_i \in \{1, 0\}$ be a binary random variable which represents whether the user take into consideration position i for a selection. The probability distributions of the variables V_i's should reflect the bias users have for considering the different positions of the ranking for selections. We will use the same bias estimation technique proposed in the previous chapter.

The utility of a query is defined as the expected relevance of the documents considered by users in the query. For a URL u and a category c we define:

$$RS(u,c) = \begin{cases} \texttt{relevance}(u,c) & if S(u,c) \geq \texttt{MinSupp} \\ 0 & \text{Otherwise.} \end{cases}$$

Let us denote by $q(i)$ the document that the query q returns at position i. We next define the utility of a query to a category. Let c be a category c and q be a query. We define the utility of q to c, as follows:

$$\texttt{utility}(q,c) = \sum_i P(V_i = 1) \times RS(q(i), c).$$

6.2.2 Experimental Results

We present an experimental evaluation of the method, using the directory and the logs from TodoCL considering the L_6 log file. We consider ten categories of the directory of TodoCL and the first ten results for each recommendation method. The ten nodes were selected by doing a random sampling over the categories with more than 10 query sessions, which corresponds to 212 out of 468 nodes. The list of documents for each category includes document whose estimated relevance is greater or equal to 0.8. The original nodes along with the results shown in this section are translated from Spanish to English. Selected categories are shown in Table 6.2.

Here we show a graphic regarding the classification of query sessions to categories. In the experiments, we used query-sessions with at least three documents selected, which yield 20,536 query sessions. Figure 6.4 shows the distribution of the distances of the query sessions to the categories to which they were classified. Frequency values represent the number of query-sessions with the distance value to their closest category indicated in

	Category
(1)	*business:finances:banks*
(2)	*society:law:norm:codes*
(3)	*business:building-industry:builders*
(4)	*business:environment:engineering*
(5)	*business:sales:gifts:flowers*
(6)	*society:history*
(7)	*leisure:sports:motorcycling*
(8)	*business:informatics:support*
(9)	*leisure:gastronomy:drinks:wine*
(10)	*business:foreign trade:customs duty*

Table 6.2: Categories of the TodoCl directory selected for the experiments.

the abscissa. From this histogram, we can see that an important proportion of the query-sessions are not appropriated to use for category classification. Based on this fact, we use a distance threshold between categories and query sessions of 0.7. In this setting, only 3,656 query sessions where classified into a category.

The precision of the listed documents at the categories is a central quality of a Web directory. Users of Web directories do not tolerate errors, and their primary motivation to browse the directory is to find documents of high relevance to the category. We carried out a user evaluation to compare the retrieval precision of our method with the precision of the TodoCL directory. Documents in both the TodoCL directory and the directory automatically computed by the method were mixed and their relevance to the categories they belong to were judged by members of our laboratory. Our evaluation method is similar to the comparative precision technique of Chakrabarti et al. [Cha03]. We presented to a group of 19 persons of different backgrounds the selected categories and the document list. We asked one question to the participants: Is the document relevant to the category? The answer could be expressed using relevance degrees varying from 0 - 4, from lower to higher relevance. Subsequently, we calculated the average relevance of every category and document on every opinion expressed by the users. The average precision for a given position is calculated over the total number of categories considered in the experiment.

The average precisions obtained for the two directories are shown in figure 6.5. As the graphic shows, both methods behave well regarding precision. For example, for the first

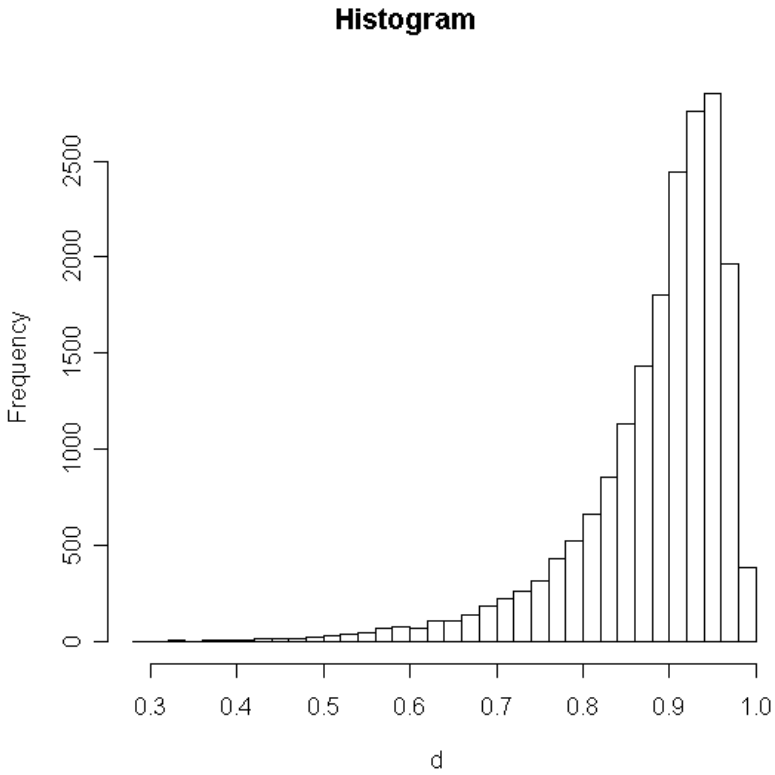


Figure 6.4: Histogram of distances of query sessions to their assigned categories.

seven positions, both methods have over than 80% precision. Intuitively, we expect that human recommendations (in the TodoCL directory) are more precise than the automatic recommendations but the graphic shows that our method is even slightly better for the data set studied. Our experiment shows that the proposed method avoids manual maintenance without paying a cost in precision.

Table 6.3 shows the distribution of the set of documents over the two taxonomies. Here A (automatic) and H (human) represent the directory built by our method and the TodoCL directory, respectively. As an example, among the 100 documents in the TodoCL directory 48 appear in A directory, and only 5 of them where identified as non-relevant document in the experiments. The table shows that editors help to improve the precision of the directory built from the logs, and common users also help to improve the taxonomy built by the editors. The table suggests that the two independent sources of evidence could be mixed to

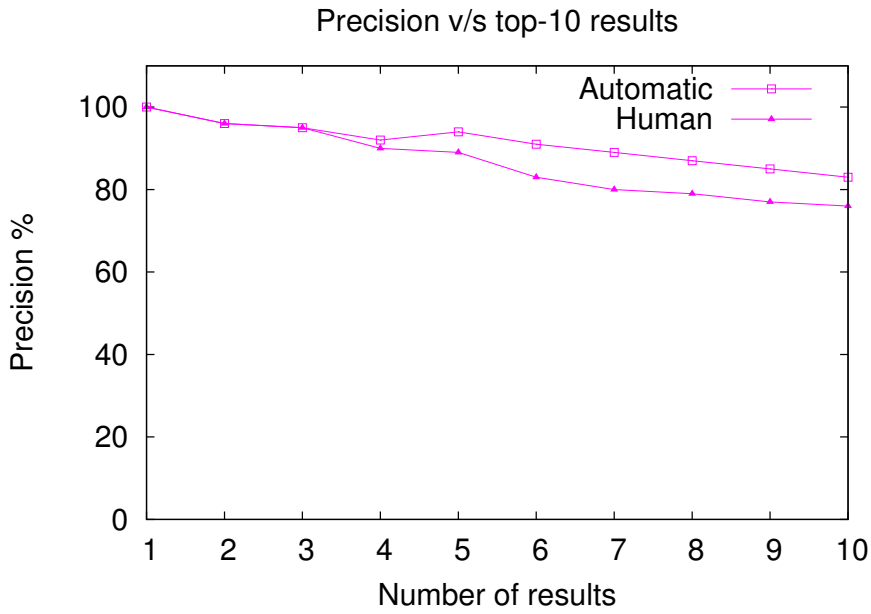


Figure 6.5: Average precision for the two methods over the top ten results.

Set	Docs.	Relevant	Precision	Recall
A	100	83	83%	71%
H	100	76	76%	65%
$H \cap A$	48	43	93%	37%
$H - A$	52	33	63%	28%
$A - H$	52	40	77 %	34%

Table 6.3: Precision and Recall.

obtain a better directory.

Table 6.3 also shows the recall of the two directories over the collection of documents in the selected nodes. Among the 200 documents selected for the experiments, 116 were found to be relevant. Thus, the recall percentage of the last column is over the 116 relevant documents. Again, the new directory slightly improves the manual. Notice that our method allows us to identify relevant documents which have not been included in the original directory. For the 10 selected categories, we found 40 new relevant documents that do not appear in the TodoCL directory.

We also evaluated our query recommendation method. For each category selected for

the experiment, we evaluated the precision of the top-5 answers of each of the first 10 queries recommended to the category. The precision was evaluated by members of our laboratory.

Figure 6.6 shows the average precision obtained compared with the average precision of the search engine, according to the evaluation provided in [BYSJC02]. For example, for the first five position, the average precision is over 60%. As the graphic shows, both precision curves are similar. Therefore, for a given category c, our method recommends queries q so that the precision of the answers of q to c reaches the average precision of the search engine.

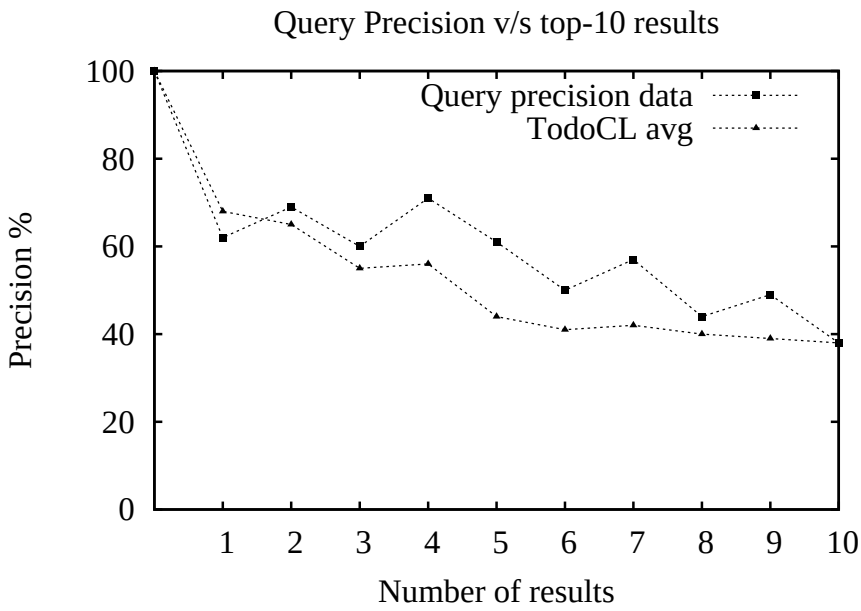


Figure 6.6: Average precision for the query recommendation method over the top ten results.

In table 6.4, we depict the first five queries recommended by our method to each of the categories selected in the experiment. The first column shows the *id* of the category according to the enumeration of table 6.2. The second column shows the topic related to the category. The third column shows the queries recommended. Queries are ordered according to their utility to the category. The utility of the query is shown in the last column. Wrong recommendations are shown in cursive (six queries over the 50 recommended

queries).

As table 6.4 shows, some recommendations are very interesting. Most queries recommended do not share common terms with the category descriptive text and each one represents clearly defined information needs.

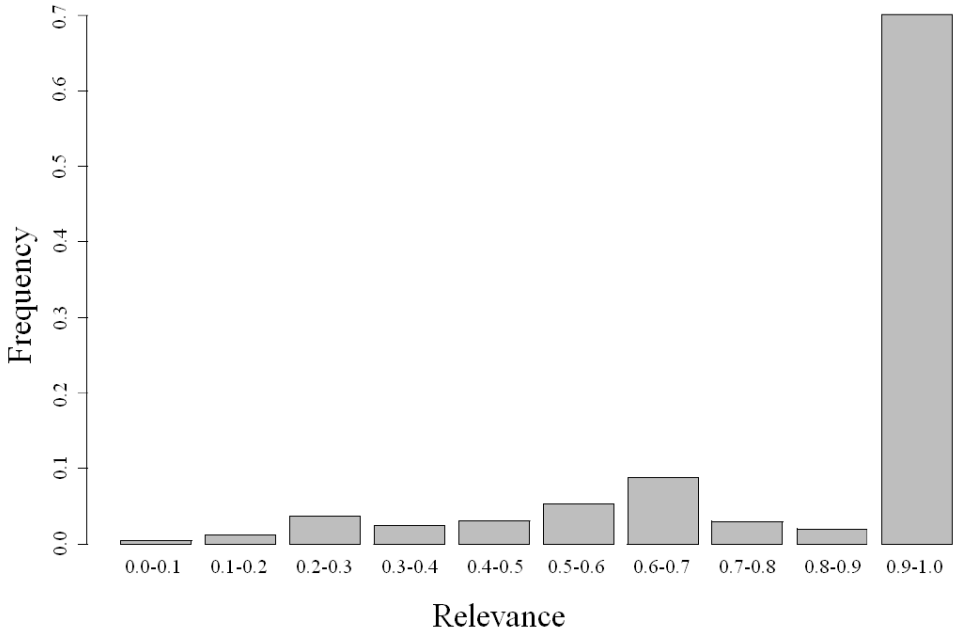


Figure 6.7: Histogram of estimated precision of documents in the manual directory.

As we have shown, our method allows us to find new relevant documents, which are not originally in the directory. In this section, we evaluate the ability of the method to predict the preferences of the human editors that have built the directory. Figure 6.7 shows the estimated relevance of the documents that appear in the TodoCL directory for the ten categories selected.

The histogram shows that the vast majority of the documents recommended by editors have a high estimated relevance. For 70% of the documents, their estimated relevance is greater than 0.9.

ID	Topic	Queries Recommended	Utility
(1)	banks	banks	1.26
		banking	1.16
		image banks	1.13
		photo banks	0.33
		financier	0.29
(2)	law codes	labor code	2.83
		bar code	2.33
		code of civil procedure	2.09
		code of judicial conduct	2.07
		code of penal procedure	2.05
(3)	builders	houses	10.97
		prefabricated houses	10.09
		wooden houses	6.99
		home building	4
		builders	3.99
(4)	environmental	environmental engineering	1.75
		environmental impact	1.08
		resolution	0.66)
		environmental defense	0.59
		environmental campaigns	0.55
(5)	gifts flowers	*teddy bears*	4.38
		flower stores	4.12
		cakes	2.96
		roses	2.73
		flowers	2.21
(6)	history	local history	2.66
		history of Chile	2.24
		naval battle of Iquique	1.86
		the independence of Chile	1.68
		Chilean folklore	0.93
(7)	motorcycling	spare parts	2.33
		motorcycling	1.93
		motorcycle racers	1.13
		professional riders	1.21
		motorcycle racing	1.04
(8)	informatics support	used notebooks	5.59
		housing	3.98
		monitors	3.09
		hardware	0.75
		hosting	0.66
(9)	wine	wine	1.84
		wine train	1.25
		wine grapes	1.23
		wine tasting	1.15
		wine glasses	0.87
(10)	customs duty	customs duty	3.56
		Iquique	1.4
		Zofri	1.33
		duty free areas	1.12
		Iquique's duty free zone	1.04

Table 6.4: Top-5 queries recommended for each category.

6.3 Conclusion

The classification method proposed in this chapter allows the identification of well defined concepts for each query classified. The maintenance method proposed allows the Web directory to be automatically maintained without losing precision and recall. By considering the logs in the maintenance process we take into account information based on user's click data. The preferences are bound to query sessions, and thus we may obtain preferences sensible to topics users are looking for. The importance and quality of a Web page is reflected by the preferences registered in the Web log. This allows us to find many new relevant documents not considered in the directory built by human editors.

Chapter 7

Conclusions

In this chapter we present concluding remarks about the methods and the experimental results introduced in this thesis. We will also discuss relevant properties of the user preference data. Finally we will enumerate open problems and future work.

7.1 Concluding Remarks

A first question of interest after exploring different recommendation methods based on user's click data along this thesis, is the following: what have we learned? The objective of this question lies in the identification of the new knowledge acquired, derived from this thesis. The answer to this question will allow us to identify the contributions of this thesis. This answer will basically be the final objective of this chapter.

First, we will summarize what we have learned in the Chapter about data analysis. As we can observe in the thesis objectives, its goal is to identify the interesting data properties that would later allow us to design recommendation techniques of queries and documents based on the user's click data, which are favorably compared to the traditional recommendation methods. These properties, of course, would also make it possible to improve them. The experiment shows mainly the following:

1. The frequency of the user queries follows a Zipf distribution. This means that few queries are very popular, that is to say, they are submitted frequently. On the other

hand, most of the queries are submitted very few times. As a consequence of the above mentioned fact, most of the queries have few associated query sessions. This has a direct consequence on the methods we propose which require multiple sessions.

2. User's clicks are biased to the ranking. The distribution of the user's clicks depending on the location of the documents in the TodoCL ranking, shows a Zipf distribution. This means that most of the user's clicks are preformed on the documents shown in the first positions of the ranking. On the other hand, very few documents which appear in non-favorable positions in the ranking register preferences. This is, the users check few documents in their sessions, and those that they check are among the first recommended ones. Then, the recommendation methods of queries and documents based on user's click data must approach the problem of reduction of bias inherent to the user's click data generation.

3. A significant proportion of the users reformulate their initial query several times before selecting a document. This means that the queries do not always represent the user's needs. That is to say, few words are frequently used and they appear in most of the documents. Due to this fact, its descriptive capacity is lower. Consequently, the term-based methods elaborate deficient and inaccurate recommendations. Every search engine must allow the user to reformulate his/her query suggesting more accurate terms which finally permit us to represent exactly the meaning of the query. This suggests modeling query refomulation and last clicked document.

4. A large part of the users submit queries and then do not select any of the recommended documents. This means that the satisfaction degree of the user concerning the answer lists provided by the traditional methods is low. Every search engine must then improve the precision of the provided answer lists.

Regarding what we have learned in the chapter of user's click data analysis, each of the techniques was focused on improving some of the aspects presented. Result 1 (the frequency of the queries follows a Zipf distribution) was adressed in the chapter about query clustering and in the one about query classification, by using recommendation techniques based on identification of similar query clustering. Result 2 (the preferences are biased to

the ranking) was explained in the chapter about query clustering, introducing a method of bias reduction. Result 3, (a significant proportion of the users reformulate his/her initial query several times before selecting a document) was approached in the chapter about query association and also, to some extent, in the other chapters whose final objective was to improve the precision of the answer lists. Result 4 (an important proportion of the users submit queries and then do not select any of the recommended documents) was explained on the whole in the 3 approaches by proposing methods whose objective is to identify the need behind the query and to improve the quality of the provided answer lists.

As a conclusion, and with the purpose of synthesizing the contributions of this thesis, we can list the following:

1. The query recommendation techniques presented in this thesis allow us to identify the user needs in most of the cases studied in the experiment. Among them, the term-based method shows very good performance when the initial query must be specialized and also when the terms of the original queries denote their meaning. These terms are also expected not to be polysemic, in order to avoid the interruption of noise in the recommendations. The co-citation based technique is very accurate for queries with several associated sessions, that is, it permits us to disambiguate frequent queries, since these are the ones that have a larger number of associated preferences. Also by using this data the method performs in a better way. The recommendation techniques are complementary, since they work on different types of queries (they are frequent in the case of co-citation and accurate in the case of the method based on the terms).

2. The query clustering techniques based on the terms of documents or snippets allow us to make relevant recommendation of queries and/or documents, even for those with few sessions. Therefore, most of the recommended queries are relevant to the users and allow them to satisfy the need behind them. Also, the ranking technique based on query clusters is favorably compared with the original ranking of the search engine. Finally, the bias reduction method allows the proposed recommendation techniques to identify documents in the last positions of the original ranking, reducing effectively, in most cases, the bias inherent to the user's click data. In conclusion,

the experiments show that the clusters successfully represent the well-defined needs of the users, and therefore, their application in the elaboration of documents and/or query recommendations make it possible to improve the precision of the answer lists, compared to the techniques based only on terms or hyperlinks.

3. The query classification techniques allow us to identify general concepts behind the queries. The proposed classification technique is appropriate and in the presence of a quality taxonomy, it also allows the user to refine his/her query. The directory maintenance technique allows us to associate relevant documents to each taxonomy node and they also permit us to identify related queries. They are suitable for users who need to better define their queries more specifically and to check short but highly accurate lists of documents.

4. The recommendation techniques of queries and documents based on user's click data are favorably compared to methods based on other data sources, allowing us to elaborate accurate recommendations. The descriptive experiments, as well as the ones based on the experts' opinions, make it possible to establish that the data source is useful and that the recommendation methods based on user's click data are able to identify documents and queries semantically related to the initial query, even when they do not share terms or they are non-frequent queries.

Now we can focus on the nature of the data. Regarding the usefulness of the data source we can state the following:

1. The user's click data allows us to elaborate recommendations of queries and/or documents, which have been favorably evaluated by the users.

2. The methods of query reformulation, as well as the query clustering and the directory-based one have been able to show that the appropriate application of the user's click data, allows us to elaborate quality recommendations.

Therefore, we can state that regardless of the used recommendation method; the user's click data is useful in the elaboration of recommendations. Now, I will consider the document recommendation technique based on query clustering to formulate some conclusions.

If we consider that the proposed ranking is proportional to the popularity, then it is possible to conclude that the relevance of the recommended documents is proportional to the popularity variant. It is worth observing that we are considering the popularity adjusted according to the bias reduction method. The previous conclusion is not valid if we consider the popularity biased to the ranking. We can state then, that the user's clicks data is useful for the elaboration of rankings of documents relevant for the users.

At this point, it is relevant to state that we used selections and reading time as measures of user interests because these behaviors are easily retrieved from the query log data, without the cooperation of users. We used these behaviors also based on the literature. For example, Kelly and Teevan [KT03] show that these behaviors show interesting properties as measures of implicit feedback and are highly correlated with user interests. Thus, they can be used as implicit feedback measures. For example, we assume that the number of selections is useful to measure the relevance of a document. We also used selections and reading times in distance functions introduced in chapters five and six, in order to represent queries and query sessions. Our experiments show enough evidence of the success of these decisions, even though as a future work we could explore other user behaviors in order to measure feedback, such as scrolling.

Regarding another relevant property called heterogeneity –that is to say the property of the user's click data related to the capacity for distinguishing different meanings associated to the queries submitted by the users– we can state the following: we have said that the data is heterogeneous if it has enough information which allows us to disambiguate the polysemic queries. Although, the experiments show that the space of query terms is sparse, it has been possible to generate query reformulation structures which have been well-evaluated by the users. By applying the proposed query reformulation techniques. We have been able to point out that the use of document terms selected in the sessions of these queries has been useful to map them out in a less sparse space, where it is effectively possible to disambiguate them with few associated sessions. We have also pointed out that this has been possible due to the fact that the proposed methods identify semantic relations among the queries, by using co-citation or structures as well as directories. Therefore, we can state then, that the user's preferences data is heterogeneous enough to be able to elaborate query recommendations or to provide query reformulation structures even for queries

with little history, allowing in most cases to identify better queries.

7.2 Future work

As a follow up, we will include in this section the problems associated with the efficient implementation of the recommendation methods presented in this thesis. There are many interesting problems in connection with the efficient implementation of the proposed recommendation methods. All these problems have to do with the algorithmic aspect inherent to the implementation of the methods proposed in this thesis, which opens a new field of research regarding the development of efficient technology which allows us to support the recommendation methods presented here.

It is also possible to improve the proposed recommendation methods or to explore some of their characteristics to reinforce the search process. For example, the descriptive terms obtained from the process of query clustering can be used to make an expansion. It is also necessary to explore the design of new recommendation methods which are able to provide quality recommendations to queries which are not registered in the logs. Finally, a problem associated with the searches supported by directories lies in the design of taxonomy construction methods. By applying the user's click data, it should be possible to identify specialization/generalization concepts and relations among them, which should provide us with the definition of a concept taxonomy. This is one of the topics on which I am diligently working on at the moment.

As an open problem, we pose the necessity of approaching the design of a new generation of high-precision search engines. A new generation of Web search engines would be able to identify the need behind the query answering appropriately according to the taxonomy of possible searches defined by Broder [Bro02]. In order to achieve this goal, it is necessary not only to develop new recommendation methods, as the ones proposed in this thesis, but also to develop a new search paradigm. In this new paradigm, the search process must be considered as a dynamic process, where the user needs can be modified in the process itself, based on its feedback by the user. One way to address this challenge is to add semantic elements to search systems. There exists two strategies to do this. First, it is possible to enrich the document description adding annotations that can be written in

RDF in order to provide semantics interpretable for a computer. Unfortunately, common users are reluctant to add annotations to their pages because it requires additional efforts and the benefits are not evident to them. The second strategy is to extract the content from web pages using automatic methods. Using information about specific topics organized into ontologies, it is possible to identify concepts behind queries. Some works show that by using this approach, it is possible to identify the context of keywords during typical searches [WBNBY04]. However, due to the lack of a big ontology, the method is limited in its results and conclusions. Of course, with the existence of a big and dynamic ontology these kinds of *semantic search engines* could be successful in the future.

As we can see, there are more open problems than answers regarding the future of web search engines. We know that the technology of the new search engines will be able to enlighten the search process, making the search flexible instead of rigid. This thesis has been intended to be a contribution in this sense.

Bibliography

[BB98] K. Bharat and A. Broder. A technique for measuring the relative size and overlap of public web search engines. *Computer Networks*, 30(1-7):379–388, 1998.

[BB00] D. Beeferman and A. Berger. Agglomerative clustering of a search engine query log. In *Proceedings of the sixth ACM SIGKDD International Conference on Knowledege Discovery and Data Mining, Boston, MA, USA*, pages 407 – 416. ACM Press, August 2000.

[BCB94] B. Bartell, G. Cottrell, and R. Belew. Optimizing parameters in a ranked retrieval system using multi-query relevance feedback. In *Third Annual Symposium on Document Analysis and Information Retrieval*. University of Nevada, 1994.

[BJC+04] S. Beitzel, E. Jensen, A. Chowdhury, D. Grossman, and O. Frieder. Hourly analysis of a very large topically categorized web query log. In *SIGIR 04*, pages 321–328, New York, NY, USA, 2004. ACM Press.

[BKM+00] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener. Graph structure in the web. *Computer Networks*, 33(1-6):309–320, 2000.

[BP98] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.

[Bro02] A. Broder. A taxonomy of web search. *SIGIR Forum 36*, 2002.

[BSWZ03] B. Billerbeck, F. Scholer, H. Williams, and J. Zobel. Query expansion using associated queries. In *CIKM 03: Proceedings of the twelfth international conference on Information and knowledge management*, pages 2–9, New York, NY, USA, 2003. ACM Press.

[BY04] R. Baeza-Yates. Query usage mining in search engines. *Web Mining: Applications and Techniques, Anthony Scime, editor. Idea Group*, 2004.

[BYCG06] R. Baeza-Yates, L. Calderón, and C. González. The intention behind web queries. 4209:98–109, October 2006. Glasgow, UK.

[BYHM04a] R. Baeza-Yates, C. Hurtado, and M. Mendoza. Query clustering for boosting web page ranking. In *AWIC 2004, Cancun, Mexico, May 16-19*, volume 3034 of *Lecture Notes in Artificial Intelligence*, pages 164–175. Springer, 2004.

[BYHM04b] R. Baeza-Yates, C. Hurtado, and M. Mendoza. Query recommendation using query logs in search engines. In *EDBT 2004 Workshops, Heraklion, Crete, Greece, March 14-18*, volume 3268 of *Lecture Notes in Computer Science*, pages 588–596. Springer, 2004.

[BYHM07] R. Baeza-Yates, C. Hurtado, and M. Mendoza. Improving search engines by query clustering. *To appear In Journal of the American Society for Information Systems and Technology (JASIST)*, 56, 2007.

[BYHMD05] R. Baeza-Yates, C. Hurtado, M. Mendoza, and G. Dupret. Modeling user search behavior. In *Proceedings of the 3rd Latin American Web Conference (LA-WEB)*, pages 242–251, Buenos Aires, Argentina, October 2005. IEEE Computer Society Press.

[BYRN99] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley, 1999.

[BYSJC02] R. Baeza-Yates, C. Saint-Jean, and C. Castillo. Web dynamics, age and page quality. In Springer, editor, *LNCS Proceedings of SPIRE*, September 2002. Lisbon, Portugal.

[CC02] S. Chuang and L. Chien. Towards automatic generation of query taxonomy: A hierarchical query clustering approach. *In Proc. of the Second IEEE International Conference on Data Mining (ICDM'02)*, 3268:588–596, 2002.

[Cha03] S. Chakrabarti. *Mining the Web*. Morgan Kaufmann Publishers, San Francico, 2003.

[CHM06] A. Cid, C. Hurtado, and M. Mendoza. Automatic maintenance of web directories using click-through data. In *Proceedings of the 2nd International Workshop on Challenges in Web Information Retrieval and Integration*, Atlanta, Georgia, USA, April 2006. IEEE Computer Society Press.

[CIW] CIW/CWR. *Center of Web Research*. `http://www.cwr.cl/`.

[CLL+02] Z. Chen, F. Lin, H. Liu, W. Ma, and L. Wenyin. User intention modelling in web applications using data mining. *World Wide Web*, 5(2):181–192, 2002.

[CLWB01] M. Claypool, P. Le, M. Waseda, and D. Brown. Implicit interest indicators. In *Proceedings of the 2001 International Conference on Intelligent User Interfaces, January 14-17, 2001, Santa Fe, NM, USA. ACM*, pages 33–40, 2001.

[CMS99] R. Cooley, B. Mobasher, and J. Srivastava. Data preparation for mining world wide web browsing patterns. *Knowledge and Information Systems*, 1(1):5–32, 1999.

[CP03] F. Crestani and G. Pasi. Handling vagueness, subjectivity, and imprecision in information access: an introduction to the special issue. *Information Processing and Management*, 39(2):161–165, 2003.

[DH99] J. Dean and M. Henzinger. Finding related pages in the world wide web. *Computer Networks Amsterdam, Netherlands: 1999*, 31(11-16):1467–1479, 1999.

[DK01] M. Deshpande and G. Karypis. Selective markov models for predicting web-page accesses. In *1st SIAM Data Mining Conference, April 5-7, Chicago, USA*, 2001.

[DM05] G. Dupret and M. Mendoza. Recommending better queries from click-through data. In *SPIRE 2005, Buenos Aires, Argentina, Nov 2-4*, volume 3772 of *Lecture Notes in Computer Science*, pages 41–44. Springer, 2005.

[DM06] G. Dupret and M. Mendoza. Automatic query recommendation using click-through data. In *Symposium on Professional Practice in Artificial Intelligence, 19th IFIP World Computer Congress*, pages 303–312, Santiago, Chile, August 2006. Springer Science and Business Media Press.

[DMO] DMOZ. *Open Directory Project*. `http://dmoz.org/`.

[ENC] ENCARTA. *Encarta Encyclopedia*. `http://encarta.msn.com/`.

[FGDMZ03] B. M. Fonseca, P. B Golgher, E. S. De Moura, and N. Ziviani. Using association rules to discovery search engines related queries. In *First Latin American Web Congress (LA-WEB 03), November 10-12, Santiago, Chile*, 2003.

[FS91] H. Frei and P. Schäuble. Determining the effectiveness of retrieval algorithms. *Information Processing and Management*, 27(2):153–164, 1991.

[GGM05] Z. Gyongyi and H. Garcia-Molina. Link spam alliances. In *Proceedings of the 31st VLDB Conference*, 2005.

[Goo] Google. *Google search engine*. `http://www.google.com/`.

[GS05] A. Gulli and A. Signorini. The indexable web is more than 11.5 billion pages. In *WWW '05: Special interest tracks and posters of the 14th international conference on World Wide Web*, pages 902–903, New York, NY, USA, 2005. ACM Press.

[Jee] Ask Jeeves. *Ask Jeeves search engine*. `http://www.ask.com/`.

[JS05] B. Jansen and A. Spink. An analysis of web searching by european alltheweb.com users. *Information Processing and Management*, 41(2):361–381, 2005.

[Kle99] J. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.

[KT03] D. Kelly and J. Teevan. Implicit feedback for inferring user preference: a bibliography. *SIGIR Forum*, 37(2):18–28, 2003.

[Lal98] M. Lalmas. *Information Retrieval: Uncertainty and Logics*. Kluwer Academic Publishers, Norwell, MA, USA, 1998.

[LLC05] U. Lee, Z. Liu, and J. Cho. Automatic identification of user goals in web search. In *Fourteenth International World Wide Web Conference, May 10-14, Chiba, Japan*, pages 391–400. ACM Press, 2005.

[LS68] M. Lesk and G. Salton. Relevance assessments and retrieval system evaluation. *Information Storage and Retrieval*, 4(3):343–359, 1968.

[LYC] LYCOS. *Lycos Search Engine*. `http://www.lycos.com/`.

[Mas51] F. Massey. The kolmogorov-smirnov test for goodness of fit. *Journal of the American Statistical Association*, (46):68–78, 1951.

[MS94] M. Morita and Y. Shinoda. Information filtering based on user behavior analysis and best match text retrieval. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Ireland*, pages 272–281. ACM, 1994.

[Nat98] I. Nathenson. Internet infoglut and invisible ink: Spamdexing search engines with meta tags. *Harvard Journal of Law and Technology*, 12(1):44–93, 1998.

[OTHK04] S. Otsuka, M. Toyoda, J. Hirai, and M. Kitsuregawa. Extracting user behavior by web communities technology on global web logs. In *DEXA*, pages 957–988, 2004.

[Piw00] B. Piwowarski. Learning in information retrieval: a probabilistic differential approach. In *Proceedings of the BCS-IRSG, 22nd Annual Colloquium on Information Retrieval Research*, Cambridge, England, 2000. Sidney Sussex College.

[PP02] N. Papadopoulos and D. Plexousakis. The role of semantic relevance in dynamic user community management and the formulation of recommendations. Technical report, Institute of Computer Science - Hellas, Department of Computer Science, University of Grecce, 2002.

[PSC+02] J. Pitkow, H. Schatze, T. Cass, R. Cooley, R. Turnbull, A. Edmonds, E. Adar, and T. Breuel. Personalized search. *Communications of the ACM*, 45(9):50–55, 2002.

[RL04] D. Rose and D. Levinson. Understanding user goals in web search. In *Thirteenth International World Wide Web Conference, May 17-22, 2004, New York, USA*, pages 13–19. ACM Press, 2004.

[RS67] A. Rees and D. Schultz. A field experimental approach to the study of relevance assessments in relation to document searching. Technical report, NFS, 1967.

[Sea] MSN Search. *MSN search engine*. `http://www.msn.com/`.

[SG01] A. Spink and O. Gunar. E-commerce web queries: Excite and ask jeeves study. *First Monday*, 6(7), 2001.

[SH06] M. Sahami and T. Heilman. A web-based kernel function for measuring the similarity of short text snippets. In *Proceedings of the 15th international conference on World Wide Web, WWW 2006, Edinburgh, Scotland, UK, May 23-26*, pages 377–386. ACM, 2006.

[SMHM99] C. Silverstein, H. Marais, M. Henzinger, and M. Moricz. Analysis of a very large web search engine query log. *SIGIR Forum*, 33(1):6–12, 1999.

[SPS+05] D. Shen, R. Pan, J. Sun, J. Junfeng, K. Wu, J. Yin, and Q. Yang. Q2caust: Our winning solution to query classification in kddcup 2005. *SIGKDD Explorations*, 7(2):100–110, 2005.

[SW02] F. Scholer and H. Williams. Query association for effective retrieval. In *CIKM 02: Proceedings of the eleventh international conference on Information and knowledge management*, pages 324–331, New York, NY, USA, 2002. ACM Press.

[Tod] TodoCL. *TodoCL search engine*. `http://www.todocl.cl/`.

[VBH+05] D. Vogel, S. Bickel, P. Haider, R. Schimpfky, P. Siemen, S. Bridges, and T. Scheffer. Classifying search engine queries using the web as background knowledge. *SIGKDD Explorations*, 7(2):117–122, 2005.

[WBNBY04] K. Wechsler, J. Baier, M. Nussbaum, and R. Baeza-Yates. Semantic search in the www supported by a cognitive model. In *Advances in Web-Age Information Management: 5th International Conference, WAIM 2004, Dalian, China, July 15-17*, volume 3129 of *Lecture Notes in Computer Science*, pages 315–324. Springer, 2004.

[WNZ01] J. Wen, J. Nie, and H. Zhang. Clustering user queries of a search engine. In *Proceedings of the Tenth International World Wide Web Conference, Hong-Kong, China, May 1-5*, pages 162–168, 2001.

[WNZ02] J. Wen, J. Nie, and H. Zhang. Query clustering using user logs. *ACM Transactions on Information Systems*, (1):59–81, 2002.

[XS00] J. Xu and A. Spink. Web research: The excite study. In *Proceedings of WebNet 2000, San Antonio, Texas, USA, October 30 - November 4*, pages 581–585. AACE, 2000.

[XZC+02] G. Xue, H. Zeng, Z. Chen, W. Ma, and C Lu. Log mining to improve the performance of site search. In *WISE Workshops*, volume 33, pages 238–245, 2002.

[XZC+04] G. Xue, H. Zeng, Z. Chen, Y. Yu, W. Ma, S. Wen, and W. Fan. Optimizing web search using web click-through data. In *CIKM 04*, pages 118–126, New York, NY, USA, 2004. ACM Press.

[Yah] Yahoo! *Yahoo search engine*. `http://www.yahoo.com/`.

[YH03] A. Ypma and T. Heskes. Automatic categorization of web pages and user clustering with mixtures of hidden markov models. In *WEBKDD 2002, Edmonton, Canada,July 23*, volume 2703 of *Lecture Notes in Computer Science*, pages 35–49. Springer, 2003.

[ZD02] D. Zhang and Y. Dong. A novel web usage mining approach for search engines. *Computer Networks*, 39(3):303–310, 2002.

[ZHC+04] H. Zeng, Q. He, Z. Chen, W. Ma, and J. Ma. Learning to cluster web search results. In *SIGIR 04*, pages 210–217, New York, NY, USA, 2004. ACM Press.

[ZK02] Y. Zhao and G. Karypis. Criterion functions for document clustering. Technical report, University of Minnesota, Minneapolis, USA, 2002.

[ZK04] Y. Zhao and G. Karypis. Empirical and theoretical comparisons of selected criterion functions for document clustering. *Machine Learning*, 55(3):311–331, 2004.

[ZS02] O. Zaiane and A. Strilets. Finding similar queries to satisfy searches based on query traces. In *Advances in Object-Oriented Information Systems, OOIS 2002 Workshops, Montpellier, France, September 2*, volume 2426 of *Lecture Notes in Computer Science*, pages 207–216. Springer, 2002.