PERSONALIZED WEB SEARCH USING CLICKTHROUGH HISTORY

A THESIS

submitted by

U. ROHINI

(rohini@research.iiit.ac.in)

(200407019)

in partial fulfillment of the requirements for the degree

of

MASTER OF SCIENCE by Research

in Computer Science and Engineering

LANGUAGE TECHNOLOGY RESEARCH CENTER

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY.

July 2007

THESIS CERTIFICATE

I certify that I have read this thesis and that, in my opinion, it is fully adequate in scope and qualify as a thesis for the degree of Master of Science by Research.

Vasudeva Varma (Principal Advisor)

Date:

ACKNOWLEDGMENTS

The research presented in this thesis was supported by Language Technologies Research Centre (LTRC).

I would like to thank Prof. Sangal for his guidance and providing me this opportunity for pursuing Masters through Research.

I would like to thank my guide Dr. Vasudeva Varma for his guidance, support and encouragement. I am thankful to him for providing the open and friendly research environment and giving me freedom for doing my research work. I should particularly thank him for giving me the opportunity to present my work at major research organizations which helped for my future opportunities.

I would not have entered into the area of research with out the enlightening and motivational discussions by my good friend and colleague Mr. Vamshi Ambati which helped me choose research as my career. I would like to thank him for involving in the discussions about my research and providing his valuable feedback and providing some resources for conducting my experiments. I would like to thank him for providing me opportunity to present my work and get feedback from researchers at top universities. I would like to thank him for the motivation, ideas, suggestions and constant moral support.

I would like to thank my parents for their support, encouragement and patience through out my thesis. I would like to thank my sister and brother for their encouragement.

I would like to thank the major search engine organization which made available its search engine query logs for research purposes. The release of this data has greatly helped my research. Last but not least, I would like to thank my friends, and colleagues for the support and interest.

U. Rohini

ABSTRACT

Keywords: *Personalized Search; Personalized Web Search; Web Search; Re ranking; User Modeling; Improving Web Search; User Profiling; Improving Ranking; Query Log Analysis; Simulated Feedback*

The main problem with web search is as follows : There is too much information available on the web; query words used by users often are confusing, ambiguous (a query "Java"can mean the Java island in Indonesia or the Java programming language), and some times are poor descriptors of information need ('SBH" can mean "State Bank of Hyderabad" or "Syracuse Behavioral Healthcare" among others) ; users are often not patient enough to see long list of results given by search engines to find relevant information. It has been observed that users typically view top few results, usually top 5 or 10, some times 20 and much fewer times 30 and so on. In this scenario, web search can be made more useful, effective and less burdensome to users by - trying to infer what would be relevant for the current user for a given query considering individual users' interests and provide those results on the top so that the user does not have to scroll down a long list of results.

The problem of Personalized Search aims to customize search results according to each individual user for him to find the most relevant documents to him on the top by considering his idiosyncrasies. This would possibly satisfy them and help in finding relevant information easily and quickly.

The major challenges for personalized search are two fold. The first is, modeling appropriate user context and learn a user model. The second is, how to utilize the user model to improve search accuracy. Another important challenge is evaluation of experiments. There are no standard and bench mark datasets available on which experiments can be performed. This makes comparison with earlier work in the litera-

ture and replicating their results difficult. There are also no standard metrics available to effectively evaluate personalized search algorithms. Some commonly used metrics used to evaluate Information Retrieval systems are usually used.

We propose three approaches for personalized web search. Our first approach is based on Statistical Language Modeling techniques. In spite of the progress made in language modeling and IR recently, there has not been much work applying language modeling techniques to personalized web search. In this approach, we learn a user model by capturing statistical properties of text from his past searches. We explored different contexts. The different contexts include using single word and two adjacent words (Simple N-Gram based method) and capturing relationship between query and document words (Noisy Channel model based method). Our second approach is based on Machine Learning algorithms which show an interesting and promising framework for learning user profiles. We make use of ranking SVM, a variation of the classification support vector machines for learning the user model. The above two approaches are a class of approaches that have exploited user feedback data either explicit or implicit. The third approach is another interesting approach, where, we attempt to model a particular user based on only past queries posed by corresponding user. The basic idea is to see if there is enough information available in just the previous queries without having to use teh clickthrough data made by the user. We have employed simple approaches based on language modeling to learn the user model from the previous queries of the user.

We performed experiments on data extracted from query log data from a popular search engine. We performed experiments by comparing the proposed approaches with a common baseline(a variation of rocchio algorithm). Our experiments have shown intesting results and have outperformed the baseline.

We also describe some of our interesting studies and observations from a query log of a popular search engine which are closely related to the problem addressed in this thesis. The study from the query log has also motivated us to see if clickthrough data can be created in an artificial way by simulating behaviour of users searching a

search engine. This provides a scope for a simulated environment for evaluation for personalized search algorithms. It is a potential area where the outcome of research can directly be used for the benefit of research communities in web search engines and personalization. We developed a basic system which performs the same.

The contributions from this thesis include: a suite of algorithms of personalized web search, our analysis and observations from query log study and a method to simulate user search behaviour and create user feedback in an artificial way. Also several observations were made from several studies during this thesis which provide interesting directions to personalized search research. Two important such observations are i) Can Personalization of Search be done without Relevance Feedback ? and ii) Can Simulation of User Search Behaviour be done.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

1.1 MOTIVATION

There has been a tremendous growth in the amount of information on the web. Information retrieval systems are critical for overcoming this information overload and providing the information of interest to users of the systems. Users typically pose a short query consisting of a few keywords describing their information need. Information Retrieval systems perform a 'word to word' match of the query words with all documents in their document collection and return documents containing the words entered. Retrieval in a web scenario is much more harder due to the large and dynamic content on the web.

Major web search engines usually cater to hundreds of millions of users and hundreds of millions queries every day. It is very unlikely that the millions of users are similar in interests and search for similar information. Also, it is probable that the query words entered by users exhibit polysemy (same word used in different senses like 'Java' can be used to mean Java the programming language or Java islands in Indonesia) and synonymy (different words can be used to convey similar information like OOP and Object Oriented Programming) due to ambiguous nature of natural language. Therefore, given different backgrounds of users, different interests of users and ambiguities in natural language, it is very likely that query words of two different users may appear exactly same even though information needs are different. However, current retrieval systems perform a 'word to word' match of the query words and work in a "one size fits all" fashion using the same search procedure for all the users. This makes the current retrieval systems far from optimal. This inherent non-optimality is

seen clearly in the following three cases: (1) When a query contains ambiguous terms : Different users may use exactly the same query (e.g., "Java") to search for different information (e.g., the Java island in Indonesia or the Java programming language), but existing IR systems return the same results for these users. Without considering the actual user, it is impossible to know which sense "Java" refers to in a query. (2) When a query contains partial information: A query can contain an acronym or a shorter usage of a longer phrase. Then there might not be sufficient information required to infer information need of user. For example a query like "SBH" can mean "State Bank of Hyderabad" or "Syracuse Behavioral Healthcare" among others. Existing IR systems return mixture of results containing the exact word which might contain different expansions. Knowledge of interests and/or location of the user could be helpful in gathering more information required to understand the query. (3) When information need of the user changes: A users information needs may change over time. The same user may use "Java" sometimes to mean the Java island in Indonesia and some other times to mean the programming language. Without recognizing the search context, it would be again impossible to recognize the correct sense. Thus using user context information about user and query is necessary for improving the retrieval performance. Indeed, personalized search essentially boils down to capturing and exploiting related user context information of a query to improve search accuracy.

1.2 BACKGROUND

Current retrieval systems (or search engines) return a long list of results obtained by 'word to word' match with query words. However, it has been observed that users typically view only top few (usually 10) documents out of the long list of results returned by search engines. This requires retrieval systems to show the most relevant documents to a user on the top to improve user satisfaction with the search engine. However, without knowledge about the user context, this task is difficult to do because "relevance" of a document depends on the individual user and the individual query.

	Short term (dynamic)	Long term (static)
Explicit	immediately judged relevant document	hobbies, occupation interests
Implicit	immediately clicked document	query log

Table 1.1: Classification and examples of user context

The goal of personalized search is to *customize* or *personalize* the search results returned by a search engine according to each individual user. There are two main challenges in personalized search. The first is to capture and model user context information. The second is to use the captured model and tune search results according to the given user.

Many kinds of user context information can be potentially exploited [20] as shown in Table 1.1. Explicit context consists of information given by a user explicitly, whereas implicit context refers to any context information naturally available while a user interacts with a retrieval system. Relevance feedback [39] can be considered as a way for a user to provide more context of search and is known to be effective for improving retrieval accuracy. Explicit user context provided by the user is called explicit relevance feedback (eg: marking documents as relevant or websites as interest etc). While explicit context information is more reliable than implicit context, it is often not available because it requires extra effort from the user. Implicit context information is thus more interesting to exploit.

For this reason, implicit feedback has attracted much attention recently [22, 24, 8, 56]. In general, retrieval results using users initial query may not be satisfactory; often, the user would need to revise the query to improve the retrieval/ranking accuracy. For a complex or difficult information need, the user may need to modify his query and view ranked documents with many iterations before the information need is completely satisfied. In such an interactive retrieval scenario, the information naturally available to the retrieval system is more than just the current query and the document collection: In general, all the interaction history can be available to the retrieval system, including past queries, information about which documents the user has chosen to view, and even

how a user has read a document (e.g., which part of a document the user spends a lot of time in reading). We define implicit feedback broadly as exploiting all such naturally available interaction history to improve retrieval results. This is also called *search history* or *clickthrough history*. Logs of such clickthrough history collected are also called *Query logs*.

1.3 PROBLEM DESCRIPTION

The main problem with web search is as follows : There is too much information available on the web; query words used by users often are confusing, ambiguous (a query "Java"can mean the Java island in Indonesia or the Java programming language), and some times are poor descriptors of information need ('SBH" can mean "State Bank of Hyderabad" or "Syracuse Behavioral Healthcare" among others) ; users are often not patient enough to see long list of results given by search engines to find relevant information. It has been observed that users typically only view top few results, usually top 5 or 10, some times 20 and much fewer times 30 and so on. In this scenario, web search can be made more useful, effective and less burdensome to users if the results are customized to each individual user by considering individual user's idiosyncrasies. This could possibly help the user to find the relevant document easily rather than having to scroll through a long list of results.

The problem of *Personalized Search* aims to customize search results according to each individual user by considering his idiosyncrasies. The goal is to show the most relevant documents to a user among the top few documents. This would possibly satisfy them and help in finding relevant information easily and quickly.

1.3.1 Issues

Some of the important issues related to the problem of personalized search are as follows.

1. *What to use to Personalize*

   What is the information to be used for personalization.

   It has been believed that the past user context can be an important information to learn about the user. The past user context can either be short term or long term. Short term user context like the immediately clicked/printed/saved /book marked document, or document immediately marked as relevant. Long term user context is referred to as the user context ranging over a longer period of time. It can consist of log of all the past search history consisting of past queries, click through history etc. Much better ways of obtaining richer context are being studied in literature.

2. *How to Personalize ?*

   How to actually go about to personalize?

   There are mainly two challenges here. 1) How to collect past user search contexts, 2) how to use it to learn to personalize the future searches of the user.

3. *When not to Personalize ?*

   When is personalization of search to be done. Can we assume that all the users would want their searches to be personalized for all the queries? Can there be some cases where their searches are not to be personalized. When should we stop personalizing.

4. *How to know Personalization helped?*

   Another important challenge is the lack of evaluation framework for personalized search. To the best of our knowledge, there is no standard publicly available test collections for evaluation of personalized search algorithms. The creation of sharable test collections facilitates discovery and allows for more rapid progress since building a good test collection is such a difficult, laborious, and time-consuming task. Standard test collections also allow for multiple modes of inquiry including those that involve the comparison of various techniques, examination of alternative hypotheses and replication of previous findings. Indeed

it has been identified that there is a great need for such standardization [2].

Also, there are no standard metrics for evaluating personalized search approaches and systems. Metrics commonly used in Information Retrieval like Precision, Recall, Mean Reciprocal Rank(MRR) etc. are used. These metrics do not take a user or a user's experience with the search engine into account while evaluating the performance. There is need for better metrics which can better evaluate personalized search systems on how the information provided was satisfactory to the user.

Each issue discussed in this sub section is an interesting challenge ahead in the area.

1.3.2 Problem Statement

This thesis focuses on personalized web search. In particular, *how to learn to personalize for the future searches of the user, using his past search history.* The problem is divided into two sub problems. The first, is how to learn a user model representing the user interests, from his past search history. The second is how to improve the web search using the user model.

1.4 SOLUTION OUTLINE

In this thesis, we studied the problem of personalized search. While there is some existing related work, it is far from optimal. We attempt to address the issue *How to Personalize* (See Section 1.3.1 for all the issues).

The major challenges for personalized search are two fold. The first is, modeling appropriate user context and learn a user model. The second is, how to utilize the user model to improve search accuracy. Another important challenge is evaluation of experiments. There are no standard and bench mark datasets available on which experiments can be performed. This makes comparison with earlier work in

the literature and replicating their results difficult. There are also no standard metrics available to effectively evaluate personalized search algorithms. Commonly used metrics in Information Retrieval systems are usually used.

The first challenge is modeling the user context. Modeling the user context itself is a huge research area called *User Modeling*. In this thesis, user context refers to the context of past searches made by the user. Many kinds of context information can be potentially exploited. For example, *explicit context, implicit context, short term context, long term context* etc. *Explicit context* consists of information given by a user explicitly like one or more documents explicitly marked by a user to be relevant to him. *Implicit context* refers to any context information naturally available while a user interacts with a retrieval system. For example, if a user clicks a result among the list of results given for a query, the clicked result was probably found to be interesting to the user. While explicit context information is more reliable than implicit context, it is often not available because it requires extra effort from the user. Due to this, implicit context information became an interesting alternative. *Short term context* refers to short term or immediate search need of the user. For example, a user search for restaurants in the near by area during lunch time or searching for gifts for birthday etc. . The short term search context has shorter life and is dynamic. On the other hand, *Long term context* refers to search needs spanning over a longer period of time. For example, consider a user who is a researcher interested in 'artificial intelligence'. He would pose queries related to it probably to keep him updated or know more about it. The same user might have searched for some restaurants in a particular area during lunch time. The searches related to 'artificial intelligence' can been seen to be long term context and the searches related to restaurants can be seen as short term context. This is because, once lunch time is over, the information need is over. In fact, exact definitions of long term and short term context itself is not established. It is often assumed that short term context typically lasts for a few hours or a day and long term context spans over longer period of time i.e., over a few months. In this thesis, we focus on capturing long term implicit search context. In doing so, we use the past

search history namely the queries and their corresponding clicked documents.

The second major challenge identified is: how to appropriately use the user model to improve search results. One way to do this is to embed the user model in the information retrieval process. However, this becomes a difficult task in case we want to perform experiments on the web because we do not have access to the retrieval function of major search engines operating on the web. Also, modifying retrieval function for each and every user is difficult. The main goal of improving search results is to show more relevant results to the user on the top few results because a user mostly sees only the top few (typically 10) results. Our approach of re-ranking is as follows: We first retrieve results for a query from a major search engine and consider the top few results and then compute a score for each document from the top few results based on the user profile of the corresponding user. Then, we re arrange the documents in descending order of the score.

We propose three approaches for personalizing of search results. Each of the three approaches, has two phases. The first phase is called *User Profile Learning* or *User Modeling*. We learn a model representing a user's potential interests from his past search history. The first two approaches are based on utilizing the past search history consisting of the past queries and their corresponding clickthroughs. The third approach attempts personalization based on this past queries alone. We focus on long term user context obtained through implicit feedback. Since it is difficult to define long term context, we assume that it spans over a few months say 5 to 6 months. The second phase is called *Reranking* where user model learned is used to personalize search results.

Our first approach is based on Statistical Language Modeling techniques. We propose two methods based on language modeling techniques. The first is a simple method which captures the statistical properties of how words are distributed in user contexts. The second method is based on Noisy Channel model which aims to capture and model the relationship between the query and document words [42].

Our second approach is based on Machine Learning algorithms [44, 59]. We make

use of ranking SVM, a variation of the classification support vector machines for learning user model.

The above two approaches are a class of approaches that have exploited user feedback data either implicit and explicit. The third approach is another interesting method where we model user based on only the past queries posed by the user[41] without using the clickthroughs made by the user. This is beneficial to explore because, not only obtaining queries is an easier task but also, user feedback data becames stale especially in a web scenario due to dynamic web content.

In all these approaches, for reranking the results, we followed a common set up. User query is posed to a web search engine (Google) and top few results matching the query are retrieved. These top results are then reranked using the user profile[1].

We performed experiments on data extracted from query logs collected over 3 months by a popular search engine. We performed our experiments on 17 users from the query log. The first two months of data is used to learn a profile and the third month data is used for evaluation. Our experiments showed that our approaches outperformed the baseline with a wide margin.

Our study of the same data has lead to some interesting observations closely related to the problem addressed in this thesis. This has also motivated us to see if clickthrough data can be created in an artificial way by simulating behaviour of users searching a search engine [40]. This provides a scope for a simulated environment for evaluation for personalized search algorithms. It is a potential area where the outcome of research can directly be used for the benefit of research communities in web search engines and personalization. Given the constraint of non-availability of implicit feedback data it is difficult to evaluate personalization algorithms. Simulated Feedback can be of great use here and help benefit web search community. The benefit from Simulated feedback is two fold. Firstly, it is easy to obtain and also the process of obtaining the feedback data is repeatable. Given a document set and a search engine deployed on this document set, one can start generating simulated feedback in much larger volumes

[1]This is a set up used in many other earlier works See [57]

than what can be obtained by either explicit or implicit feedback methods. Secondly, it enjoys the benefit of customizability, where a researcher can customize the creation of the feedback for his purposes, be it testing search engines for specific domains, testing research algorithms in personalization research or testing query modification techniques. We developed a basic system which performs the same.

1.4.1 Contributions

The contributions from this thesis include a suite of approaches of personalized web search, our analysis and observations from query log study and a method to simulate user search behaviour and create user feedback in an artificial way.

- Basic IR approaches

  We implemented some basic Information retrieval methods consisting of indexing and retrieval algorithms based on Nutch and Lucene. Also API calls to Internet web search engines have also been created which has been customized to Google.

- Proposed and Baseline personalization approaches

  We implemented the approaches proposed in this thesis and also the baseline(s) used in this work.

- Evaluation metrics for IR and personalized search

  Programs for automatic evaluation of IR and Personalized search systems using standard IR metrics are implemented.

- Simulated Feedback Creation

  The simulated feedback is an artificial way of creating relevance feedback given by users with motivations from real user studies. A basic version of the system is created.

Also several observations were made from several studies during this thesis which provide interesting directions to personalized search research. The two important such

observations are i) Can Personalization of Search be done without Relevance Feedback ? and ii) Can Simulation of User Search Behaviour be done.

1.5 ORGANIZATION OF THE THESIS

The rest of the thesis is organized as follows. In Chapter 2, we give an overview of some personalized search approaches proposed in the literature. In Chapter 3, we describe the thesis outline consisting of an overview of the suite of the personalized search approaches proposed. In Chapters 4,5 and 6 we describe each of the approach in detail. In Chapter 7, we give the experimental evaluation and in Chapter 8 we describe our Conclusions also throwing light on future directions. We also did a study of a query log from a popular search engine. Our query log study has lead to some interesting observations which we believe could be useful to work on web search and personalization. It is described in Appendix I. Some observations from our study has motivated us to see if user web search behaviour can be simulated using the observations from our query log study. We developed a basic system which performs the same and is described in Appendix II.

CHAPTER 2

Review of Personalized Search

There has been a vast and growing literature with regard to personalized search. In a recent workshop in 2002 about challenges in information retrieval and language modeling [2], personalized and contextual search is considered as one of the two grand challenges in information retrieval.

In this section, we briefly describe some work related to personalized search. The approaches are grouped based on the common resources used or learning methods employed.

2.1 QUERY LOGS

Search Query logs consist of logs of searches made by users of search engines. They are usually collected at the search engine server. They typically consist of : user identity (ip address or anonymous id etc), search queries, corresponding clickthroughs made by the user and click information regarding it like the click time, no of clicks made etc. Some times the query logs are also captured on the client side i.e., on the user's computers. Clickthrough data/Query logs have been the most important source for capturing user context for user modeling. There has been some work in this connection some of which are described below.

In [56], Sugiyama et. al used web browsing history in past N days for personalized search. They partition the browsing history data into three categories according to the time stamp, i.e., persistent data (before today), today data (today but before the current session) and current session data. They found that the performance of using web browsing history is competitive with that using relevance feedback. Speretta et.

al[55] also used users search history to construct user profiles. Several other works have made use of past queries mined from the query logs to help the current searcher. (see [[37], [21], [11], [13]]).

2.2 MACHINE LEARNING BASED APPROACHES

Machine learning algorithms have received a wide attention recently to learn functions that can perform desired operations when trained on required amount of data. Personalized search is one such potential problem where the use of machine learning can be made. Given the previous history of the user, can we learn a model representing the user. This makes user modeling a perfect application for machine learning.

Most work on machine learning especially in information retrieval simplifies the task to a binary classification problem with two classes relevant and non-relevant. Such a simplification has several drawbacks. Firstly, because, often when a user gives relevance feedback, he only gives information of documents which he thinks are relevant to him. The user typically does not give information about the documents not relevant to him. For example, assuming all the documents not identified as relevant to be irrelevant, leads to problems. Due to a strong majority of non-relevant documents, a learner will typically achieve the maximum predictive classification accuracy, if it always responds non-relevant, independent of where the relevant documents are ranked. Consider implicit relevant feedback - clickthrough data, it has been shown that users typically click documents which appear to be relevant in the top few results. A click only implies that, the clicked document appeared to be more relevant than the documents not clicked. Hence, clickthrough data contains partial information and it would be an over simplification to assume a binary classification. Joachims et al[22, 36] has proposed a new machine learning algorithm Ranking SVM, a variation of the Support Vector Machine learning algorithm which can learn from the partial feedback data present in the clickthrough data of users.

In [58], Teevan et. al use desktop search index as the user profile for personalized

search. They used rankingNets to learn the user profile. They consider the user profile as the implicit feedback and incorporate them into the ranking of web search results. They test different combination of corpus representation, user representation and document representation. It is found that the combination of personalized search ranking and original web ranking can achieve better results than the original ranking.

2.3 LANGUAGE MODELING BASED APPROACHES

Statistical language modeling for information retrieval (IR) that has emerged within the past several years as a new probabilistic framework for describing information retrieval processes. Generally speaking, statistical language modeling, or more simply, language modeling (LM), refers to the task of estimating a probability distribution that captures statistical regularities of natural language use. Applied to information retrieval, language modeling refers to the problem of estimating the likelihood that a query and a document could have been generated by the same language model, given the language model of the document and with or without a language model of the query. There has been a lot of work applying LM to IR and related applications. [35, 5, 54, 25, 64, 26] etc.

In spite of the progress made in language modeling and IR recently, there has not been much work to explicitly capture information about the user and context of the information retrieval process, and integrate models of users into the retrieval models. Indeed this has been identified as one of the long term goals [2].

Shen et. al [47, 48] proposed a decision theoretic framework for implicit user modeling for personalized search. They consider the short term context in modeling user. A language model is computed from the short term history and is used to improve the retrieval performance. In [57], long term search history of the users is mined and language models are computed which are then used to improve retrieval performance.

2.4 COMMUNITY AND COLLABORATIVE FILTERING BASED APPROACHES

Recently approaches motivated by social networking have been proposed for improving the web search results. It is assumed that there exists a community of users who have similar interests. The search history of the users in the community is saved. The query repetition among the users within the community is exploited. Similar queries and their corresponding clicked documents are used to either recommend documents or is used in improving search results. Different approaches have been proposed varying the way communities are defined and employing different learning techniques for discovering communities, user and community profiles.

Chidlovski et. al [7] describes the architecture of a system performing collaborative re-ranking of search results. The user and community profiles are built from the documents marked as relevant by the user or community respectively. These profiles essentially contain the terms and their appropriate weights. Re-ranking of the search results is done using the term weights using adapted cosine function. The search process and the ranking of relevant documents are accomplished within the context of a particular user or community point of view. However the paper does not discuss much about the experimental details. Lin et. al [28] presented an approach to perform personalized web search based on PLSA, Probabilistic Latent Semantic Analysis, a technique which stems from linear algebra. They extracted a co-occurrence triple consisting of a user, query, and corresponding web pages viewed for a query, by mining the web-logs of the users and modeled the latent semantic relationship between them using PLSA. Armin Hust [19] performed query expansion by using previous search queries by one or more users and their relevant documents. This query expansion method reconstructs the query as a linear combination of existing old queries. The terms of the relevant documents of these existing old queries are used for query expansion. However, the approach does not take the user into account. In [[51], [53], [52], [4], [52], [12]], a static community is assumed. Users can either join the existing communities

or create a new community. All the queries and clicked URLs by all the users in the community are saved. When a user in a particular community poses a query, all the previous queries and clicked documents from the same community are made use of. Similarities between the present query and the past queries from all the users within the community are measured and all those queries and their corresponding clicked documents whose similarity is more than a threshold are considered. Results that have been reliably selected for similar queries in the past are promoted. Uppuluri and Ambati [43] proposed an approach for re-ranking of search results in a digital library scenario. The user profiles were constructed from the documents marked as relevant or irrelevant. Re-ranking of the results is done using the user profile and profiles of others users in the community. They assumed and assigned a set of static communities for each user which the user has selected while registering with the system. Also, the user also selects the community before posing the query and the re-ranking is done based on the community selected.

2.5 OTHER APPROACHES

Pretschner [1] used ontology to model a users interests, which are studied from users browsed web pages. Liu et. al [29] performed personalized web search by mapping a query to a set of categories using a user profile and a general profile learned from the user's search history and a category hierarchy respectively.

2.6 COMPARISON WITH OUR WORK

Although some work has been done in personalized web search, the work is far from optimal. To the best of our knowledge, the work done by Shen et. al [47, 48] is the only published and known work done applying language modeling techniques to personalized search. Most of their work focused on modeling short term context. They employed simple language modeling techniques capturing the statistical properties of

user context from single words in text for modeling short term user context. Our work focuses on modeling long term user context. We propose an approach to capture statistical properties of user context for modeling long term user context from single words in the text data. We also extend the approach by trying to capture richer context by using bigrams i.e., two words appearing together in text. We believe bigrams provide richer context than just single words. We also aim to capture and model the relationship between query and document words which we believe captures much richer context. We propose an approach based on Noisy channel model for the same.

Machine learning methods provide an interesting learning framework for learning user context. However, not much work has been done using machine learning methods for personalized search. We propose an approach for personalized search using Ranking SVM. We experimented with different features for learning the SVM model.

To the best of our knowledge, previous work on personalized search has assumed and focused on modeling the user from the relevance feedback provided by him either explicitly or implicitly. We believe that the queries posed by the user also convey some user context information though in a concise way. To our knowledge, this resource has not been exploited well so far. We show how the past queries posed by the user can be effectively used to personalized search. The results from this research can be extremely useful because obtaining just the queries is comparatively easy than relevance feedback either implicit or explicit. The queries can then be used in directly for personalization or in combination with available relevance feedback.

CHAPTER 3

I Search: A Suite of approaches for Personalized Web Search

In this thesis, we propose a suite of approaches for personalized web search called *I Search*. *I Search* means search for *I* to mean personalized search.

I Search consists of three proposed approaches for personalized web search. Each of the three approaches, has two phases. The first phase called *User Profile Learning* or *User Modeling*. In this phase, we learn a model of the user representing his potential interests from the past search history. The first two proposed approaches use the user's implicit relevance feedback or user clickthroughs. They are based on utilizing the past search history consisting of the past queries and their corresponding clickthroughs. The third approach attempts personalization based on user's past queries alone without having to make use of user clickthrough data. We focus on long term user context obtained from clickthroughs. Since it is difficult to define long term context, we assume that it spans over a few months say 5 to 6 months. The second phase is called *Reranking* where the user model learned is used to personalize the search results.

In rest of the chapter, we give an overview of each of our approach.

3.1 PERSONALIZED SEARCH WITH USER'S RELEVANCE FEEDBACK

In the first two approaches, user's implicit relevance feedback is used to learn a model representing the user's interests. In doing so, the past queries and their corresponding clicked documents are used. The first approach is based on statistical language modeling techniques. The second approach is based on machine learning and uses ranking SVM.

3.1.1 Language Modeling based Approaches

Statistical language modeling for information retrieval (IR) has emerged within the past several years as a new probabilistic framework for describing information retrieval processes. Generally speaking, statistical language modeling, or more simply, language modeling (LM), refers to the task of estimating a probability distribution that captures statistical regularities of natural language use. Applied to information retrieval, language modeling refers to the problem of estimating the likelihood that a query and a document could have been generated by the same language model, given the language model of the document and with or without a language model of the query. There has been a lot of work applying LM to IR and related applications [35, 5, 54, 25, 64, 26] etc.

In spite of the progress made in language modeling and IR recently, there has not been much work to explicitly capture information about the user and context of the information retrieval process, and integrate models of users into the retrieval models. Indeed this has been identified as one of the long term goals [2].

We propose two methods based on language modeling for personalized search.

3.1.1.1 Simple N-gram based method

The first approach is a simple approach based on language modeling techniques. The approach is to capture statistical properties of the user data from user context from his previous searches. Given a new instance of data, these statistical properties captured can then be used to see how likely are they to be liked by the user. This is a simple yet effect approach and has been used recently for a number of applications.

The approach is to first analyze the text from the past user contexts and capture statistical properties of words in the text. We use the user past search history consisting of the past queries and clicked documents in learning the same. The training data consists of all the words in snippets extracted from the clicked documents (short snippet of the documents similar to what is display typically by search engines con-

sisting of a few words around the query words in the document). A language model is learned from the user training data by computing the probabilities of occurrence of all the words in the user training data. This language model constitutes the user profile.

Ranking in language model framework in general is done by estimating which is the most likely document that generated this query. This is computed by calculating in which document, the probability of occurrence of the query words (done by multiplying the probability of occurrence of all the query words in the document) is maximum. In our approach in the reranking phase, given a query, a bag of documents matching the given query are retrieved by a search engine (ex: Google). Top few documents from the retrieved documents are considered and a score is computed. The score measures which is the most likely document for the given query and user. This is computed by calculating a score as a weighted combination of the probability of the word in the document and the probability of the word from the user profile. The documents are then sorted in descending order of this score.

We experimented with language models using unigrams and bigrams. Unigrams are single words occurring in the text. Bigrams are pairs of words occurring adjacent together. It is believed that bigrams carry more information than unigrams. This is because, a word occurring together with another word is less common than the word occurring in the same text, so when a pair of words are seen together a pattern can be inferred. Our experiments have shown similar results with the bigram language model based user profile outperforming unigram based language model based user profile.

3.1.1.2 Noisy Channel model based method

It has been pointed out that queries and documents share different information spaces. Queries are often short, concise, brief and depend on the linguistic usage of the user. Documents are more descriptive. However, most of the methods either to retrieval or to personalized web search do not consider the same. It is our assumption that by modeling them as belonging to different spaces and capturing the relationship between

the query and document words, we can improve search performance. We model query and document differently using a noisy channel model. The scenario is explained as follows. When a user has an information need, a user has an ideal document in mind, a document which contains the information he requires. With this ideal document in mind, he formulates a query and poses it a search engine. The formulation of a query is the result of "converting" the content in the ideal document into a set of query words. Some of the words from the ideal document could be omitted, some words changed and translated etc in formulating the query. This can be viewed as a distillation process or process of translation of content in one language (document language) to content in another language(query language). The ideal document can be considered to be at the source end and queries at the target end and the information at source end is believed to be transmitted through the noisy channel to the target end. The information undergoes slight transformation or translation in the transmission through the noisy channel. By carefully observing how the information need is getting translated into queries, we can estimate the channel probabilities (i.e., how a source word is getting translated into a target word). In this way, we can capture the relationship between query words and document words. Given a new query, we can now estimate which document from a set of documents is most likely to the ideal document.

This problem draws similarities to statistical machine translation where a sentence in one language is translated to another language while passing through a noisy channel. The channel probabilities are computed which contain information about how a word in the source language is getting translated into another a word in another language. A translation model is constructed from the same which contains a source word (word in source language), a target word(word in target language) and the probability of translation of the source word to the target word. The channel probabilities are computed using statistical techniques using Expectation-Maximization(EM) algorithm [6] from "parallel texts" consisting of sentences in source language and corresponding sentences in target language. When a new sentence in source language is given, it is translated into target language using the translation model.

In our approach, we use the past search history of the user. A clicked document is assumed to be relevant and an approximation of an ideal document. Given logs of past queries and corresponding "approximate" ideal documents, we can now estimate how an ideal document is being translated to an ideal query and thereby capture the relationship between query and document words. By considering the ideal documents and queries as parallel texts i.e., ideal documents as texts in one language and queries as texts in another language, we compute a translation model using statistical machine translation algorithms. The translation model thus learned constitutes the user profile. In the Reranking phase, given a query, a bag of documents matching the given query are retrieved using a search engine (ex: Google). Top few documents from the retrieved documents are considered and a new score is computed. The new score measures how much the word translates into query word and its probability of occurrence in the document. The documents are then sorted in descending order of this score. The better the relationship between a query and document words, the higher the new score, the better the document and the better the document is supposed to be.

3.1.2 Machine Learning based Approach

Machine learning algorithms have received a wide attention recently to learn functions that can perform desired operations when trained on required amount of data. Personalized search is one such area to which the application of machine learning can be beneficial. Given the previous history of the user, can we learn a model representing the user. This makes user profile learning a perfect problem for application of machine learning.

Most work on machine learning especially in information retrieval simplifies the task to a binary classification problem with the two classes relevant and non-relevant. Such a simplification has several drawbacks. Firstly, because, often when a user gives relevance feedback, he only gives information of documents which he thinks are relevant to him. The user typically does not give information about the documents not relevant

to him. For example, assuming all the documents not identified as relevant to be irrelevant leads to problems. Due to a strong majority of non-relevant documents, a learner will typically achieve the maximum predictive classification accuracy, if it always responds non-relevant, independent of where the relevant documents are ranked. Consider implicit relevant feedback - clickthrough data, it has been shown that the user always clicks documents which appear to be relevant in the top few results. In this case, the user has clicked those results because they might have appeared to be more relevant than the others seen by him in the top few. Hence, clickthrough data only contain partial information and it would be an over simplification to assume a binary classification.

Joachims et al[22, 36] has proposed a new machine learning algorithm called Ranking SVM, a variation of the Support Vector Machine learning algorithm which can learn from the partial feedback data present in the clickthrough data of users as mentioned above for improving web search. Ranking SVM learn from preference constraints given by the user. For example, for learning retrieval functions, they consider the preference constraints i.e., for a given query, the current document is more relevant than the previous document. The optimization problem for Ranking SVM is similar to classification SVM on for such pair of constraints. The output after training is a model containing the alpha values and support vectors similar to classification SVM.

We propose an approach for personalized search in this framework. We learn user model by training a model using Ranking SVM from the clickthrough history of the user. The input to the Ranking SVM is pairs of preference constraints inferred from the clickthrough history. Preference constraints are the constraints specified by the user. For each query, with at least 1 clicked document, a few preference constraints are constructed. It is believed that clicked documents are more preferred than unclicked documents because a user only might have clicked documents because he thought it might be relevant. However, an unclicked document were not clicked because the user did not consider relevant in comparison to a clicked document. Preference constraints are constructed for each pair of clicked and unclicked document for a given query

specifying that a clicked document is more preferred than an unclicked document. We used SVM^{light}[1] for training SVM.

It is believed in the Machine Learning literature that the learned model depends on the choice of features and feature weights. We considered all the words in snippets [2]. We experimented with different features for training the SVM model. The first binary - each feature is either present or absent giving a value 1 to features present and 0 to the ones not present. But, sometimes this cannot capture the rich information contained in the repetition of words. We experimented by considering the frequency of the times each feature occurred. We also experimented with normalized frequency of the words. After training, we obtain a model consisting of the α values and the support vectors. We compute the weight vector from the same (We only consider linear kernel).

Reranking is done using this weight vector. Given a query, a bag of documents matching the given query are retrieved by an underlying search engine (Google). Top few documents from the retrieved documents are considered and a new score is computed. The score measures a match between the weight vector and the document. The documents are then sorted in descending order of this score. The more the score, the better match between the user interest and the document and hence the better the document.

3.2 PERSONALIZED SEARCH WITHOUT USER'S RELEVANCE FEEDBACK

In this approach, we investigate to see what information do the queries carry. To see if, a log of just the past queries can serve as a source to learn a user model. The

[1] A machine learning software that performs SVM training and is widely used in machine learning literature and available for free for research purposes.

[2] short snippet of the documents similar to what is display typically by search engines consisting of a few words around the query words in the document

results from this research can be extremely useful because obtaining just the queries is comparatively easy than relevance feedback. The queries can then be used directly for personalization or in combination with available relevance feedback. To the best of our knowledge, this study has not been explicitly carried out.

We experiment to learn user model from his past queries alone using simple language modeling techniques.

3.2.1 Simple Language Modeling based Method

We considered a simple N-gram based language model. Language model is constructed from just the past queries of the user consisting of words and their probabilities of occurrence in the string concatenation of all the past queries. This constitutes the user profile.

In the Reranking phase, given a query, a bag of documents matching the given query are retrieved by a search engine (ex: Google). Top few documents from the retrieved documents are considered and a new score is computed. The score measures which is the most likely document for the given query and user. This is computed by calculating a score as a linear combination of the probability of the word in the document and the probability of the word from the user profile. The documents are then sorted in descending order of this score.

In this chapter, we described *I Search*, a suite of proposed approaches for personalized web search. In the following chapters, we describe each approach in detail.

CHAPTER 4

Personalized Search using User Relevance Feedback: Statistical Language Modeling based approaches

4.1 INTRODUCTION

Statistical language modeling for information retrieval (IR) has emerged within the past several years as a new probabilistic framework for describing information retrieval processes. Generally speaking, statistical language modeling, or more simply, language modeling (LM), refers to the task of estimating a probability distribution that captures statistical regularities of natural language use. Applied to information retrieval, language modeling refers to the problem of estimating the likelihood that a query and a document could have been generated by the same language model, given the language model of document and with or without a language model of query.

The root of statistical language modeling dates back to the beginning of the 20^{th} century when Markov tried to model letter sequences in works of Russian literature [30]. Zipf [65, 66, 67, 68] studied statistical properties of text and discovered that the frequency of words decays as a power function of its rank. However, it was Shannons work [46] that inspired later research in this area. In 1951, eager to explore the applications of his newly founded information theory to human language, Shannon used a prediction game that involved n-grams to investigate the information content of English text. He evaluated n-gram models performance by comparing their cross-entropy on text with the true entropy estimated using predictions made by human subjects. For many years, statistical language models have been used primarily for automatic speech recognition. Since 1980 when the first significant language model

was proposed [45], statistical language modeling has become a fundamental component of speech recognition, machine translation, spelling correction, and so forth. It has also proven useful for natural language processing tasks such as natural language generation and summarization. In 1998, it was introduced to information retrieval and has opened up new ways of thinking about the retrieval process.

The first use of language modeling approach for IR focused on its empirical effectiveness using simple models. In the basic approach, a query is considered generated from an "ideal" document that satisfies the information need. The systems job is then to estimate the likelihood of each document in the collection being the ideal document and rank them accordingly. This query-likelihood retrieval model, first proposed by Ponte & Croft [35], and later described in terms of a noisy channel model by Berger & Lafferty [5], has produced results that are at least comparable to the best retrieval techniques previously available. The basic model has been extended in a variety of ways including modeling documents as mixtures of topics [18] and considered phrases [54]. Progress has also been made in understanding the formal underpinnings of the statistical language modeling approach, and comparing it to traditional probabilistic approaches. Connections were found and differences identified. Recent work has seen more sophisticated models developed that are more closely related to the traditional approaches. For example, a language model that explicitly models relevance [27] has been proposed by Lavrenko & Croft, and a risk minimization framework based on Bayesian decision theory has been developed by Lafferty & Zhai [25]. Successful applications of the Language Modeling(LM) approach to a number of retrieval tasks have also been reported, including cross-lingual retrieval [64, 26] and distributed retrieval [63, 49]. Research carried out by a number of groups has confirmed that the language modeling approach is a theoretically attractive and potentially very effective probabilistic framework for studying information retrieval problems [3].

In spite of the progress made in language modeling and IR recently, there has not been much work to explicitly capture information about the user and context of the information retrieval process, and integrate models of users into the retrieval models.

Indeed this has been identified as one of the long term goals [2].

In this thesis, we propose a few approaches to personalized search using statistical language modeling techniques. We first give a brief description of the work related to language modeling in the next section and then describe our approaches in detail in the rest of the sections.

4.2 BACKGROUND

A statistical language model is a probability distribution over all possible sentences or other linguistic units in a language [45]. It can also be viewed as a statistical model for generating text. The task of language modeling, in general, answers the question: how likely the i_{th} word in a sequence would occur given the preceding $i-1$ words? In most applications of language modeling, such as speech recognition and information retrieval, the probability of a sentence is decomposed into a product of n-gram probabilities. Lets assume that S denotes a specified sequence of k words,

$$S = w_1, w_2, ..., w_k$$

An n-gram language model considers the word sequence S to be a Markov process with probability

$$P(S) = \Pi_{i=1}^{k} P(w_i | w_{i-1}, w_{i-2}, w_{i-3}, \cdots, w_1)$$

where n refers to the order of the Markov process. When $n = 2$ we call it a bigram language model which is estimated using information about the co-occurrence of pairs of words. In the case of n=1, we call it a unigram language model which uses only estimates of the probabilities of individual words. For applications such as speech recognition or machine translation, word order is important and higher-order (usually trigram) models are used. In information retrieval, the role of word order is less clear and unigram models have been used extensively. The role of higher order models in Ir is yet to be carefully studied.

To establish the word n-gram language model, probability estimates are typically derived from frequencies of n-gram patterns in the training data. It is common that

many possible word n-gram patterns would not appear in the actual data used for estimation, even if the size of the data is huge and the value of n is small. As a consequence, for rare or unseen events the likelihood estimates that are directly based on counts become problematic. This is often referred to as the data sparseness problem. Smoothing is used to address this problem and has been an important part in any language model.

The basic approach for using language models for IR proposed by Ponte and Croft assumes that the user has a reasonable idea of the terms that are likely to appear in the ideal document that can satisfy his/her information need, and that the query terms the user chooses can distinguish the ideal document from the rest of the collections [35]. The query is thus generated as the piece of text representative of the ideal document. The task of the system is then to estimate, for each of the documents in the collection, which is most likely to be the ideal document. That is, we calculate $\arg\max_D P(D|Q)$. Where D is a document and Q is the query. Rewriting it using bayesian formualtion, we get

$$\arg\max_D P(D|Q) = \arg\max_D P(Q|D)P(D)$$

The prior probability $P(D)$ is usually assumed to be uniform for simplicity and $P(Q|D)$ is estimated for every document. This essentially boils the above equation down to that of computing $\arg\max_D P(Q|D)$. The problem now is compute which of the given set of document have mostly likely generated the given query. This actually is a re-phrasement of the original problem. In other words, we estimate a probability distribution over words for each document and calculate the probability that the query is a sample from that distribution. Documents are scored according to this probability. This is generally referred to as the query-likelihood retrieval model and was first proposed by Ponte & Croft [35]. In their paper, Ponte & Croft take a multi-variate Bernoulli approach to approximate $P(Q|D)$. They represent a query as a vector of binary attributes, one for each unique term in the vocabulary, indicating the presence or absence of terms in the query. The number of times that each term occurs in the query is not captured. There are a couple of assumptions behind this approach: 1)

the binary assumption: all attributes are binary. If a term occurs in the query, the attribute representing the term takes the value of 1. Otherwise, it takes the value of 0. And, 2) the independence assumption: terms occur independently of one another in a document. These assumptions are the same as those underlie the binary independence model proposed in earlier probabilistic IR work [38, 60]. Based on these assumptions, the query likelihood $P(Q|D)$ is thus formulated as the product of two probabilities the probability of producing the query terms and the probability of not producing other terms.

$$P(Q|D) = \Pi_{w \in Q} P(w|D) \Pi_{w \notin Q} (1.0 - P(w|D))$$

where $P(w|D)$ is the probability of occurence of the word w in the document D.

In contrast to Ponte & Crofts approach [35], Hiemstra [17], Miller et al. [31], and Song & Croft [54] employ a multinomial view of the query generation process. They treat the query Q as a sequence of independent terms (i.e. $Q = q_1, \cdots, q_m$), taking into account possibly multiple occurrences of the same term. The ordered sequence of terms assumption behind this approach states that both queries and documents are defined by an ordered sequence of terms [17]. A query of length k is modeled by an ordered sequence of k random variables, one for each term occurrence in the query. While this assumption is not usually made in traditional probabilistic IR work, it has been essential for many statistical natural language processing tasks (e.g. speech recognition). Based on this assumption, the query probability can be obtained by multiplying the individual term probabilities (assuming the query words are independent of each other)

$$P(Q|D) = \Pi_{i=1}^{m} P(q_i|D) \tag{4.1}$$

where q_i is the i^{th} term in the query and $P(q_i|D)$ is the probability of occurence of q_i in D. While through different theoretical derivations, these models all arrived at a similar way of computing $P(w|D)$ (with w denoting any term) - combining a component estimated from the document and one from the collection by linear interpolation

$$P(w|D) = \lambda P_{document}(w|D) + (1 - \lambda) P_{collection}(w) \tag{4.2}$$

where λ is a weighting parameter between 0 and 1. This can also be viewed as a combination of information from a local source, i.e. the document, and a global source, i.e. the collection.

Taking a different angle, Berger and Lafferty [5] view a query as a distillation or translation from a document. The query generation process is described in terms of a noisy channel model. To determine the relevance of a document to a query, their model estimates the probability that the query would have been generated as a translation of that document. Documents are then ranked according to these probabilities. More specifically, the mapping from a document term w to a query term q is achieved by estimating translation models $t(q|w)$. Using translation models, the retrieval model becomes

$$P(Q|D) = \Pi_{i=1}^{m} \sum_{w} t(q_i|w) P(w|D)$$

A notable feature of this model is an inherent query expansion component and its capability of handling the issues of synonymy (multiple terms having similar meanings) and polysemy (the same term having multiple meanings). However, as the translation models are context independent, their ability to handle the ambiguity of word senses is only limited. While significant improvements over the baseline language model through the use of translation models is reported, this approach is not without its weaknesses: the need of a large collection of training data for estimating translation probabilities, and inefficiency for ranking documents.

4.3 SIMPLE N-GRAM BASED APPROACHES

In this section, we describe our approaches to personalized search based on simple N-gram based approaches. We capture the statistical properties of text from past user search contexts. The approaches are based on Query likelihood model using unigram (i.e., single words in text) and bigram (i.e., two adjacent words in texts). We learn user profiles from the user's past history using the unigrams and bigrams. Reranking is done using these appropriately for each approach. In the rest of this section, we

describe the same.

4.3.1 Learning user profile

Learning the user profile involves computing a unigram or bigram language model accordingly from the relevance feedback provided by the user. A language model is a probabilistic distribution of either unigrams or bigrams accordingly learned from text. It consists of words and their corresponding probabilities. The user profile comprises the language model learned.

We first collect the feedback provided by the user for all the queries and compute a concatenation of the feedback. Let H_u represent the search history of the user u. H consists of $\{(q_1, rf_1), (q_2, rf_2), (q_3, rf_3), ...(q_n, rf_n)\}$ where q_1 is the past query posed by the user and rf_1 is the text of the relevance feedback given by the user. We compute a concatenation of all the past relevance feedback from the user called rf_{all}. User profiles using unigrams and bigrams from texts are learned as described below.

4.3.1.1 Using Unigrams

In case of unigrams, the user profile consists of unigrams i.e., single and their respective probabilities. All unique unigrams i.e., single words in the texts are extracted and their frequency of occurrence in the text is computed from rf_{all}. Probability of occurrence of each unique word is computed by dividing with the total number of words. The probability is computed as follows:

$$P_{w_i} = \frac{c(w_i|rf_{all})}{\sum_{w_i \in rf_{all}} c(w_i|rf_{all})}$$

where w_i is a unigram in rf_{all}, P_{w_i} is called the probability of occurrence of the unigram w_i and $c(w|rf_{all})$ is the frequency of occurence of w in rf_{all}. The user profile consists of

$$\{(w_1, P_{w_1}), (w_2, P_{w_2}), ..., (w_i, P_{w_i}), ...(w_n, P_{w_n})\}$$

A sample user profile is as shown in Figure 4.1.

w	P(w)	w	P(w)
PlayStation	0.0012	Fantasy	0.0012
RubySapphire	0.0024	Origin	0.0012
Note	0.0012	FireRed	0.0012
America	0.0024	available	0.0012
Hints	0.0012	deoxes	0.0012
codebreaker	0.0024	records	0.0012
access	0.0012	GameSpot	0.0023
Catch	0.0011	Ruby	0.0059

Table 4.1: Sample unigram user profile

4.3.1.2 Using Bigrams

In case of bigrams, the user profile consists of bigrams (pair of words occurring adjacent to each other) and their conditional probabilities. The conditional probability between a pair of words measures how probable is the second word given the first word. It helps us to capture the relationship between a word and its previous word and measures probability of dependence of word on its previous word. It is computed by dividing the number of times the pair of words occurred together with the total number of times the first word occurred. We extract unigrams and bigrams and count their frequency of occurrence respectively. The conditional probability is computed as follows:

$$P_{w_{i+1}|w_i} = \frac{c(w_i w_{i+1}|rf_{all})}{c(w_i|rf_{all})}$$

where $w_i w_{i+1}$ is a bigram or words that are adjacent to each other in rf_{all}, $P_{w_{i+1}|w_i}$ is called the conditional probability of occurrence of the unigram w_{i+1} given w_i, $c(w_i w_{i+1}|rf_{all})$ is frequency of occurence of the bigram $w_i w_{i+1}$ together in rf_{all} and $c(w_i|rf_{all})$ is the frequency of occurence of w_i in rf_{all}. The user profile consists of

$$\{(w_1 w_2, P_{w_2|w_1}), (w_2 w_3, P_{w_3|w_2}), ..., (w_i w_{i+1}, P_{w_{i+1}|w_i}), ...(w_{n-1} w_n, P_{w_n|w_{n-1}})\}$$

A sample user profile is as shown in Figure 4.2. The user profile thus learned is used in the reranking phase which is described in the next section.

$\mathbf{w_1 w_2}$	$\mathbf{P(w_2|w_1)}$	$\mathbf{w_1 w_2}$	$\mathbf{P(w_2|w_1)}$
red box	0.5	pokemon firered	0.0344
ruby cheats	0.2	pokemon ruby	0.0344
codes pokemon	0.125	defense version	1
pokemon safari	0.0344	tips gameboy	0.5
par deoxes	1	codes walkthroughs	0.1875
cheats gamespot	0.0526	color light	0.333
game boy	0.875	mystery gift	1

Table 4.2: Sample bigram user profile

4.3.2 Reranking

Reranking phase in our approach consists of first retrieving documents from a search engine matching the query and then re scoring the documents using the user profile which is described in the following sub sections. The documents are then arranged in descending order of this score.

4.3.2.1 Based on unigrams

In the unigram model, a query Q is represented as word sequence $q_1, q_2, ..., q_n$ and is modeled independently for each query word as follows: $P(q_1, q_2, ..., q_n) = P(q_1)P(q_2)...P(q_n)$. In general unigram language model for IR, the score of each document is computed by calculating the probability of occurrence of each query word in the document.

$$P(Q|D) = \prod_{i=1}^{n} P(q_i|D)$$

where Q is the query, D is the document and $P(q_i|D)$ is the probability of occurence of the word q_i in the document D.

Let $\mathcal{D}$ be the set of all documents returned by the search engine for a query Q. The score of each document is computed by calculating the probability that a document generates a query using the user profile. This is computed by calculating

the probability of occurrence of each query word in the document tuned using the user profile. It is computed as follows:

$$P(Q|D,UP) = \prod_{i=1}^{n} \alpha P(q_i|UP) + (1-\alpha)P(q_i|D) \tag{4.3}$$

where UP represents user profile, $P(q_i|UP)$ is the probability of the word from the user profile of the user u. and $P(q_i|D)$ is the probability of the word in the document and α is a weighting parameter with value between 0 and 1.

4.3.2.2 Based on bigrams

In the bigram model a query Q represented as word sequence $q_1, q_2, ..., q_n$ is modeled where a word is modeled conditioned on the previous word as follows: $P(q_1, q_2, ..., q_n) = P(q_1)P(q_2|q_1)...P(q_n|q_{n-1})$. In general bigram language model for IR, score of each document is computed by calculating the probability of occurrence of each query word in the document given the previous word.

$$P(Q|D) = P(q_1) \prod_{i=2}^{n} P(q_i|q_{i-1})$$

where Q is the query, D is the document and $P(q_i|q_{i-1})$ is the probability of occurence of the word $q_{i-1} followed by q_i$ in the document D.

Let $\mathcal{D}$ be set of all the documents returned by the search engine for a query Q. The score of each document is computed by calculating the probability that a document generates a query using the user profile. This is computed by calculating the probability of occurrence of each query word in the document given the previous word tuned using the user profile. It is computed as follows:

$$P(Q|D,UP) = P(q_1) \prod_{i=2}^{n} \alpha P(q_i|q_{i-1},D) + (1-\alpha)P(q_i|q_{i-1},UP) \tag{4.4}$$

where UP represents user profile, $P(q_i|q_{i-1},UP)$ is an entry from the bigram user profile of the user u and $P(q_i|q_{i-1},D)$ is the conditional probability of the word in the document α is a weighting parameter with value between 0 and 1. The conditional

probability helps us to capture the relationship between a word and its previous word and measures probability of dependence of word on its previous word.

The documents are scored based on this probability $P(Q|D, UP)$ and documents are arranged based on descending order of this score.

4.4 NOISY CHANNEL BASED APPROACH

In order to address the problem of personalization one needs to clearly understand the actual process of search. Consider the case when a user has an information need that he would like to fulfill. He is the only entity in the process that knows the exact information he needs and also has a vague notion of the document that can fill his specific information. A query based search engine is at his disposal for identifying this particular document or set of documents from among a vast repository of them. He then formulates a query that he thinks is congruent to the document he imagines to fulfill his need and poses it to the search engine. The search engine now returns a list of results. Every user is different and has a different information need, perhaps overlapping sometimes. The way a user conceives an ideal document that fulfills his need also varies. It is our hypothesis that if one can learn the variations of each user in this direction, effective personalization can be done.

We effectively model the process of query formulation and better characterize how a user relates his query to the document that he intends to retrieve as discussed in the web search process above. A user profile learned from the relevance feedback that captures the query generation process is used as a guide to understand user's interests over time and personalize his web search results.

Interestingly, recently a new paradigm has been proposed for retrieval rooted from statistical language modeling that views the query generation process through a Noisy channel model [5] . It was assumed that the document and query are from different languages and the query generation process was viewed as a translation from the document language (which is more verbose) to the query language (which is more

compact and brief). The noisy channel model proposed by Berger and Lafferty [5] inherently captures the dependencies between the query and document words by learning a translation model between them. As we intend to achieve personalized search by personalizing the query formulation process, we also perceive the user profile learning through a Noisy Channel Model. In the model, when a user has an information need, he also has an ideal document in mind that fulfills his need. The user in a way tries to translate the notion of the ideal document into a query that is more compact but congruent to the document. He then poses this query to the search engine and retrieves the results. By observing this above process over time, we can capture how the user is generating a query from his ideal document. By learning this model of a user, we can predict which document best describes his information need for the query he poses. This is the motive of personalization.

This problem draws similarities to statistical machine translation where a sentence in one language is translated to another language while passing through a noisy channel. The channel probabilities are computed which contain information about how a word in the source language is getting translated into another a word in another language. A translation model is constructed from the same which contains a source word (word in source language), a target word(word in target language) and the probability of translation of the source word to the target word. The channel probabilities are computed using statistical techniques using Expectation-Maximization(EM) algorithm [6] from "parallel texts" consisting of sentences in source language and corresponding sentences in target language. When a new sentence in source language is given, it is translated into target language using the translation model.

In our approach, we use the past search history of the user. A clicked document is assumed to be relevant and an approximation of an ideal document. Given logs of past queries and corresponding "approximate" ideal documents, we can now estimate how an ideal document is being translated to an ideal query and thereby capture the relationship between query and document words. By considering the ideal documents and queries as parallel texts i.e., ideal documents as texts in one language and queries

as texts in another language, we compute a translation model using statistical machine translation algorithms. The translation model thus learned constitutes the user profile. In the Reranking phase, given a query, a bag of documents matching the given query are retrieved using a search engine (ex: Google). Top few documents from the retrieved documents are considered and a new score is computed. The new score measures how much the word translates into query word and its probability of occurrence in the document. The documents are then sorted in descending order of this score. The better the relationship between a query and document words, the higher the new score, the better the document and the better the document is supposed to be.

In the rest of the section, we describe the different steps in the approach in detail.

4.4.1 Learning user profile

In this approach, a user profile consists of a statistical translation model. A translation model is a probabilistic model consisting of the triples, the source word, the target word and the probability of translation. Our user profiles consists of the following triples, a document word, a query word and the probability of the document word generating the query word.

Consider a user u, let $\{ \{Q_i, D_i\},\ i = 1, 2, ..., N\}$ represent the past history of the user u. where Q_i is the query and D_i is the concatenation of all the relevant documents for the query Q_i and let $D_i = \{w_1, w_2, ..., w_n\}$ be the words in it. The user profile learned from the past history of user consists of the following triples of the form $(q, w_i, p(q|w_i))$ where q is a word in the query Q_i and w_i is a word in the document D_i.

Translation model is typically learned from parallel texts i.e., a set of translation pairs consisting of source and target language sentences. In learning the user profile, we first extract parallel texts from the past history of the user and then learn the translation model which is essentially the user profile. In the sub sections below, we describe the process in detail.

4.4.1.1 Extracting Parallel Texts

By viewing documents as samples of a verbose language and the queries as samples of a concise language, we can treat each document-query pair as a translation pair, i.e. a pair of texts written in the verbose language and the concise language respectively. The extracted parallel texts consists of pairs of the form $\{Q_i, D_{rel}\}$ where D_{rel} is the concatenation of contexts extracted from all relevant document for the query Q_i.

We believe that short snippets extracted in the context of the query would be better candidates for D_{rel} than using the whole document. This is because there can be a lot of noisy terms which need not be right in the context of the query. We believe a short snippet usually N (we considered 15) words to the left and right of the query words, similar to a short snippet displayed by search engines can better capture the context of the query. In deed, we experimented with different context sizes for D_{rel}. The first is using the whole document i.e., considering the query and concatenation of content all the relevant documents as a pair in the parallel texts extracted. This is called $D_{documents}$. The second is using just a short text snippet from the document in the context of query instead of the whole document which is called $D_{snippets}$. Experiments are performed by learning user profile using $D_{documents}$ and $D_{snippets}$. Details are described in the experiments section.

4.4.1.2 Learning Translation Model

A Translation model consists of the triples source word, target word and the probability of translation of source word to target word. According to standard statistical machine translation methods [6], it is learned from parallel texts consisting of source and target language sentences. by calculating relative co-occurence of query and document words using Expectation Maximization(EM) algorithm.

A translation model is learned by finding an optimal model M^* by maximizing the

q	w	P(q|w)	q	w	P(q|w)
green	red	0.0074	pokemon	cheat	0.0117
zone	pokemon	0.0343	sappire	gift	0.0008
ticket	capture	0.0987	deoxes	pokemon	0.0019
america	gba	0.0014	deoxes	sapphire	0.0096
walkthrough	ticket	0.0335	pokemon	boy	0.0278
leaf	codes	0.0300	game	color	0.0034

Table 4.3: Noisy Channel based user profile

probability of generating queries from documents

$$M^* = \arg\max_M \prod_{i=1}^{N} P(Q_i|D_i, M)$$

To find the optimal word translation probabilities $P(qw|dw, M*)$, EM algorithm is used. The details of the algorithm can be found in the literature for statistical translation models, such as [6].

Consider a user u, let $\{\{Q_i, D_i\}, i = 1, 2, ..., N\}$ be the parallel texts extracted as described in sub section 4.4.1.1 represent the past history of the user u. An optimal model is learned by training on the parallel texts using standard machine translation techniques. IBM Model1 [6] is a simplistic model which takes no account of the subtler aspects of language translation including the way word order tends to differ across languages. Similar to earlier work [5], we use IBM Model1 because we believe it is more suited for IR because the subtler aspects of language used for machine translation can be ignored for IR. GIZA++ [34], an open source tool which implements the IBM Models which we have used in our work for computing the translation probabilities. A sample user profile is as shown in Figure 4.3.

4.4.2 Reranking

Reranking phase in our approach consists of first retrieving documents from a search engine matching the query and then re scoring the documents using the user profile

which is described below. The documents are then arranged in descending order of this score.

Let $\mathcal{D}$ be set of all the documents returned by the search engine. The rank of each document D returned for a query Q for user u is computed by calculating a new score using his user profile. It is based on how a word in the ideal document gets transformed while getting translated into a query word while passing through the noisy channel. This is obtained from the translation model trained or the user profile of the user. It is computed as shown in Equation 6.1.

$$P(Q|D,u) = \prod_{q_i \in Q} \alpha P(q_i|GE) + (1-\alpha) \sum_{w \in D} P(q_i|w,u)P(w|D) \tag{4.5}$$

where $P(q_i|GE)$ is the smoothed or general probability obtained from a large general corpus. $P(q_i|w,u)$ is an entry in the translation model of the user. It represents the probability of generation of the query word q_i for a word w in the document. $P(w|D)$ is the probability of the word w in the document and α is a weighting parameter which lies between 0 and 1. In this way we capture the rich context in user profile by modeling the relationship between query words and document words in a translation model using statistical machine translation methods.

In the chapter, we described two approaches of how richer context can be captured and modeled using language modeling. The first one captures relationship between adjacent words and the second one captures the relationship between query and document words and models using noisy channel method.

CHAPTER 5

Personalized Search using User Relevance Feedback: Machine learning based approach

5.1 INTRODUCTION

Machine learning algorithms have received a wide attention recently to learn functions that can perform desired operations when trained on required amount of data. Personalized search is one such potential area which the use of machine learning can be made. Given the previous history of the user, can we learn a model representing the user ? This makes user profile learning a potential problem for use of machine learning.

However, most work on machine learning especially in information retrieval simplifies the task to a binary classification problem with the two classes relevant and non-relevant. Such a simplification has several drawbacks. Firstly, because, often when a user gives relevance feedback, he only gives information of documents which he thinks are relevant to him. The user typically does not give information about the documents not relevant to him. For example, assuming all the documents not identified as relevant to be irrelevant leads to problems. Due to a strong majority of non-relevant documents, a learner will typically achieve the maximum predictive classification accuracy, if it always responds non-relevant, independent of where the relevant documents are ranked. Consider implicit relevant feedback - clickthrough data, it has been shown that the user always clicks documents which appear to be relevant in the top few results. In this case, the user has clicked those results because they might have appeared to be more relevant than the others seen by him in the top few. Hence, clickthrough data only contain partial information and it would be an over simplification to assume

a binary classification.

Joachims et al[22] has proposed a new machine learning algorithm Ranking SVM, a variation of the Support Vector Machine learning algorithm which can learn from the partial feedback data present in the clickthrough data of users.

In our approach, we make use of Ranking SVM for learning user profile from the past search history of the user. We experimented with several features and feature weights for training SVM.

In the next subsections, we give brief description about Ranking SVM followed by our approach for personalized search based on the same.

5.2 BACKGROUND

Given an independently and identically distributed training sample S of size n containing queries q with their target rankings r^*

$$(q_1, r_1^*), (q_2, r_2^*), ..., (q_n, r_n^*)$$

The learner $\mathcal{L}$ will select a ranking function f from a family of ranking functions F that maximizes the empirical τ

$$\tau_S(f) = \frac{1}{n} \sum_{i=1}^{n} \tau(r_{f(q_i)}, r_i^*) \tag{5.1}$$

on the training sample. Note that this setup is analogous classification by minimizing training error, just that the target is not a class label, but a binary ordering relation.

Is it possible to design an algorithm and a family of ranking functions F so that (a) finding the function $f \in F$ maximizing Equation 5.1 is efficient, and (b) that this function generalizes well beyond the training data. Consider the class of linear ranking functions analogous to

$$(d_i, d_j) \in f_{\vec{w}}(q) \Leftrightarrow \vec{w}\Phi(q, d_i) > \Phi(q, d_j) \tag{5.2}$$

where w is a weight vector that is adjusted by learning. $\Phi(q, d)$ is a mapping onto features that describe the match between query q and document d. Such features are, for example, the number of words that query and document share etc.

For the class of linear ranking functions Equation(5.2), this is equivalent to finding the weight vector so that the maximum number of the following inequalities is fulfilled.

$$\begin{aligned} \forall(d_i, d_j) \in r_1^* \quad &: \quad \overrightarrow{w}\Phi(q_1, d_i) > \overrightarrow{w}\Phi(q_1, d_j) \\ &\ldots \\ \forall(d_i, d_j) \in r_n^* \quad &: \quad \overrightarrow{w}\Phi(q_n, d_i) > \overrightarrow{w}\Phi(q_n, d_j) \end{aligned}$$

Unfortunately, a direct generalization of the result shows that this problem is NP-hard. However, just like in classification SVM [10], it is possible to approximate the solution by introducing (non-negative) slack variables $\xi_{i,j,k}$ and minimizing the upper bound $\sum \xi_{i,j,k}$. Adding SVM regularization for margin maximization to the objective leads to the following optimization problem, which is similar to the ordinal regression approach in [16]

OPTIMIZATION PROBLEM 1. (RANKING SVM)

$$\begin{aligned} \textit{minimize} \quad &: \quad V(\overrightarrow{w}, \overrightarrow{\xi}) = \frac{1}{2}\overrightarrow{w} \cdot \overrightarrow{w} + C \sum \xi_{i,j,k} \\ \textit{subject to} \quad &: \quad \forall(d_i, d_j) \in r_1^* : \overrightarrow{w}\Phi(q_1, d_i) \geq \overrightarrow{w}\Phi(q_1, d_j) + 1 - \xi_{i,j,1} \\ &\quad \ldots \\ &\quad \forall(d_i, d_j) \in r_n^* : \overrightarrow{w}\Phi(q_n, d_i) > \overrightarrow{w}\Phi(q_n, d_j) + 1 - \xi_{i,j,n} \\ &\quad \forall i \forall j \forall k : \xi_{i,j,k} \geq 0 \end{aligned}$$

C is a parameter that allows trading-off margin size against training error. Geometrically, the margin δ is the distance between the closest two projections within all target rankings. Optimization Problem 1 is convex and has no local optima. By rearranging the constraints in

$$\forall(d_i, d_j) \in r_1^* : \overrightarrow{w}\Phi(q_1, d_i) \geq \overrightarrow{w}\Phi(q_1, d_j) + 1 - \xi_{i,j,1}$$

we get

$$\overrightarrow{w}(\Phi(q_1, d_i) - \Phi(q_1, d_j)) \geq 1 - \xi_{i,j,1}$$

it becomes apparent that the optimization problem is equivalent to that of a classification SVM on pairwise difference vectors $\Phi(q_1, d_i) - \Phi(q_1, d_j)$. Due to this similarity, it can be solved using decomposition algorithms similar to those used for SVM classification. In the following, an adaptation of the SVM^{light}algorithm [14] is used for training [1]

It can be shown that the learned retrieval function $f_{\overrightarrow{w}}$ can always be represented as a linear combination of the feature vectors

$$\begin{aligned}(d_i, d_j) &\in f^*_{\overrightarrow{(}w)}(q) \\ &\Leftrightarrow \overrightarrow{(}w)^*\Phi(q, d_i) > \overrightarrow{(}w)^*\Phi(q, d_j) \\ &\Leftrightarrow \sum \alpha^*_{k,l}\Phi(q_k, d_l)\Phi(q, d_i) > \sum \alpha^*_{k,l}\Phi(q_k, d_l)\Phi(q, d_j)\end{aligned}$$

where $\alpha^*_{k,l}$ and $\Phi(q, d_i)$ are chosen such that they are to be greater than zero. This makes it possible to use Kernels and extend the Ranking SVM algorithm to non-linear retrieval functions. The $\alpha^*_{k,l}$ can be derived from the values of the dual variables at the solution. Most commonly, $f_{\overrightarrow{w}}$will be used for ranking the set of documents according to a new query q. In this case it is sufficient to sort the documents by their value of

$$rsv(q, d_i) = \overrightarrow{w}\Phi(q, d_i) = \sum \alpha^*_{k,l}\Phi(q_k, d_l)\Phi(q_k, d_j) \tag{5.3}$$

If Kernels are not used, this property makes the application of the learned retrieval function very efficient. Fast algorithms exists for computing rankings based on linear functions by means of inverted indices

If clickthrough logs are the source of training data, the full target ranking r^* for a query q is not observable. A subset of r^* can be available in the clickthrough data. It has been shown that ranking SVM can learn in this case of partial data. The details can be found in [22].

[1] Available at http://svmlight.joachims.org

5.3 PERSONALIZATION USING RANKING SVM

We propose an approach for personalized search in this framework. We learn the user model by learning a model using Ranking SVM from the clickthrough history of the user. The input to the Ranking SVM is pairs of preference constraints inferred from the clickthrough history. Preference constraints are the constraints specified for training the SVM model. For each query, with at least 1 clicked document, a few preference constraints are constructed. It is believed that clicked documents are more preferred than unclicked documents because the users only might have clicked documents because they thought it might be relevant. However, an unclicked document were not clicked because the user did not consider relevant in comparison to a clicked documents. Preference constraints are constructed for each pair of clicked and unclicked document for a given query specifying that a clicked document is more preferred than an unclicked document. We used SVM^{light}[2] for training.

It is believed that the learned model depends on the choice of features and feature weights. We considered all the words in snippets as features (short snippet of the documents similar to what is display typically by search engines consisting of a few words around the query words in the document). We experimented with different features and feature weights for training the SVM model. The first is the bag of words where each feature is either present or absent given a value 1 to features present and 0 to the ones not present. But, sometimes this cannot capture the rich information contained in the repetition of words. We experimented by considering the frequency of the times each feature occurred. We also experimented with normalized frequency of the words. After training, we obtain a model consisting of the α values and the support vectors. We compute the weight vector from the same (We only consider linear kernel).

The approach has two phases. The first phase is learning user profile. The second

[2] A free machine learning software for research purposes used widely in the machine learning literature for training SVM

phase is called Reranking. They are described in detail in rest of the section.

5.3.1 Learning user profile

Learning user profile involves training an SVM model. It involves the following steps. 1) Extracting features 2) Computing Feature weights 3) Training SVM. We describe each of the steps in detail in the rest of the sub section.

5.3.1.1 Extracting Features

We experimented with two types of features - unigram and bigram. Unigrams correspond to a single word in a text. Bigram corresponds to two adjacent words in a text. For example, given a text like "this is a piece of text" The unigrams are {this, is, a, piece, of , text} and bigrams are {this is, is a, a piece, piece of, of text}. Removing stop words (i.e., words like is, a,of etc which occur very often in a text and most of the times do not carry much information) we get the unigrams {this,piece,text} and bigrams {this piece, piece text}.

We first collect the feedback provided by the user for all the queries and compute a concatenation of the feedback. Let H_u represent the search history of the user u. H consists of $\{(q_1, d_1), (q_1, d_2), (q_2, d_3), ...(q_i, d_j), ...(q_n, d_m)\}$ where q_1 is the past query posed by the user and d_1 is the text of the relevance feedback given by the user. We compute a concatenation of all the past relevance feedback from the user called rf_{all}.

We now extract from rf_{all} all the unigrams or bigrams when unigrams are used as features or bigrams as features respectively.

5.3.1.2 Computing Feature Weights

Consider unigrams as features. We experimented with three types of feature weighting strategies namely boolean, term frequency and normalized term frequency.

For *Boolean* weighting, for each clicked document d_i for each unique unigram we assign a weight of 1 if it occurs in d_i and 0 otherwise. For *Term Frequency* weighting,

for each clicked document d_i for each unique unigram, we assign a weight of the number of times the word occurs in d_i. Let w be a word in d_i and tf_w be the number of times it occurs in d_i, then weight of w is tf_w. For *Normalized Term Frequency* weighting, for each clicked document d_i for each unique unigram, the weight assigned is as follows. Let w be a word in d_i and tf_w be the number of times it occurs in d_i, then weight of w is $\frac{tf_{w,D}}{|D|\ |Q|}$. Similarly we do it for bigrams.

5.3.1.3 Training SVM

We used SVM^{light} for training. It requires each feature to be represented with a unique numerical identifier. Therefore, we assign a unique numerical identifier for each unique feature. Each document now is represented as vector of features extracted from it and feature weights are calculated as described above. Sample features and weights are shown in Table 5.1. Each line in the training data is as follows. "70 qid:0 1308:1 2618:1 2657:1 2831:1 5783:1". The number 70 is called the target value which typically specifies the class to which the training example belongs to. In the ranking SVM, the target value is used to generate pairwise preference constraints. A preference constraint is included for all pairs of examples in the training file, for which the target value differs. The special feature "qid" can be used to restrict the generation of constraints. For example, given the training file

$$\begin{aligned} d_1 &\rightarrow 3\ qid:11:0.532:0.12 \\ d_2 &\rightarrow 2\ qid:11:0.132:0.1 \\ d_3 &\rightarrow 7\ qid:21:0.872:0.12 \end{aligned}$$

a preference constraint is included only for the first and the second example (i.e., the first should be ranked higher than the second), but not with the third example, since

Boolean			
	70	qid:0	1308:1 2618:1 2657:1 2831:1 5783:1
	69	qid:0	1308:1 2618:1 2657:1 2831:1 5783:1
	68	qid:0	1308:1 2618:1 2657:1 2831:1 5783:1
	0	qid:0	2618:1 2657:1 2831:1 5783:1
	0	qid:0	1308:1 2618:1 2657:1 2831:1 5783:1
Term Freq			
	70	qid:0	1308:4 2618:5 2657:5 2831:2 5783:1
	69	qid:0	1308:1 2618:2 2657:3 2831:1 5783:4
	68	qid:0	1308:2 2618:2 2657:2 2831:1 5783:2
	0	qid:0	2618:4 2657:4 2831:1 5783:2
	0	qid:0	1308:2 2618:2 2657:2 2831:4 5783:1
Norm Term Freq			
	70	qid:0	1308:0.053 2618:0.066 2657:0.066 2831:0.0266 5783:0.013
	69	qid:0	1308:0.009 2618:0.018 2657:0.027 2831:0.009 5783:0.036
	68	qid:0	1308:0.028 2618:0.028 2657:0.028 2831:0.014 5783:0.028
	0	qid:0	2618:0.042 2657:0.042 2831:0.010 5783:0.021
	0	qid:0	1308:0.016 2618:0.016 2657:0.016 2831:0.033 5783:0.008

Table 5.1: Sample: Different Features weights for unigrams

it has a different "qid". These preference constraints can be written as

$$\forall(d_1, d_2) \in r_1^* \quad : \quad \overrightarrow{w}\Phi(q_1, d_1) > \overrightarrow{w}\Phi(q_1, d_2)$$

These preference constraints are then used to learn an SVM model. A sample weight vector learned is as shown in Table 5.2. This weight vector corresponds to the user profile of the user. Reranking is done using the user profile of the user and is described in the next sub section.

Boolean	
	251:0.059 286:0.019 658:-2.220 1308:0.299 2488:-0.5 2618:-6.66 2657:0.299 2831:-6.661 2907:0.08 3065:-0.5 3095:-0.6 3147:0.94 3174:0.1 4170:0.099 5219:-0.5 5319:0 5429:-0.3 5783:-1.054 6411:-8.32 e-17
Term Freq	
	251:0.310 7 286:-4.440 658:-0.199 1308:0.515 2488:-0.394 2618:-0.118 2657:0.134 2831:-0.460 2907:0.089 3065:-0.426 3095:-0.356 3147:0.379 3174:0.287 5219:0.105 5319:5.55e-17 5429:-0.3 5783:-1.637e-15 6411:-0.013
Norm Term Freq	
	251:0.009 286:0.001 658:-0.003 1308:0.025 2488:-0.044 2618:0.018 2657:0.010 2831:-0.004 2907:0.010 3065:-0.006 3095:0.012 3147:0.021 3174:0.031 4170:0.003 5219:-0.036 5319:0.0002 5429:-0.001 5783:0.019 6411:-0.003

Table 5.2: Sample: User Profile for different weighting strategies for unigrams

5.3.2 Reranking

Reranking phase in our approach consists of first retrieving documents from a search engine matching the query and then re scoring the documents using the user profile which is described below. The documents are then arranged in descending order of this score.

Let $\mathcal{D}$ be set of all the documents returned by the search engine. The rank of each document d_i returned for a query q for user u is computed using his user profile as shown in Equation 5.4. $\Phi(q, d_i)$ measures the similarity between the query q and the document d_i. Features are extracted and corresponding weights are computed as described in sub section (5.3.1). Let w_u be the user profile of the user u. Then, score

of a document d_i using the user profile of user u is computed as

$$rsv(q, d_i, u) = \overrightarrow{w_u}\Phi(q, d_i) = \sum \alpha^*_{k,l,u}\Phi(q_k, d_l)\Phi(q_k, d_j) \tag{5.4}$$

This essentially computes the product of weight of a term in query, document and the user profile. The higher this score of a document, the better there is match between the learned weight vector (i.e., the user profile) and the document, the better is the document. The documents are then sorted in descending order of this score.

In this chapter, we described an approach for learning user profile using Ranking SVM, a machine learning algorithm. Partial feedback available from the past click-throughs of the user is used to learn an SVM model which constitutes the user profile. It consists of a weight vector with features and their corresponding weights. This weight vector is then used to rerank the documents. This approach shows a promising and interesting direction for effective user profile learning using machine learning methods.

CHAPTER 6

Personalized Search without User Relevance Feedback: Simple Language Modeling based Method

6.1 INTRODUCTION

Relevance feedback has received wide attention recently, as a means to capture the search context to improve search accuracy. Feedback is explicitly collected from user by asking him, to mark the documents relevant to them which is called explicit relevance feedback [39]. Unfortunately, in real world applications, users are usually reluctant to make the extra effort to provide relevant examples for feedback [23]. Implicit relevance feedback methods have been proposed that try to infer user's relevance feedback by observing the user's interactions with the retrieval systems like, dwelling time on a page, amount of scrolling on a page, clicks etc [9]. These approaches do not require any extra user effort [23]. One method of acquiring large volumes of such implicit data is by aggregating the "clickthrough data" or "query log" of a real world search engine which has been used to improve ranking of search engines to improve retrieval accuracy [22], [56] etc.

While implicit and explicit feedback have seen quite interesting results, both need feedback from the user either explicitly and implicitly. Due to this and many other reasons, there are not many publicly available data sets of the kind. Another major problem with either explicit or implicit feedback data when operating on the web search domain is as follows. Web data is extremely dynamic - millions of pages are created and deleted every day, and the content of existing pages changes quite frequently. Consequently, snapshots of the Web taken several weeks apart may give rise to very

different indices. Thus, the feedback data may rapidly become stale, with new pages replacing old ones as the most appropriate online resources for queries. In other cases, even though the "old" expected results remain good resources, search engines will not retrieve them in response to queries . Instead, the engines will return near-duplicate pages, that have equivalent content but different URLs.

To the best of our knowledge, previous work on personalized search has assumed and focused on modeling the user from the relevance feedback provided by him either explicitly or implicitly. Although interesting results have been seen, it is not without problems. We believe that the queries posed by the user convey the user context information though in a concise way. To our knowledge, this resource has not been well exploited so far. We show how the past queries posed by the user can be effectively used to personalized search [41].

6.2 APPROACH

We first learn a user profile from the past queries of the user and then re rank the results using the user profile.

6.2.1 Learning user profile

A user profile is a probabilistic distribution over words. It consists of words(unigrams) and their respective probabilities. We first collect all the queries posed by the user and form a concatenation of all the queries. Then, we count the number of times each word occurs in the same. Let Q_{past} be the set of all the past queries posed by the user and q_{concat} be the concatenation of all the queries. The user profile consists of $\{(w_1, P_{w_1}), (w_2, P_{w_2}), ..., (w_n, P_{w_n})\}$ where

$$P_{w_i} = \frac{c(w_i, q_{concat})}{\sum_{w_i \in q_{concat}} c(w_i, q_{concat})}$$

where $c(w_i, q_{concat})$ is the frequency of the word w_i in q_{concat}. A sample user profile is shown in Figure 6.1.

w	$P(w)$	w	$P(w)$
mystery	0.0219	sapphire	0.0106
silver	0.0105	green	0.05
cave	0.0099	ruby	0.0098
pokemon	0.0973	gift	0.0232
zone	0.0076	safari	0.0075
crystal	0.0074	ticket	0.0074
leaf	0.0357	cheats	0.0405
games	0.0064	deoxes	0.0064

Table 6.1: Sample query based user profile

6.2.2 Reranking

Reranking phase in our approach consists of first retrieving documents from a search engine matching the query and then re scoring the documents using the user profile which is described below. The documents are then arranged in descending order of this score.

Let $\mathcal{D}$ be set of all the documents returned by the search engine. The rank of each document D returned for a query Q for user u is computing using his user profile as shown in

$$P(Q|D,u) = \prod_{q_i \in Q} \alpha P(q_i|UP) + (1-\alpha)P(q_i|D) \tag{6.1}$$

where $P(q_i|UP)$ is the probability of the word from the user profile and $P(q_i|D)$ is the probability of the word in the document.

In this approach, we investigate to see what information do the queries carry and if a log of just the past queries can serve as a source to learn a user model. The results from this research can be extremely useful and provides an interesting research direction for personalized search because obtaining just the queries is comparatively easy than relevance feedback. The queries can then be used directly for personalization or in combination with available relevance feedback.

CHAPTER 7

Experimental Evaluation

In this section we describe the experiments performed on our proposed approaches. We first describe the problems in evaluating personalized web search algorithms in general followed by the description of the data used in our experiments and the experimental set up and the metrics used. We then describe the experiments and results of each of our approaches in detail.

7.1 PROBLEMS

To the best of our knowledge, there is no standard publicly available test collections for evaluation of personalized search algorithms. The creation of sharable test collections facilitates discovery and allows for more rapid progress since building a good test collection is such a difficult, laborious, and time-consuming task. Standard test collections also allow for multiple modes of inquiry including those that involve the comparison of various techniques, examination of alternative hypotheses and replication of previous findings. Indeed it has been identified that there is a great need for such standardization [2].

7.2 DATA DESCRIPTION

The data used in this thesis is implicit feedback in the form of clickthrough data collected by a popular search engine recently released for research purposes. We first describe the clickthrough data followed by the method of extracting the data set used in our experiments.

7.2.1 Clickthrough data

This data consists of about 20M web queries collected from approximately 650k users over three months. The data is sorted by anonymous user ID and sequentially arranged.

The data set includes AnonID, Query, QueryTime, ItemRank, ClickURL.

- AnonID - an anonymous user ID number.
- Query - the query issued by the user, case shifted with most punctuation removed.
- QueryTime - the time at which the query was submitted for search.
- ItemRank - if the user clicked on a search result, the rank of the item on which they clicked is listed.
- ClickURL - if the user clicked on a search result, the domain portion of the URL in the clicked result is listed.

Each line in the data represents one of two types of events: 1. A query that was NOT followed by the user clicking on a result item. 2. A click through on an item in the result list returned from a query. In the first case (query only) there is data in only the first three columns/fields – namely AnonID, Query, and QueryTime (see above). In the second case (click through), there is data in all five columns. For click through events, the query that preceded the click through is included. Note that if a user clicked on more than one result in the list returned from a single query, there will be TWO lines in the data to represent the two events. Also note that if the user requested the next "page" or results for some query, this appears as a subsequent identical query with a later time stamp. Sample data is shown in Table 7.1.

Some basic statistics about the data are as follows.

Dates: 01 March, 2006 - 31 May, 2006

Normalized queries:

36,389,567 lines of data

21,011,340 instances of new queries (w/ or w/o click-through)

ID	Query	Query Time	Pos	url
2722	charles drew	2006-03-01 18:00:07	10	http://www.cdhcmedical.com
2722	tricare	2006-03-16 19:07:38	2	http://www.tricareonline.com
142	rentdirect.com	2006-03-01 07:17:12		
142	westchester.gov	2006-03-20 03:55:57	1	http://www.westchestergov.com
142	vera.org	2006-04-08 08:38:427	1	http://www.vera.org
142	broadway.vera.org	2006-04-08 08:39:30		

Table 7.1: A snapshot of the Query log data

7,887,022 requests for "next page" of results

19,442,629 user click-through events

16,946,938 queries w/o user click-through

10,154,742 unique (normalized) queries

657,426 unique user ID's

7.2.2 Test Data set Preparation

The clickthrough data described above consists of queries and clickthroughs of millions of users. We prune this data to suit to our research purposes. In this section, we describe how we extract our data set. Firstly, we assume that each anonymous id given in the query log corresponds to each user. It is a reasonable assumption to make and has been used in some earlier works. Since we are working on modeling long term modeling of the user, we ensure that the data we pick exhibits long term behaviour. To

satisfy this, we ensure that the data follows the following conditions. 1) A query has at least one click 2) Queries posed by users exhibit long term behaviour i.e., queries are posed for more than 3 months by a user and exhibit similar interests. We measure the same by seeing the repetition of query words.

The data finally extracted consists of 17 users and queries and corresponding clicked URLs by users. This data is divided into two parts using the time stamp of query. The entire data spans over 3 months. The queries posed in the first 2 months are used in learning user profile and queries, their corresponding clickthroughs of a user form the training set of the user. The queries posed in the third month are used for testing and queries and corresponding clickthroughs of the user form the testing set of the user.

7.3 EVALUATION METRICS

The evaluation metrics chosen in this work are Mean Reciprocal Rank (MRR) and Precision @N (P@N). MRR was defined in [61]. Given a set of queries Q, the set of documents D, the subset A of which are considered as the "relevant" documents for $q \in Q$ and the set of the first documents $R_{D,Q,n}$ of D returned for every query q so that $R_{D,q,n} = \{D_{q,1}, D_{q,1}, ..., D_{q,n}\}$, the MRR is defined by:

$$mrr(Q, D, n) = \frac{\sum_{q \in Q} rr(q, R_{D,Q,n})}{|Q|}$$

where $rr(q, R_{Q,D,n})$ is the *Reciprocal Rank*(RR) that depends on the position of the first returned snippet from the result list which contains a relevant document. Or 0 if there is no relevant document is found in the first n documents. The function is defined by

$$rr(q, R_{D,Q,n}) = \begin{cases} \frac{1}{i} & \exists i | i = \min_{1 \leq j \leq n} j | D_{q,j} \in A_{D,q} \\ 0 & \text{otherwise} \end{cases}$$

Precision is a widely used metric in information retrieval and search search systems. It measures the proportion of the retrieved documents that are relevant. It is defined

as follows

$$\text{Precision} = \frac{|\text{relevant documents} \cap \text{retrieved documents}|}{\text{retrieved documents}}$$

Precision takes all retrieved documents into account. It can also be evaluated at a given cut-off rank, considering only the topmost results returned by the system. This measure is called precision at n or P@n.

In web search systems, there are typically very large number of retrieved results. In order to compute Precision, we need to know whether each document retrieved is relevant or not. Practically, this is very difficult to calculate. Moreover, related work on web search systems has shown that users typically view only top few results. In such a case, it is highly important to show the most relevant results among the top few. In need, the problem of personalized search is to devise systems to show most relevant results to the user in the top few which could possibly reduce burden on a user to scroll through a long list of results. For these reasons, precision @n or P@n has been the widely used metric for evaluation of personalized search algorithms.

$$\text{Precision at n} = \frac{|\text{relevant documents} \cap \text{top n retrieved documents}|}{n}$$

In this thesis, we mainly used MRR for evaluating our approaches.

7.4 EXPERIMENTAL SETUP

The experimental set up is as shown in Figure 7.1. The data for each user consists of queries and their corresponding clicked URLs. The training data is used for learning user profile and testing data is used for evaluating the approaches. For training, first, the query is posed to a web search engine and the results returned by the search engine are collected. The corresponding snippets from the search engine are collected and used for learning user profile according to the chosen method. For testing (See Figure 7.1), first, the query is posed to a web search engine and the top M results returned by the search engine are collected. These results are then passed to the reranker. The reranker then re ranks the results and returns the reranked results. The top N results

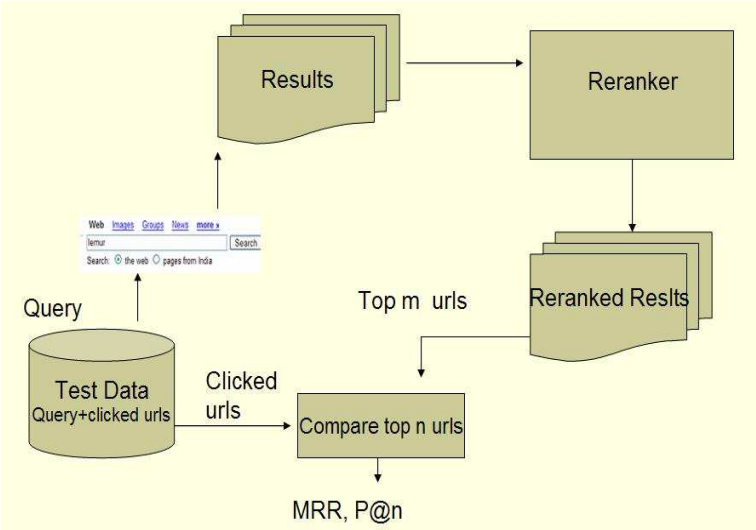

Fig. 7.1: Experimental Set up

from the reranked results are compared with the clicked URLs and the performance of the reranking algorithms is determined.

While comparing the top N results, we compare the domain names of the URLs because, the clickthrough data only contained the domain name. After comparing the top N URLs, we determine MRR and P@N which are the chosen metrics of our work and are described in detail in Section 7.3.

7.5 BASELINE

There are no standard baselines available in the literature as mentioned earlier. This poses difficulties in comparison with previous work. Many approaches use Rocchio algorithm, which uses relevance feedback to improve retrieval performance for comparison. Similarly, we used a variation of the Rocchio[39] algorithm as the baseline.

Standard Rocchio algorithm uses user relevance feedback and computes word frequencies from it. It incrementally modifies the query by adding terms from the relevance feedback. This has shown improvement in retrieval performance. Standard rocchio algorithm has been slightly modified for comparison for work in personalized

search. User profile is learnt from the user feedback and it involves computing word frequencies from it. Then, the user profile is used to re rank the results. Our implementation of the modified rocchio algorithm is similar.

We first collect the feedback provided by the user for all the queries and compute a concatenation of the feedback. Let H_u represent the search history of the user u. H consists of $\{(q_1, rf_1), (q_2, rf_2), (q_3, rf_3), ...(q_n, rf_n)\}$ where q_1 is the past query posed by the user and rf_1 is the text of the relevance feedback given by the user. We compute a concatenation of all the past relevance feedback from the user called rf_{all}.

The user profile consists of words and their respective frequencies. We extract all the words from it and count the number of times each word occurs from rf_{all}. The user profile consists of

$$\{(w_1, tf_{w_1}), (w_2, tf_{w_2}), ..., (w_i, tf_{w_i}), ...(w_n, tf_{w_n})\}$$

where w_i is a word in rf_{all} and tf_{w_i} is called the frequency of occurrence of the word w_i.

Let $\mathcal{D}$ be set of all the documents returned by the search engine for a query Q. The score of D using the above user profile(called RUP) of user u is computed as follows.

$$Sim(Q, D) = \sum_{w} \left(\frac{tf_{w,Q}}{|Q|} + \frac{tf_{w,RUP}}{|RUP|} \right) . \frac{tf_{w,D}}{|D|} \tag{7.1}$$

where $tf_{w_i,Q}$ = number of times the word w_i occurs in Q,
$tf_{w_i,RUP}$ = number of times the word w_i occurs in Rocchio user profile(RUP)
and$tf_{w_i,D}$ = number of times the word w_i occurs in D. D is the length of the document i.e., number of words in the document and Q is the length of the query i.e., number of words in the query. Documents are sorted based on this score and arranged in descending order of this score.

7.6 RESULTS

In this section, we describe our experimental results. We performed various experiments using each of our approaches and they are described in detail. We compared

Method	MRR	Improvement
Baseline	0.305	
unigrams	0.332	8.85
bigrams	0.338	11.18

Table 7.2: Evaluation: Simple N-gram based methods

the effectiveness of the user profile learned using simple N-gram based methods with a rocchio as the baseline.

7.6.1 Statistical language modeling based Approaches

In the rest of this sub section, we describe in detail the experiments performed using the language modeling based methods.

7.6.1.1 Simple N-gram based Methods

The unigram based user profile outperformed the baseline with 8.85% improvement and bigram based profile performed the best among the three with 11.18% improvement over the baselie. Bigrams are two consecutive words in a text and it is believed that they typically carry more information than just simple unigrams. Also, the chance of same bigram re-occurring is less and the occurrence of the same makes it a good evidence. Therefore, bigrams carry more context information than unigrams and hence bigram based user profile performed the best among the three.

7.6.1.2 Noisy Channel Model

In this approach, we performed three experiments. The first experiment is comparison with baseline. In The second experiment, we experiment different ways of extracting parallel texts and if there is any effect of this on the performance. We present a comparison of the performance of different ways tried out. The third experiment is comparison of different training schemes: With different training models and differ-

Method	MRR	Improvement
Baseline	0.305	
Noisy Channel	0.339	11.51

Table 7.3: Noisy Channel Results: Comparison with baseline

ent different context sizes. We investiage to see how the training scheme effects the performance and present a comparison between them (see Table 7.5).

Expt 1: Comparison with baseline

In this experiment, we compared our approach of personalized web search using noisy channel model with the baseline. Results are shown in Table 7.3. The approach showed an improvement of about 11.51% over baseline. Our assumption that capturing the relationship between query and document words help in modeling the context better has been well verified with these results.

Expt 2: Extracting Parallel Texts: Comparison of methods

In this experiment, we compared four methods of extracting parallel texts. Each method and the results and described in detail in this section.

In the first method called "NS1", parallel texts consists of each query and its corresponding relevant document. The extracted parallel texts consists of pairs $\{(Q_i, D_{1,i})\}$ where $D_{1,i}$ is the context extracted from the first relevant document for the query Q_i. We believe that a short snippets extracted in the context of the query would be better candidates for $D_{1,i}$ than using the whole document. This is because there can be a lot of noisy terms which need not right be in the context of the query. We believe a short snippet usually N (we considered 15) words to the left and right of the query words, similar to a short snippet displayed by search engines can capture the context better. We use short snippets which are typically given by search engines for the purpose. The extracted parallel texts consists of pairs of the form $\{(Q_i, D_{1,i}), ...(Q_i, D_{n,i}), ..., (Q_j, D_{m,j})\}$. Each pair $(Q_i, D_{1,i})$ represents a query and snippet from its corresponding relevant document. This method is called "NS1".

The parallel texts extracted in "NS1" are typically very small. Its size is equal

to the number of relevant documents for all the queries. Also, some times, there could be queries in similar context which might not contain exact query words but certain words that occur to the right or left of them in the snippet. Including such words in the training process and capturing the relationship between those words and the document words, can help us obtain a richer and enhanced model. Hence, we also extract such neighbouring words and assume them to be pseudo queries posed by the user. We call such queries pseudo or synthetic queries. Parallel pairs are extracted from such queries by using the snippet from which it is extracted as the parallel side. We extracted synthetic queries by extracting non overlapping trigrams from snippet and/or title of each relevant document. In methods NS2, NS3 and NS4 we extract synthetic queries. In NS2, synthetic queries are extracted from snippets of the relevant documents. In NS3, synthetic queries are extracted from snippets and title of the relevant documents. In NS4, synthetic queries are extracted from snippets of the relevant documents and title of the document is used a pseudo snippet parallel to query.

Consider NS2: We extracted trigrams from each of $D_{j,i}$. Trigrams are consecutive three words occurring in text. Consider the text $\{I\ went\ to\ a\ shop\}$. The trigrams are $\{I\ went\ to\}$, $\{went\ to\ a\}$ and $\{to\ a\ shop\}$[1]. For example, consider the pair of query and the document $(Q_1, D_{1,1})$ then we add to pairs to the set of parallel texts $(Q_{syn,1}, D_{1,1})$. We do so for all queries. Then, the extracted parallel texts consists of pairs of the form $\{(Q_1, D_{1,1}), (Q_{syn,1}, D_{1,1}), ..., (Q_j, D_{m,j}), (Q_{syn,m}, D_{m,j})\}$. This method is called "NS2".

In the third method, to NS2, we also add the pair consisting of document title of the respective document and the document. For example, consider the pair of query and the document $(Q_1, D_{1,1})$ then we add to pairs to the set of parallel texts $(Q_{syn,1}, D_{1,1})$ and $(Q_{title,1}, D_{1,1})$. Then, the extracted parallel texts consists of pairs of the form $\{(Q_1, D_{1,1}), (Q_{syn,1}, D_{1,1}), (Q_{title,1}, D_{1,1}), ...(Q_j, D_{m,j}), (Q_{syn,m}, D_{m,j}), (Q_{title,m}, D_{m,j})\}$. This method is called "NS3".

[1]Some times $\{a\ shop\ EOS\ \}$ where EOS means end of sentence is also considered but we ignore it in our work.

Method	**Parallel Texts**
NS1	$\{(Q_i, D_{1,i}), ...(Q_i, D_{n,i}), ..., (Q_j, D_{m,j})\}$
NS2	$\{(Q_1, D_{1,1}), (Q_{syn,1}, D_{1,1}),$ $..., (Q_j, D_{m,j}), (Q_{syn,m}, D_{m,j})\}$
NS3	$\{(Q_1, D_{1,1}), (Q_{syn,1}, D_{1,1}), (Q_{title,1}, D_{1,1}),$ $...(Q_j, D_{m,j}), (Q_{syn,m}, D_{m,j}), (Q_{title,m}, D_{m,j})\}$
NS4	$\{(Q_1, D_{1,1}), (Q_{syn,1}, D_{1,1}), (Q_1, Q_{title,1}),$ $...(Q_j, D_{m,j}), (Q_{syn,m}, D_{m,j}), (Q_j, Q_{title,m})\}$

Table 7.4: Different methods of parallel texts extraction

In the fourth method, to NS2, we also add the pair consisting of query and document title of the respective document. For example, consider a pair of query and document $(Q_1, D_{1,1})$, then we add to pairs $(Q_{syn,1}, D_{1,1})$ and $(Q_1, Q_{title,1})$ to the set of parallel texts . Then, the extracted parallel texts consists of pairs of the form $\{(Q_1, D_{1,1}), (Q_{syn,1}, D_{1,1}), (Q_1, Q_{title,1}), ...(Q_j, D_{m,j}), (Q_{syn,m}, D_{m,j}), (Q_j, Q_{title,m})\}$. This method is called "NS4". Parallel text forms of each method is described in Table 7.4.

Experimental results comparing the methods NS1, NS2, NS3 and NS4 are shown in Table 7.5. NS1 is the simplistic method using noisy channel model. The model showed about 11.51% improvement in performance over the baseline. Adding synthetic queries to the parallel texts has resulted in further improvement in performance. The best performance was obtained with NS3. The title of the document can act a good candidate for pseudo query. In absence of real query logs, with only a large document collection, just titles and documents can be good candidates of parallel texts for training model. In NS4, adding the title of document as a source end i.e., a substitute of documents resulted in decrease of performance. We believe this is because, title serves as a good substitute for query rather than a good document because it is concise and brief similar to a query.

Expt 3: Comparison of Different Training and Testing Schemes

In this experiment, we compared different training context sizes for training and testing

Method	MRR	Improvement
Baseline	0.305	
NS1	0.339	11.51
NS2	0.378	24.34
NS3	**0.386**	**26.97**
NS4	0.374	23.02

Table 7.5: Evaluation: Noisy Channel Results

models. We also experimented different models for learning the translation model. Among different training models, we experimented with default parameters used by GIZA++ and IBM Model, a simple model for training. Among different context sizes, we experimented with content of the whole document and a short snippet of the document. The test data used in this thesis contains only domain name of the clicked URL but does not contain the actual document. Hence, we could not obtain the content of the whole document. Therefore, we used another small dataset for this experiment.

This experiment was carried out on a data set consisting of 7 users. Each user submitted a number of queries to a search engine (Google). For each query, the user examined the top 20 documents and identified the set of relevant documents. Table 7.6 gives the statistics of the data sets. The document collection consists of top 20 documents from the search engine which is actually the set of documents seen by the user while accessing the relevance of the documents. In all, the total size of the document collection was 3,469 documents. We did not include documents of type doc and pdf files. Since this data set contained only documents and not click URLs, we could not use Google as the base search engine to retrieve initial set of documents. We used Lucene[2], an open source search engine as the general search engine to first retrieve a set of results matching the query. For evaluation, we use the 10-fold cross-validation strategy [32]. We divide the data of each user into 10 sets each having (approximately)

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

User	No. Queries	Total No. Rel. docs	Avg. Rel. docs
User1	37	236	6.378
User2	50	178	3.56
User3	61	298	4.885
User4	26	101	3.88
User5	33	134	3.88
User6	29	98	3.88
User7	29	115	3.88

Table 7.6: Statistics of the data set of 7 users

equal number of search queries (For example, for user1 had 37 queries in total, we divided this into 10 sets with 4 queries each approximately). Learning of user profile is done 10 times, each time leaving out one of the sets from training, but using only the omitted subset for testing. Performance is computed in the testing phase for each time and average of the 10 times is taken. In the testing phase, we take each query and re rank the results using the proposed approach using his profile learned from nine other sets. For each query, we compute Precision @10 (P@10), a widely used metric for evaluating personalized search algorithms. It is defined as the number of relevant documents among the top 10 results for a ranking. P@10 is computed by comparing with the relevant documents present in the data.

In extracting parallel texts consists of pairs of the form $\{(Q_i, D_{1,i})\}$ where $D_{1,i}$ is the context extracted from the first relevant document for the query Q_i. We believe that a short snippets extracted in the context of the query would be better candidates for $D_{1,i}$ than using the whole document. This is because there can be a lot of noisy terms which need not right in the context of the query. We believe a short snippet usually N (we considered 15) words to the left and right of the query words, similar to a short snippet displayed by search engines can better capture the context of the query. We use the short snippet given by search engines from the document containing the query terms for the purpose.

We experimented with two context sizes. The first is using the whole document i.e., considering the query and concatenation of all the relevant documents as a pair in the parallel texts extracted which is called $D_{documents}$. The second is using just a short text snippet from the document in the context of query instead of the whole document which is called $D_{snippets}$. The user profile learning from pairs of parallel texts $\{Q, D_{documents}\}$ is called *Document Train*. The user profile learning from pairs of parallel texts $\{Q, D_{snippets}\}$ is called *Snippet Train*. The user profiles are trained using both IBM Model1 and GIZA++ and comparison of the two is shown in Table 7.7.

We also experimented with the size of the context used for testing. Using the document for reranking (called *Document Test*) [3] and using just a short snippet extracted from the document for testing (called *Snippet Test*). We observed that, not only did the model used for training affected P@10, but also the data used in training and testing, whether it was a snippet or document, showed a large variation in the performance. Training using IBM Model1 using the snippet achieved the best results. This is in agreement with the discussion that the snippet surrounding the query captures the context of the query better than a document which may contain many words that could possibly be unrelated to the query, therefore diluting the strength of the models learned. The detailed results for all the users are shown in Figure 7.2 and Figure 7.3.

In Summary, we experimented with two different context sizes for training and test and two different training models. Among the various context sizes, we experimented with snippet and content of the whole document. Our experiments showed best results when training was carried out using snippet using IBM Models for training the model. Over all, training using snippets out performed that using the whole document irrespective of the model. We believe this is because, documents usually contain various terms some of which can be noisy and not relevant and would some times result in deteriorating the performance. Over all, training using IBM Models outperformed that using GIZA++ default parameters. We believe this is because, subtler aspects

[3] It is to be noted that *Snippet Train* and *Document Test* and training using IBM Model1 is the default configuration used for all the reported results unless explicitly specified.

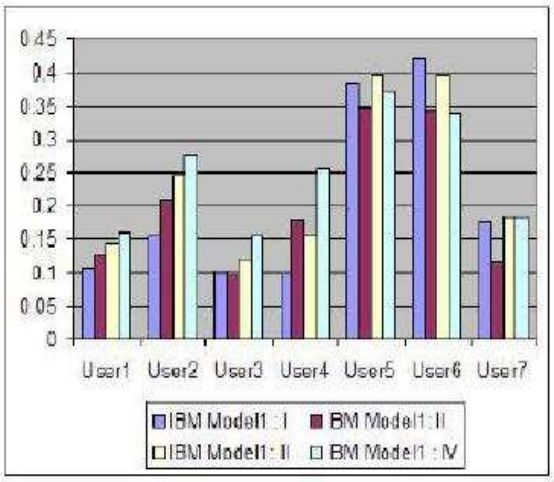

Fig. 7.2: Detailed results for each user with different contexts for training and testing for IBM Model 1

Training Model	**IBM Model1**		**GIZA++**	
	Document Train	Snippet Train	Document Train	Snippet Train
Document Test	0.2062	0.2333	0.1799	0.2075
Snippet Test	0.2028	**0.2488**	0.1834	0.2034

Table 7.7: Comparison of Document Vs Snippet

of language like word order etc might not be very important for retrieval which IBM Models neglect and hence perform better. Table 7.7 shows overall results.

7.6.2 Machine Learning based Approach

In this section, we describe our experiments performed using machine learning methods for personalized web search.

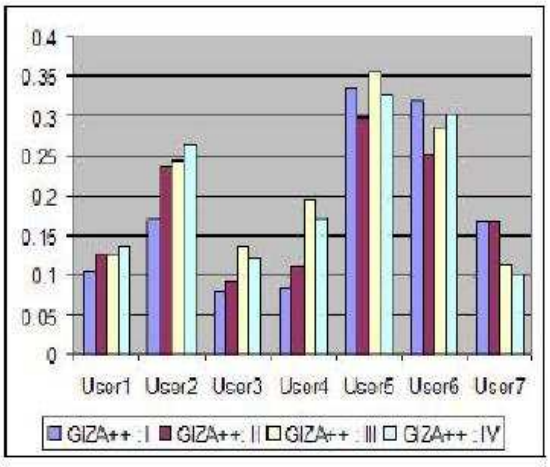

Fig. 7.3: Detailed results for each user with different contexts for training and testing for GIZA++

7.6.2.1 Ranking SVM

We used the ranking SVM framework for learning the user profile. We experimented with a variety of features *Boolean, Term Frequency* and *Normalized Term Frequency*. For *Boolean* weighting, for each clicked document d_i for each unique unigram we assign a weight of 1 if it occurs in d_i and 0 otherwise. For *Term Frequency* weighting, for each clicked document d_i for each unique unigram, we assign a weight of the number of times the word occurs in d_i. Let w be a word in d_i and tf_w be the number of times it occurs in d_i, then weight of w is tf_w. For *Normalized Term Frequency* weighting, for each clicked document d_i for each unique unigram, the weight assigned is as follows. Let w be a word in d_i and tf_w be the number of times it occurs in d_i, then weight of w is $\frac{tf_{w,D}}{|D|\ |Q|}$ where $|D|$ is the length of the document i.e., the number of words in the document and Q is the query length i.e., the number of words in the query. Similarly we do it for bigrams (See Table 7.8 for details).

Our experimental results has shown that unigram features with normalized term frequency weighting has achieved the best performance. The performance with bigram

Method	Features
SVM1	unigrams or single words, Boolean weighting
SVM2	unigrams,Term Frequency weighting
SVM3	unigrams, Normalized Term Frequency weighting
SVM4	bigrams, Normalized Term Frequency weighting
SVM5	unigrams and bigrams, Normalized Term Frequency weighting

Table 7.8: Different features and features weights used in Ranking SVM

Method	MRR	Improvement
Baseline	0.305	
SVM1	0.290	-4.61
SVM2	0.334	9.689
SVM3	**0.369**	21.38
SVM4	0.304	0
SVM5	0.359	18.09

Table 7.9: Evaluation: Ranking SVM

features has not shown improvement. We believe this is because of the huge sparsity introduced due to high dimensionality of the feature space. Since bigrams occur rarely, their frequencies of occurrence was almost the same and probably did not sever as a good indicator for distinguishing.

7.6.3 Without Relevance Feedback

We performed experiments on our approach of personalization without relevance feedback using just the past queries of the user

We compared 4 methods. The first is the Baseline, which is a small variation of the original Rocchio based algorithm [39]. The second method is simple language modeling approach also called *LM* (See Section 4.3). The third is proposed approach called *PSWRF*. The fourth is an extension of the proposed approach called *PSWRF+Query*

LM. We describe the approaches in brief below:

- Baseline

  The baseline is variation of rocchio based algorithm described in Section 7.5.

- LM

  For learning the user profile (See Section 4.3), we simply collect all the words and their probabilities from the implicit relevance feedback for all the training queries. For reranking, for each query, we first retrieve top few results from Google and then rerank them as follows

  $$P(Q|D,u) = \prod_{q_i \in Q} \alpha P(q_i|LM) + (1-\alpha)P(q_i|D)$$

  where $P(q_i|LM)$ is the probability of the word from the user profile and $P(q_i|D)$ is the probability of the word in the document and α is a weighting parameter and lies between 0 and 1.

- PSWRF

  This method (also called *PSWRF*) is the proposed method described in Equation 6.1

- PSWRF+Query LM

  This is an improved version of the proposed method. We smooth the probability of the word in the user profile with a general language model computed from a large number of queries from the query log.

  $$P(q_i|UP) = \beta P(q_i|UP) + (1-\beta)P(q_i|Qlog)$$

  where $P(q_i|Qlog)$ is the probability of the word q_i in the query, $P(q_i|UP)$ is the probability of the word q_i in the user profile UP and β is a weighting variable which takes values between 0 and 1.

The queries posed by the user are short and concise and describe the information need of the user. Though increasing the context to capture user information by using

Method	**MRR**	**Improvement over baseline** (%)
Baseline	0.305	
LM	0.332	8.85
PSWRF	0.350	15.131
PSWRF+Query LM	**0.370**	21.31

Table 7.10: Evaluation: Without Relevance Feedback

the relevance feedback provided by the user has received much attention, there is a possibility that there is sometimes noise in the relevance feedback provided by the users. Our experiments have shown that our approach of personalized search using only queries has shown improvement over a variation of rocchio algorithm. 'PSWRF+Query LM' performed the best. This is because, there was a lot of sparsity in the user profiles learned because only past queries were used in learning the user profile. The rich knowledge present in query language model served as a very good back ground model to fall back to in case some word was missing in the user profile.

7.7 SUMMARY

In this thesis, we proposed three approaches for personalized search.

Language modeling based techniques were found to be quite effective, even a simple method like N-gram based method performed well. We experimented with unigrams and bigrams. Bigram based method outperformed unigram based method. We believe this is because bigrams capture context better. Noisy Channel based method shows an interesting and promising direction of modeling the user context in a well established framework of statistical machine translation. We performed three experiments in this approach. In the first experiment, we compared the noisy channel approach with the baseline where the approach outperformed the baseline. The approach captured the relationship between query and document word well and used it to re rank the results.

In the second experiment, we experimented with different methods of extracting parallel texts. We found that by extracting synthetic queries from snippets of relevant documents, there was improvement in performance. The best performance was found by adding pairs of synthetic queries and snippets of relevant documents and also the pairs of document titles and snippets of relevant documents to the parallel texts. In the third experiment, we experimented with different training and testing schemes i.e., with different training models and training, testing context sizes. Among the various context sizes, we experimented with snippet and content of the whole document. Our experiments showed best results when training was carried out using snippet using IBM Models for training the model. Over all, training using snippets out performed that using the whole document irrespective of the model. We believe this is because, documents usually contain various terms some of which can be noisy and not relevant and would some times result in deteriorating the performance. Over all, training using IBM Models outperformed that using GIZA++ default parameters. We believe this is because, subtler aspects of language like word order etc might not be very important for retrieval which IBM Models neglect and hence perform better.

Ranking SVM provided an potential resource for learning user profile from partial feedback present in click through history. The results show a potential way of making use of ranking SVM for the purpose of personalized search. We experimented with different features and feature weights. Our experimental results has shown that unigram features with normalized term frequency weighting has achieved the best performance. The performance with bigram features has not shown improvement. We believe this is because of the huge sparsity introduced due to high dimensionality of the feature space. Since bigrams occur rarely, their frequencies of occurrence was almost the same and probably did not sever as a good indicator for distinguishing.

Lastly, the results from last approach showed decent improvement over baseline even without use of relevance feedback. This provides an interesting direction to explore if Personalization can be done without relevance feedback. The approach showed improvement over the baseline and a simple unigram based user profile. A

small extension to the approach by using a language model computed from all the queries in the query log has shown a large improvement in performance. This is possibly due to huge sparsity in the user profile which is possibly addressed by the language model learned from the query log.

Our experiments show interesting and promising direction for learning long term user profiles and performing personalization of search results to improve search performance.

CHAPTER 8

Conclusions and Future Directions

In this thesis, we have studied some problems in real world information access from the web through web search engines. The main problem with web search is as follows : There is too much information available on the web; query words used by users often are confusing, ambiguous (a query "Java"can mean the Java island in Indonesia or the Java programming language), and some times are poor descriptors of information need ('SBH' can mean "State Bank of Hyderabad" or "Syracuse Behavioral Healthcare" among others) ; users are often not patient enough to see long list of results given by search engines to find relevant information. It has been observed that users typically only view top few results, usually top 5 or 10, some times 20 and much fewer times 30 and so on. In this scenario, web search can be made more useful, effective and less burdensome to users if search results are customized to each individual user by considering individual user's idiosyncrasies. This could possibly help the user to find the relevant document easily from having to scroll through a long list of results.

We observed lot of interesting issues while studying the problem of customizing search results according to each individual user. What is the information available to customize search results ? How should we customize it? When should we stop customizing it ? How do we know a user liked our customization i.e., how do we evaluate it ? We believe each issue could be an interesting and potential research problem with power to make web search experience better. For example, effective customization of search results according to each individual user and showing most relevant documents to him among the top few documents would help reduce burden on the user. The user will then have better experience with the web search engine, by searching for information of interest, with less effort and quickly.

Earlier work in the literature has studied that past search history of a user can be a good resource for inferring what is relevant to the user. In this thesis, we focused on exploiting this resource and use it to customize search results.

While there is some existing related work, it is far from optimal. We studied the problem taking a different perspective. Our aim is to learn user model from his past search history and use it to customize his search results and show him most relevant documents on the top few. We propose three methods for personalizing of search results. The first two methods use user's implicit relevance feedback or user clickthroughs. They are based on utilizing the past search history consisting of the past queries and their corresponding clickthroughs. The third method attempts personalization based on his past queries alone and does not make use of the user clickthrough data.

In general, learning a model of user interests or behaviour itself is a huge research problem. It has recently gained attention in the context of search. In this thesis, we model a user from logs of his past searches.

In the first approach described in Section 4, we learn a user model by capturing statistical properties of text from his past searches. This statistical properties are stored in what is called a user profile. This user profile of the particular user is later used to customize his search results. We experimented this by capturing different contexts. The different contexts include using single word and two adjacent words described in simple N-gram based techniques in Section 4.3, and capturing relationship between query and document words in Noisy Channel based approach in Section 4.4.

In the second approach described in Section 5, we address the problem of learning a user model using machine learning approaches. Related work on machine learning and search simplifies the task to a binary classification problem with two classes relevant and non-relevant. Such a simplification has several drawbacks. Firstly, because, often when a user gives relevance feedback, he only gives information of documents which he thinks are relevant to him. The user typically does not give information about the documents not relevant to him. For example, assuming all the documents not

identified as relevant to be irrelevant leads to problems. Due to a strong majority of non-relevant documents, a learner will typically achieve the maximum predictive classification accuracy, if it always responds non-relevant, independent of where the relevant documents are ranked. Consider implicit relevant feedback - clickthrough data, it has been shown that the user always clicks documents which appear to be relevant in the top few results. In this case, the user has clicked those results because they might have appeared to be more relevant than the others seen by him in the top few. Hence, clickthrough data is "noisy" and contains only contain partial information and it would be an over simplification to assume a binary classification. We consider the "noise" in clickthrough data which previous machine learning based approaches failed to consider. This is an interesting framework to learn a user model given partial user relevance feedback data.

The third method described in Section 6, is another interesting method where we model a user based on just the past queries posed by the user. We do not make use of clickthroughs of the corresponding queries provided by the user. We only make use of the past queries by the user. The observations and conclusions from this research can be extremely useful and provides an interesting research direction for personalized search because obtaining just the queries is comparatively easy than relevance feedback. The queries can then be used directly for personalization or in combination with available relevance feedback.

Our experiments has shown promising results. All the approaches have shown improvement over the baseline (a variation of rocchio algorithm). Best results were observed for noisy channel based approach, a language modeling based approach described in Section 4.4. This provides a great scope for further research in this direction and using language modeling based approaches in general. Machine learning based method also showed competitive results and is also an interesting direction to explore. The above two approaches have assumed existence of feedback data. The third approach described in Section 6 based on just queries showed promising results.

In language modeling based methods, we experimented with different contexts.

As we increased the context from single word to two words to relationship between query and word, the results showed improvement. In future, much richer contexts can be explored. Hyperspace Analogue to Language model (HAL) constructs the dependencies of a word w on other words based on their occurrence in the context of w in a sufficiently large corpus. The intuition underlying HAL spaces is that when humans encounter a new concept, they derive its meaning from accumulated experience of the context in which the concept appears. Thus the meaning of the new concept can be learned from its usage with other concepts within the same context. HAL can be used to capture more context from past searches of the user to model a richer user model. Other semantic spaces using LSI etc., also seem promising.

Though language models capture rich models and statistics from text, machine learning methods can be used to capture much beyond simple text. Patterns of occurrence or behaviour can also be captured using machine learning models. Consider the following cases. 1) If a user always clicks the first result (or second result) such information could be used in training using machine learning methods to obtain richer user models. 2) Suppose a user always trusts results from a particular web site. In that case, he would be interested in information from that website even though it might be the top most result. Such information about "interesting websites" could be given to the machine learning methods to learn effective user models. 3) Also, it can be possible to make machine learning methods to consider original ranking of search engines while learning user model. In this way, better models can be obtained using machine learning method.

As mentioned earlier, in our approach we use logs of user past searches. There is sometimes repetition in information need of the users. Some times, users type similar queries and search for similar information and click similar documents. Given logs of past searches of large number of users, their searches consisting of queries and clickthroughs, repetition in the queries and clicked documents among different users can potentially be exploited. A study done by us on query log from a popular search engine has shown good repetition in queries and clicked documents among different

users. Past searches of one user can be used to customize search results to another user given that both the user's interests match. This could be one potential and challenging future direction to the current work.

Some times, there could be some documents which might not contain exact query words but might contain related content. Hence, when a user types in a query, such documents would not be returned. Recommending such documents to user could also help user in improving user web search experience. The user profiles learned as described in the current work can be used to also recommend documents to users. This is another interesting future direction to the current work. Collaborative filtering methods are popular methods where they use ratings given a group of users to recommend items of interest to other users. A few data mining based and text filtering methods have also been proposed for recommending items or documents of interests to users. Such methods could also be used in recommendation of documents in extension to current work in future.

Our study of query log data from a popular search engine has lead to some interesting observations. This has also motivated us to see if clickthrough data can be created in an artificial way by simulating behavior of users searching a search engine. This provides a scope for a simulated environment for evaluation for personalized search algorithms. It is a potential area where the outcome of research can directly be used for the benefit of research communities in web search engines and personalization. Given the constraint of non-availability of implicit feedback data it is difficult to evaluate personalization algorithms. Simulated Feedback can be of great use here and help benefit web search community. The benefit from Simulated feedback is two fold. Firstly, it is easy to obtain and also the process of obtaining the feedback data is repeatable. Given a document set and a search engine deployed on this document set, one can start generating simulated feedback in much larger volumes than what can be obtained by either explicit or implicit feedback methods. Secondly, it enjoys the benefit of customizability, where a researcher can customize the creation of the feedback for his purposes, be it testing search engines for specific domains, testing re-

search algorithms in personalization research or testing query modification techniques. We developed a basic system which performs the same.

The contributions from this thesis include a suite of algorithms of personalized web search, our analysis and observations from query log study and a method to simulate user search behaviour and create user feedback in an artificial way. Also several observations were made from several studies during this thesis which provide interesting directions to personalized search research. The two important such observations are i) Can Personalization of Search be done without Relevance Feedback ? and ii) Can Simulation of User Search Behaviour be done.

The work done in this thesis enjoys multiple possible extensions and future directions. To summarize, much richer contexts could be explored to learn better language model based user profiles. Non-text information could also be used to train machine learning methods. It shows a promising direction if personalized search can be done without relevance feedback. The proposed approach for learning user profiles can be used in future for recommending related documents to users, to learn from past search histories of other users etc., The proposed approach of simulated user behaviour is a potential area where the outcome of research can directly be used for the benefit of research communities in web search engines and personalization.

CHAPTER 9

Appendix 1 - Query Log Analysis

9.1 INTRODUCTION

The use of data stored in transaction logs of Web search engines, Intranets, and Web sites can provide valuable insight into understanding the information-searching process of online searchers. This understanding can enlighten information system design, interface development, and devising the information architecture for content collections. There is a large interest in finding patterns in and computing statistics especially of search engine query logs.

Most works choose to answer questions such as statistics about queries, what are the most common queries, what is the average number of words per query, how many queries are included in the average user session, correlations between query terms, and also among other field values [50].

While this work on query log has shown some interesting results, there is very little work which tried to analyze the clicking behaviour of the user. There was some work related this using eye tracking study. An eye tracking study was performed to observe how users formulate queries, assess the results returned by the search engine and select the links they click on [14]. Thirty six undergraduate student volunteers were instructed to search for the answers to five navigational and five informational queries. The former involved finding a specific web page while the latter involved finding some specific information. The subjects were asked to start from the Google search page and find the answers. There were no restrictions on what queries they may choose, how and when to reformulate queries, or which links to follow. Users were told that the goal of the study was to observe how people search the Web, but

were not told of the specific interest in their behavior on the results page of Google. All clicks, the results returned by Google, and the pages connected to the results were recorded by an HTTP proxy. Movement of the eyes was recorded using an ASL 504 commercial eye tracker (Applied Science Technologies, Bedford, MA). More details on the experimental setup are provided in [14].

In this chapter, we discuss our analysis of the query log. We focus on analyzing the clicking behaviour of the users. In particular, we study if there is any general pattern of the user click behaviour. In particular, we aim to answer the following questions: Do all the users go from top to bottom of the results ? Do all the users see the same number of results ?

9.2 QUERY LOG DESCRIPTION

The data used in this thesis is implicit feedback in the form of clickthrough data collected by a popular search engine recently released for research purposes. This collection consists of about 20M web queries collected from approximately 650k users over three months. The data is sorted by anonymous user ID and sequentially arranged.

The data set includes AnonID, Query, QueryTime, ItemRank, ClickURL.

- AnonID - an anonymous user ID number.
- Query - the query issued by the user, case shifted with most punctuation removed.
- QueryTime - the time at which the query was submitted for search.
- ItemRank - if the user clicked on a search result, the rank of the item on which they clicked is listed.
- ClickURL - if the user clicked on a search result, the domain portion of the URL in the clicked result is listed.

Each line in the data represents one of two types of events: 1. A query that was NOT followed by the user clicking on a result item. 2. A click through on an item in the

result list returned from a query. In the first case (query only) there is data in only the first three columns/fields – namely AnonID, Query, and QueryTime (see above). In the second case (click through), there is data in all five columns. For click through events, the query that preceded the click through is included. Note that if a user clicked on more than one result in the list returned from a single query, there will be TWO lines in the data to represent the two events. Also note that if the user requested the next "page" or results for some query, this appears as a subsequent identical query with a later time stamp.

Some basic statistics about the data are as follows.

Dates: 01 March, 2006 - 31 May, 2006

Normalized queries:

36,389,567 lines of data

21,011,340 instances of new queries (w/ or w/o click-through)

7,887,022 requests for "next page" of results

19,442,629 user click-through events

16,946,938 queries w/o user click-through

10,154,742 unique (normalized) queries

657,426 unique user ID's Sample data is shown in Table 9.1.

9.3 QUERY LOG STUDY

In this section, we discuss our analysis of the query log. We analyze the clicking behaviour of the users. In particular, we study if there is any general pattern of the user click behaviour. Do all the users go from top to bottom of the results ? Do all the users see the same number of results.

Given search results, do all users view and click the results in a similar way? Do all the users go from top to bottom seeing and clicking the first result followed by second and so on i.e., Do users always click results while going from top to bottom clicking the top document first before clicking the bottom ones. This have motivated

ID	Query	Query Time	Pos	url
2722	charles drew	2006-03-01 18:00:07	10	http://www.cdhcmedical.com
2722	tricare	2006-03-16 19:07:38	2	http://www.tricareonline.com
142	rentdirect.com	2006-03-01 07:17:12		
142	westchester.gov	2006-03-20 03:55:57	1	http://www.westchestergov.com
142	vera.org	2006-04-08 08:38:427	1	http://www.vera.org
142	broadway.vera.org	2006-04-08 08:39:30		

Table 9.1: Sample Query log Data

us to perform an experiment studying the same.

9.3.1 Query Log study: Experiment 1

The query log data described in Section 9.2 contains an entry called *Position* which is the position of the clicked URL. For each query, we extract the positions of all the clicked URLs. Positions are sorted by the time of click. We then see if the positions are in ascending order, i.e., the top URLs are clicked first before the bottom ones. i.e., the i^{th} document is clicked first than the j^{th} document where i is above j. We observed that in 90% of the queries, users exhibited this kind of behaviour. The queries which violated this behaviour i.e., where users clicked a bottom result first before clicking the top result are called anomaly queries. We found that there were only 10% of anomaly queries (See Table 9.2). However, we then came across another surprising result that 50% of the users exhibited this kind of behaviour. This means that this anomaly

Collection	Total Queries	Anomaly Queries	% Anomaly	Total users	Users with Anomaly Q	%
01	965325	106087	10.99	51956	25301	48.7
02	985530	107947	10.95	52258	25584	48.95
03	1001675	110785	11.05	52309	25803	49.32
04	992893	110129	11.09	52046	25671	49.32
05	1058169	113420	10.71	51840	25381	48.96
06	960136	106323	11.07	52105	25382	48.71
07	999029	108866	10.9	52387	25737	49.12
08	984760	108645	11.03	51978	25664	49.37
09	982528	109593	11.11	52154	25678	49.23
10	978796	109666	11.20	51925	25376	48.87

Table 9.2: Query Log study:Anomaly Queries

behaviour is not dependent on the user specifically but generally common behaviour among users (with 0.5 probability) but may be it is more dependent on the type of queries which are not very common.

9.3.2 Query Log study: Experiment 2

Search engines typically give a large number of results for a query. No user practically sees all the thousands of results given by the search engines. Do all the users same the same number of results? Intuitively we feel that not all the users view the same number of results. While this depends on many factors like the quality and number of results returned by the search engines, it also depends how patient the users are. Some users are less patient and see only the top 1 or 2 results. Some other users are more patient view top 5 results, some view top 10 and some view upto 20 etc. We performed an experiment to see this behaviour. We define *Patience* as a number which describes the patience of the user in how deep the user goes through the search results

and views them. Let Patience be defined as

$$\text{Patience} = \frac{\max_{q \in Q} \max_{i \in clicksforq} Pos_i}{MAX\_CLICK\_POSITION}$$

where Q is the set of all the queries posed by the user and Pos_i is position of the click. $MAX\_CLICK\_POSITION$ is the maximum position of the clicks in the query log data. We found that this was found to be 500. Figure 9.1 shows the plot of the distribution of patience values of each user on log-log scale. Let Patience be defined as

$$\text{Patience} = \frac{\max_{q \in Q} avg_{i \in clicksforq} Pos_i}{MAX\_CLICK\_POSITION}$$

where Q is the set of all the queries posed by the user and Pos_i is position of the click. $MAX\_CLICK\_POSITION$ is the maximum position of the clicks in the query log data. Query log analysis has shown that this was found to be 500. Figure 9.2 shows the plot of the distribution of patience values of each user on log-log scale. Statistical analysis has shown that the patience values of all the users follows a zipf distribution (See Figure 9.1 and Figure 9.2).

9.4 DISCUSSION

In this chapter, we described our query log study to study user click behaviour. Our study lead to some interesting results. 1) We found that in 90% of the queries, users view results from top to bottom. From this, we can possibly infer and assume that users typically view results from top to bottom future future works since it was found to a general behaviour. 2) Another interesting observation is that there were 10% of the queries for which users view a bottom result first. Approximately half of the users exhibited such behaviour. We can infer that may be those queries were "hard" queries and users did not like the ranking of the search engine. Given that users mostly see top results, it would be advantageous if we can show most relevant results to the user on the top to satisfy him.

Statistical analysis of patience has shown that it follows a zipf's law. This kind of analysis is extremely useful especially for characterizing users with these parameters.

Such studies can be a first step in characterizing real users with certain parameters. In our work, we used this result to create artificial users characterizing him with certain parameters like patience etc. Details are described in Appendix II.

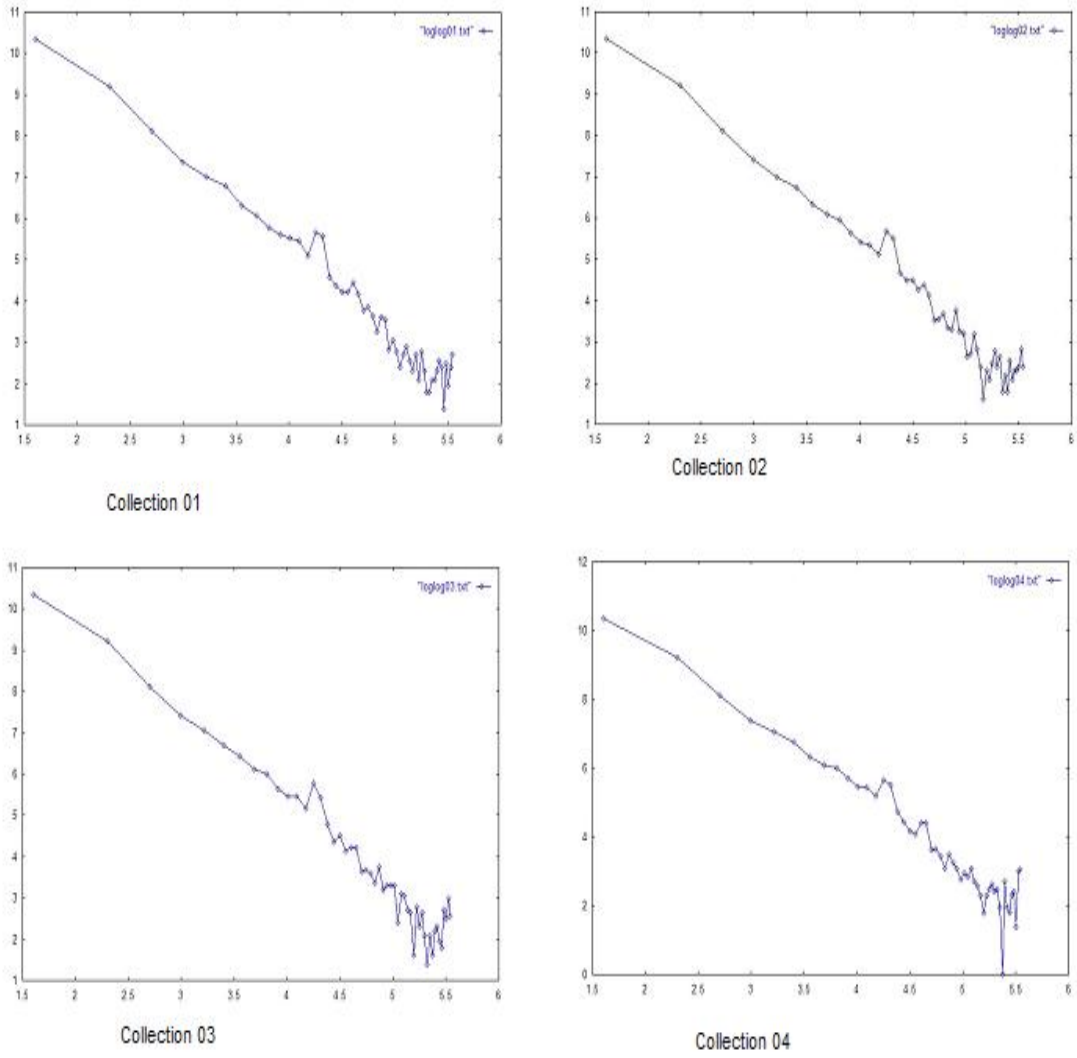

Fig. 9.1: Patience distribution for collections 1 to 4 (Max Patience)

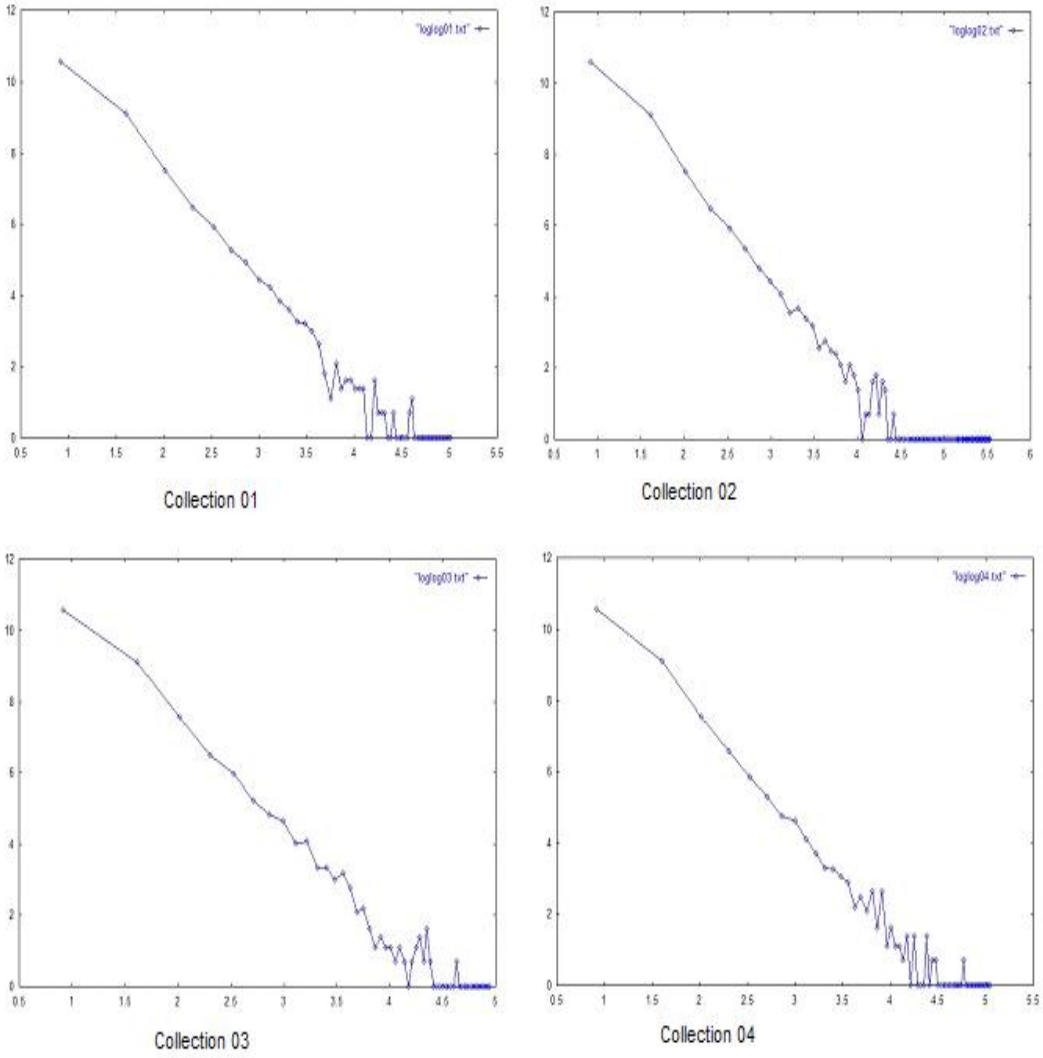

Fig. 9.2: Patience distribution for collections 1 to 4 (Average Patience)

CHAPTER 10

Appendix 2 - Simulated Feedback Creation

10.1 INTRODUCTION

Relevance feedback has received wide attention recently, as a means to capture the search context to improve the search accuracy. Feedback is explicitly collected from the user by asking the users, to mark the documents relevant to them. This is called explicit relevance feedback which has been quite effectively used to improve retrieval accuracy [39] and in personalization of search. Unfortunately, in real world applications, users are usually reluctant to make the extra effort to provide relevant examples for feedback [23].

Implicit relevance feedback methods have been proposed that try to infer user's relevance feedback by observing the user's interactions with the retrieval systems like, dwelling time on a page, amount of scrolling on a page, clicks etc [9]. These approaches do not require any extra user effort [23]. One method of acquiring large volumes of such implicit data is by aggregating the "clickthrough data" or "query log" of a real world search engine. Search engines usually receive millions of queries and clicks from the users every day and therefore act as a potential resource of implicit relevance feedback. The clickthrough data has been used to improve ranking of search engines to improve retrieval accuracy [22], [56] etc.

Explicit feedback methods are expensive to conduct as it requires a lot of manual effort and it is difficult to scale up these methods for gathering large volumes of data. Implicit feedback methods on the other hand are easy to conduct. Most search engines already have logs of user interactions over the web which is a very valuable source for research related to user feedback. However, such feedback is usually not available

to public or even research communities at large for various reasons like - it posing a threat to individual privacy of the web users. Hence researchers that rely a lot on the availability of such feedback either for testing their algorithms [62] or other search engine related problems are faced with the problems of non-availability of user feedback data.

Another major problem with either explicit or implicit feedback data when operating on the web search domain is as follows. Web data is extremely dynamic - millions of pages are created and deleted every day, and the content of existing pages changes quite frequently. Consequently, snapshots of the Web taken several weeks apart may give rise to very different indices. Thus, the feedback data may rapidly become stale, with new pages replacing old ones as the most appropriate online resources for queries. In other cases, even though the "old" expected results remain good resources, search engines will not retrieve them in response to queries . Instead, the engines will return near-duplicate pages, that have equivalent contents but different URLs.

In this thesis, we propose the creation of "Simulated Feedback" (see [40]) (drawing analogy from explicit relevance feedback which is collected by users explicitly specifying the feedback and implicit relevance feedback where the feedback is collected by implicitly inferring the feedback from the user interactions) based on insights from query logs and using artificial methods to generate feedback. It is a potential area where the outcome of research can directly be used for the benefit of research communities in web search engines and personalization. Given the constraint of non-availability of implicit feedback data it is difficult to experiment and evaluate web search related research and especially web search personalization algorithms. Also, due to dynamic content of the web and rapidly changing ranking algorithms of major web search engines the explicit or implicit feedback collected from users operating on major web search engines becomes outdated. Simulated Feedback can be of great use here and help benefit web search community. The benefit from "Simulated feedback" is two fold. Firstly it is easy to obtain and also the process of obtaining the feedback data is repeatable. Given a document set and a search engine deployed on this document

set, one can start generating simulated feedback in much larger volumes than what can be obtained by either explicit or implicit feedback methods. Secondly, it enjoys the benefit of customizability, where a researcher can customize the creation of the feedback for his purposes, be it testing search engines for specific domains, testing research algorithms in personalization research or testing query modification techniques. Creation of Simulated Feedback is done in 2 steps. In the first step, a simulated user is created. A simulated user is an artificial user created with parameters and properties that are as close as possible to a real world web search user. Where applicable we also try to characterize these parameters as probabilistic distributions using large volumes of data from existing search engine query logs. We then create a simulated user by instantiating these parameters by values drawn from these probabilistic distributions. In the second step, feedback from the simulated user is created while he searches a web search engine. We artificially created this step by studying a typical web search process conducted in a real environment The real web search process was identified to have the following steps. 1) First the user formulates the query expressing his information need. 2) The user poses the query to a search engine. 3) The user looks at the results returned by the search engine. 4) Possibly clicks one or more results. 5) If he is not satisfied with the results, he re-formulates the query. In mimicking this web search process, we simulated each of the above four steps with motivations from query log study. Using existing search query logs in our analysis ensures that the data that is generated as part of simulations is as close as possible to real world search scenario and limits any randomness in the procedures. s the benefit of customizability, where a researcher can customize the creation of the feedback for his purposes, be it testing search engines for specific domains, testing research algorithms in personalization research or testing query modification techniques [15] etc.

Although simulation-based methods have been used to test query modification techniques [15] or to detect shifts in the interests of computer users [33], to our knowledge not much research went into creating relevance feedback for web search based on search simulations. White et al [62] created searcher simulations for evaluating

implicit feedback models. The simulation assumes the role of a searcher, browsing the results of a retrieval. The actual relevant and irrelevant documents for a query are given. Simulation of the searchers are created by creating relevance paths using different strategies like they only view relevant/non-relevant information, i.e., follow relevance paths from only relevant or only non-relevant documents, or they view all relevant or all non-relevant information, i.e., follow all relevance paths from top-ranked relevant documents or top-ranked non-ranked documents etc. Their research tries to model only certain phases of the search process like clicking the results and to some extent the process of looking and identifying the results to click. It also does not consider modeling the nature of the searcher in context and also does not calculate the relevance of a document for a user. The search process is not complete without discussing or characterizing the user that participates in the search and computing the relevance of a document for a user.

Radlinski and Joachims [36] describe their system *Osmot* which simulates user behavior model on the web. Documents, queries and document topics were created artificially by sampling words using zipf's law. A typical web user is simulated by assigning him certain characteristic parameters like patience, threshold etc. Web user behavior is simulated with observations from their earlier study on eye movements [14] as follows - the user looks at all the results in succession from top to bottom assessing the relevance for each document, called the perceived relevance. The perceived relevance in this case is a noisy version of the actual relevance of the query to the document which is assumed to be provided to the system prior to the simulations. The perceived relevance is used to decide to click or not click the document. Osmot provides an interesting and promising practical approach to simulating web user and web user behavior. However, the model presented may not capture many of the actual properties and characteristics of the web user in carrying out user simulations. The assumption of availability of actual relevance of the documents makes it less scalable and applicable at places where large volumes of search documents are considered in the generation of simulated feedback. Also, there is no evaluation performed to compare

the simulated data generated with real world scenario.

In our approach we address some of these issues to improve the reliability of the simulated feedback and the scalability of the simulations. We first identify certain parameters that are natural to the search process on the whole and are generic to hold well across search engines. Wherever applicable we try to characterize these parameters as probabilistic distributions, using large volumes of data from existing search engine query logs. We then instantiate these parameters by drawing values from these probabilistic distributions. This ensures that the simulated feedback resembles as closely as possible to the real scenario and thus is of high quality. Also as there are no interventions of a human to provide any kind of extra information or relevance information on the document set, we can easily run the simulations on large sets of documents to create large amounts of simulated feedback.

10.2 CREATING SIMULATED FEEDBACK

We propose an approach for creating simulated feedback. Drawing analogy from explicit relevance feedback (where the feedback is explicitly given) and implicit feedback (where the feedback is implicitly inferred), simulated feedback is a new type of feedback which is created artificially with observations and analysis of a real search engine query log. We propose to create simulated feedback in two phases. As a first step, a simulated user is created artificially by modeling him using parameters identified by analyzing a real search engine query log and instantiating them. In a second step, the web search process is simulated where the simulated user interacts with the search engine, poses queries, looks at the results and possibly clicks one or more results. Simulated feedback consists of these queries and clicked documents by the simulated user.

Simulating a user and the web search process on the whole is an extremely complicated process and has not been much pursued earlier. We make certain assumptions in order to achieve this. The assumptions made are stated at the appropriate places in

the corresponding sub sections. We conducted experiments to cross check the validity of our assumptions for all the assumptions as much as possible. The experiments were conducted on query log data from a major search engine released recently for research purposes. The query log data consists of 20M web queries collected from 650K users over three months. The data consists of anonymous id, query, the time at which the query was posed, the rank of the clicked URL if any, and the clicked URL. The whole data has been organized into 10 collections. Each collection having approximately same number of queries. We use the data from the first collection for our research.

10.2.1 Creating a simulated user

We first observe and analyze a query log to derive certain generic parameters with which we can characterize and represent a user. We have identified the following as some important characters - user identity, patience, relevance threshold, browsing history of the user, type of the user, user interests etc.

The *User Identity* is used to identify a user uniquely and is a unique identity given to a user, say an IP address or an *anonymous id*. *Patience* represents the patience of a user in looking at the results returned by a search engine. It is, how deep a user usually goes down the search results returned by a search engine looking for relevant documents. *Patience* of users varies. Some have very little *patience* i.e., they might look at very few top documents (say 3 or 5) and some might have a lot more who might look at some tens or results. *Relevance Threshold* or Threshold of the user represents how much relevant a document has to be for the user to click on it. Even this, typically depends on the user. Some users have large *Threshold* and are not satisfied until they see a highly relevant document. However, this *Threshold* depends to a large extent on various factors the most important of which is the query. In this thesis we only consider the *anonymous id*, *Patience* and *Threshold* to characterize a user for simplicity and also because we believe that they are the most important characteristics of a user. Other characteristics like browsing history of th user, type of the user etc require much

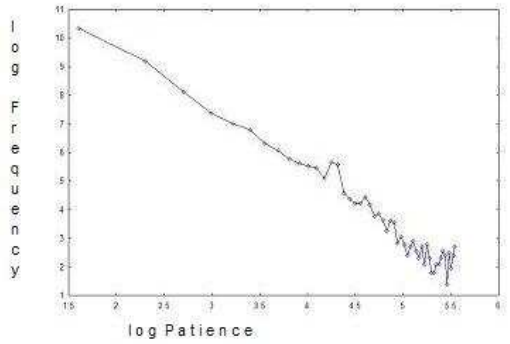

Fig. 10.1: Distribution of Patience on log-log Scale

more user analysis and shall be pursued in future.

As mentioned earlier, our main goal is to be able to build large amounts of simulated feedback which can be a great use for research which is based on user feedback. Hence the data created should be as close to the real world data as possible with high quality so that researchers can use it in real experiments. Hence, we made a small study from the query log of a popular search engine to study how the patience, one of our key parameters, varies across users. Since the threshold depends on the query, we compute the threshold using the query posed by the simulated user. The query log study of patience is as follows. For each user, we computed a patience value as follows. Patience $= \frac{\sum_Q \max_{\mathcal{C}} \text{clicked\_rank}}{\text{No. of queries posed by the user}}$ where Q is a query posed by the user and $\mathcal{C}$ is the corresponding clicked documents for the query Q. and clicked rank is rank of the document clicked among the clicked documents $\mathcal{C}$. $\max_{\mathcal{C}}$ clicked_rank gives maximum clicked rank of a document for a given query. Patience for a user is obtained by averaging this over all the queries posed by him. Statistical analysis has showed that the patience values of different users followed a power law distribution (zipf's law). Figure 10.1 shows the plot of the distribution of patience values of each user on log-log scale.

We characterized a user with the following parameters : *anonymous id, patience* and *Relevance Threshold*. A simulated user is created by instantiating the parameters

with appropriate values. *Anonymous id* is a random number generated unique to each user. *Patience* is randomly drawn from power law distribution. We now create a large number of users by instantiating the user with the above two parameters *Threshold* is picked according to the query.

10.2.2 Simulating a Web Search Process

The web search process we consider is as follows. The user formulates the query expressing his information need (Step 1), and then the user poses the query to a search engine (Step2). He then looks at the results returned by the search engine (Step3), and possibly clicks one or more results (Step4). In the rest of this section, we describe each of the steps in detail.

10.2.2.1 Step 1: Query Formulation

This step involves formulating the query to be posed to the search engine by the user. The simulation of this step can be an extremely difficult and complex step which involves capturing of the user's cognitive process and properly represent the user's information need. Also, the user's interests and his background need to be considered.

We consider a simple and practical approach to simulating this step. The input to this step is a database of search queries. Then, the simulation of query formulation is done by picking a query from the database of queries. Since, in this thesis, we study how close is to a real search engine, we randomly pick a user from a real query log and give all the queries posed by the particular user as input to this step. This helps us to preserve various inter query relations that naturally exists in the subsequent queries of a user.

10.2.2.2 Step 2: Searching the Search Engine

This step involves retrieving the document corresponding to the search query picked in step1. We used Google to retrieve the set documents for a given query. The list of

results with ranking given by Google and the major search engine used in our research are similar and hence we use Google to retrieve the documents for a given query.

10.2.2.3 Step 3: Looking at the Results

In this step, we simulate how a user looks at the results returned by the search engine. The simulation can be done in a number of ways. For example, in a random fashion, from top to bottom, bottom to top etc. We consider a sequential looking of the results by the user from top to bottom motivated by [36] and [14] and our query log study. Also, we assume that a user always clicks a top document first before clicking a document occurring later as shown by the search engine. We conducted a study from query log of a major search engine to support our assumption. For each anonymous user in the query log, we computed the number of queries for which, our assumption that a user clicks a top document first before clicking a bottom document holds good. We observed that for 89% of the queries on an average, the user always clicked a top ranked document and hence we assume the same in our work.

Hence, the user looks at the documents in a sequential fashion from top to bottom until the *Patience* of the user is greater than zero. Once, the *Patience* of the user becomes zero, he cannot look at any further results and the search session is terminated. While looking the results, the user performs clicks of the results as described in step4.

10.2.2.4 Step 4: Clicking the results

This is the most important step in our simulation process. The quality of the simulated feedback created depends to a large extent on this step. In this step, we simulate the user clicks while the user looks at the results. The clicking process of the user is usually affected by the visual information shown by the search engines pertaining to the document. These include the short snippet, the title and URL. This visual information can sometimes misguide the user, making the particular document seeming to be relevant when it is actually not and vice versa. Hence, the relevance perceived

by the user from the visual information would be different from the actual relevance of the document decided by the user upon seeing the whole document. The relevance of a document as seen by the user by seeing just the visual information provided by the search engine is called the *perceived relevance*. The actual relevance of the document is not known.

Clicking of a result would depend on the *perceived relevance* of the document and the *threshold*. The *perceived relevance* of the result has to be more than the *threshold* of the user, for the user to click the document. It has been observed in the eye tracking study by Granka et al. [14] that for clicking a document, a user usually looks at a document immediately below the current document if it is more relevant than the current document or not. If it is, he proceeds to it, otherwise, clicks the current document.

We compute *perceived relevance* using the visual information of the document as follows. Among the visual information, we only use the short snippet in this work. *Perceived relevance* is the textual similarity between the snippet s and the query q. However, the *perceived relevance* would also depend on the other visual information of the document. We also add some noise to this value in order to model various other factors that would effect the perceived relevance.

Relevance threshold for a particular user for a particular session is computed by first averaging the perceived relevances of the top 10 documents and then adding random noise to it.

10.3 EXPERIMENTS

In this section, we describe the experiments we performed using simulated feedback. We describe 2 experiments. In the first experiment, we measure the effectiveness of our proposed approach by comparing the simulated feedback with implicit relevance feedback present in the query logs and by explicit relevance feedback given by judges. In the first experiment, we compare the performance of our web search simulation

UserId	Query	Simulated Click
1	lottery	www.nylottery.org
	ameriprise.com	americanexpress.com
	ask.com	www.ask.com uk.ask.com/ uk.ask.com/?o=312 blog.ask.com
2	ford	www.ffoc.co.uk/ www.ford.com/default.htm www.fleet.ford.com/ www.fordfound.org/
	travelocity	www.travelocity.com/

Table 10.1: A sample of the Simulated Feedback data created

process with three other implementations of the simulation that are randomized which we treat as baselines.

The details of the simulated feedback data used in all the experiments discussed in the rest of the section is as follows. We first create a user by randomly assigning an anonymous id. *Patience* for the user is randomly drawn from power law distribution between 0 and 25. We now begin with simulating the user's web search process. For this, we randomly pick a user from the query log and gather all the queries posed by him. The web search process of the user is simulated for each query in succession and the simulated feedback created by the simulated user is collected. The simulated feedback consists of the anonymous id of the user, the query and the simulated clicks for the query. Similarly we created 500 users and the simulated feedback by all the users is collected. A sample of the data created is shown in Table 10.1. It consists of a total of 69 unique queries and 3 clicks made on an average for the query.

10.3.1 Experiment 1: Comparison with Implicit Feedback

We measure the effectiveness or "goodness" of our proposed approach by comparing the simulated feedback with implicit relevance feedback present in the query logs.

For each query in the simulated feedback described above, we extract a set of all possible relevant documents clicked by all the users in the query log. This is called the *Relevant Document Pool(RDP)*for that query and is assumed to be the set of all and only possible relevant documents present in the query log for this query. There were a total of 69 unique queries and 12.04 documents in the RDP of a query on an average. The effectiveness of the created simulated feedback is evaluated for each query using the RDP as a reference. Accuracy of the simulated feedback for each query is computed as follows

$$\text{Accuracy } = \frac{\text{No. of Simulated clicks in RDP of the query}}{\text{Number of clicks simulated for the query}} \tag{10.1}$$

For each of the 500 users, for each query, we compute the accuracy of the simulated clicks as described in Equation 10.1 and then average it over all the queries and users. We obtained an accuracy of **60.04**% as shown in Table 10.2. We now compare our approach of creating simulated feedback with three other randomized processes as baselines.

10.3.1.1 Random Navigation

In this approach, a user is created without any specific characteristic parameters. Only a unique id is assigned to keep track of the created feedback. In the search process simulation, the query formulation is similar to our proposed approach as discussed in Section 3.2. However we have a predefined number 'N' for the clicks that he could perform and provide as feedback. These N results are picked randomly with uniform probability from the list of the results that the search engine returns for the query.

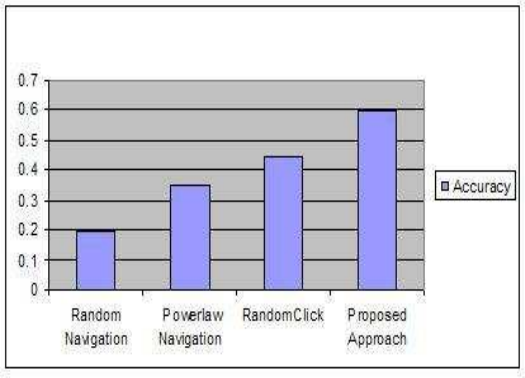

Fig. 10.2: Comparison of Accuracy of Proposed Approach with Random Navigation, Powerlaw Navigation and Random Click

10.3.1.2 Power-law Navigation

This is similar to the previous approach i.e., Random Navigation of Results except that instead of picking a random number with uniform probability, we pick a number from a power law distribution. A user typically sees and clicks top results more often and click. Also, it has been observed that the clicks made by the users follows a power law distribution.

10.3.1.3 Random Click

This approach is quite similar to our proposed approach discussed in Section 3, except for the step4. In simulating step4 where we simulate a click we usually compare if the perceived relevance is greater than the relevance threshold. In Random Click, the perceived relevance and relevance and patience are all drawn at random with uniform probability.

For each user in the simulated feedback data created (described in the beginning of this sub-section), we simulate feedback data using the above three methods and compute accuracy by comparing from the query log using Equation 10.1. As it can be seen in Figure 10.2, our approach performed better than all the three approaches.

	Judge1	**Judge2**	**Judge3**	**Judge4**
Judge1	1			
Judge2	0.8131	1		
Judge3	0.8334	0.9907	1	
Judge4	0.9586	0.9039	0.9171	1

Table 10.2: Correlation of feedback between the judges

10.3.2 Experiment 2: Comparison with Explicit Feedback

We performed another experiment in which we performed manual evaluation of our simulated feedback. Since manual evaluation requires a lot of effort, we performed it using 4 judges on a small sub set of the simulated feedback data created. We randomly selected 25 users from the simulated feedback data. We picked randomly 1 query per each simulated user hence picking 25 different queries. We also took the simulated feedback generated for these 25 queries by the respective simulated users. Each of the judge was given an evaluation form to fill his feedback in. The evaluation form contained instructions for feedback, details about the judge, his name, email etc and a table containing the query, simulated clicks. For each click, the judge was asked to give his feedback a value of 1 or 0. A value 1 is to be given if the the URL appears to be relevant for the query and 0 otherwise. From the boolean feedback provided from the judges we computed Judge Accuracy of the simulated clicks

$$Judge\ Accuracy = \frac{\text{Number of relevant clicks}}{\text{Number of clicks simulated for the query}}$$

The *Judge Accuracy* is then averaged over all users and over all queries and all judges. We achieved an average *Judge Accuracy* of **66.02**%. The correlation between the feedback from the 4 judges was found to be high (see Table 10.2). Correlation between the accuracy calculated from the judges' accuracy and the accuracy calculated from the query log was found to be **0.859**. Figure 10.3 shows the accuracy calculated from query log and *Judge accuracy* for the 25 queries.

10.4 DISCUSSION

There was a 6% increase in the accuracy over the accuracy compared with query log which is due to various reasons. The first of which is due to matching problems while computing the accuracy. The second is due to too low number of documents in the RDP for certain queries.

While matching the simulated click document with a document in the RDP of the query, we compare the URLs of the two documents. We compare only the domain names ignoring the actual documents. For example, a simulated click like "www.mymmode.com/messagecenter/" would be reduced to "www.mymmode.com" and this is then compared with the documents in the RDP. This is because, the query log data contained only the domain name of the clicked url.

We faced a few problems while making these comparisons, due to varying representations of the URL of a particular website prevalent on the Internet. Another issue is the change in the index of the search engine and the ranking of documents in some documents. For example, consider the query 'the child's wonderland company' the query log data contained 'www.wonderlandtheatre.com' as the clicked document with rank of the click as 6. However, with the current ranking, we searched for the same URL in the top 1000 documents and the top 1000 documents did not contain the URL.

Also for a focused query like "qualcomm.com" there was only one clicked document "http://www.qualcomm.com" hence, our simulator's task is then to retrieve only this clicked document to achieve a non zero accuracy. It is natural that the query has only click document because the information need in this case is focused, probably going to the home page of qualcomm. Also the number of documents in the RDP depends on the number of users who posed the query. When more number of users who the query there is greater probability of the number of documents in the RDP. A query like "qualcomm.com was focussed and posed by only one user. A more general query like "lottery" was posed by 58 users and hence had 24 unique click URLs. There were many examples present in the query log data for the issues discussed above.

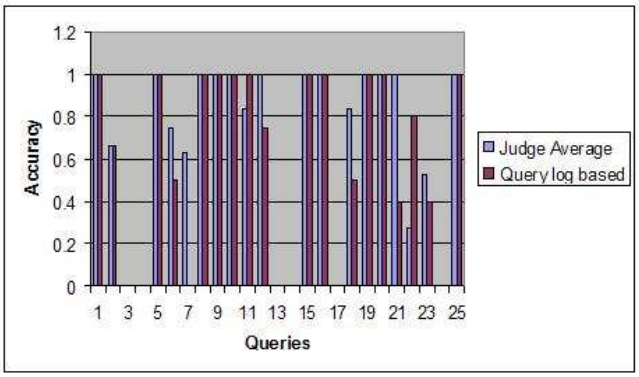

Fig. 10.3: Comparison of Query log based and Judge evaluation for 25 queries

REFERENCES

[1] Pretschner A. and Gauch S. Ontology based personalized search. In *ICTAI.*, pages 391–398, 1999.

[2] Jamie Allan and et. al. Challenges in information retrieval language modeling. In *SIGIR Forum*, volume 37 Number 1, 2003.

[3] Croft W. B and Lafferty J (eds.). *Language Modeling for Information Retrieval*. Kluwer, 2003.

[4] Evelyn Balfe and Barry Smyth. An analysis of query similarity in collaborative web search. In *In Proceedings of the European Conference on Information Retrieval*, pages 330–344. Springer-Verlag, 2005.

[5] Adam Berger and John D. Lafferty. Information retrieval as statistical translation. In *Research and Development in Information Retrieval*, pages 222–229, 1999.

[6] Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. The mathematics of statistical machine translation: parameter estimation. *Comput. Linguist.*, 19(2):263–311, 1993.

[7] Boris Chidlovskii, Nathalie Glance, and Antonietta Grasso. Collaborative reranking of search results. In *Proc. AAAI-2000 Workshop on AI for Web Search.*, 2000.

[8] M. Claypool, M. Waseda P. Le, and D. Brown. Implicit interest indicators. In *Proceedings of Intelligent User Interfaces 2001*, pages 33–40, 2001.

[9] Mark Claypool, Phong Lee, Makoto Wased, and David Brown. Implicit interest indicators. In *Intelligent User Interfaces*, pages 33–40, 2001.

[10] C. Cortes and V. N. Vapnik. Supportvector networks. *Machine Learning Journal*, 20:273–297, 1995.

[11] Larry Fitzpatrick and Mei Dent. Automatic feedback using past queries: Social searching? In *In Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 306–313. ACM Press, 1997.

[12] Jill Freyne, Barry Smyth, Maurice Coyle, Evelyn Balfe, and Peter Briggs. Further experiments on collaborative ranking in community-based web search. *Artificial Intelligence Review*, 21(3-4):229–252, 2004.

[13] Natalie S. Glance. Community search assistant. In *In Proceedings of the International Conference on Intelligent User Interfaces*, pages 91–96. ACM Press, 2001.

[14] L. Granka, T. Joachims, and G. Gay. Eyetracking analysis of user behavior in www search. poster abstract. In *Proceedings of the Conference on Research and Development in Information Retrieval (SIGIR '04)*, 2004.

[15] D. Harman. Towards interactive query expansion. In *Proceedings of the 11th Annual ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 321–331, 1988.

[16] R. Herbrich, T. Graepel, and K. Obermayer. Large margin rank boundaries for ordinal regression. *Advances in Large Margin Classifiers*, page 115132, 2000.

[17] D Hiemstra. A linguistically motivated probabilistic model of information retrieval. In *Proceedings of the Second European Conference on Research and Advance Technology for Digital Libraries (ECDL)*, pages 569–584, 1998.

[18] T Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd Annual International ACM SIGIR Conference*, 1999.

[19] Armin Hust. Query expansion methods for collaborative information retrieval. *Inform., Forsch. Entwickl.*, 19(4):224–238, 2005.

[20] P. Ingwersen and N. Belkin. Information retrieval in context irix. *SIGIR Forum*, 38(2), 2004.

[21] Jian-Yun Ji-Rong Wen and Hong-Jiang Zhang. Query clustering using user logs. *ACM Transactions on Information Systems (TOIS)*, 20(1):59–81, 2002.

[22] Thorsten Joachims. Optimizing search engines using clickthrough data. In *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142, New York, NY, USA, 2002. ACM Press.

[23] D. Kelly and N.J. Belkin. Reading time, scrolling and interaction: Exploring implicit sources of user preferences for relevance feedback during interactive information retrieval. In *Proceedings of the 24th Annual International Conference on Research and Development in Information Retrieval (SIGIR '01)*, pages 408–409, 2001.

[24] D Kelly and J Teevan. Implicit feedback for inferring user preference: A bibliography. In *SIGIR Forum*, volume 32, 2003.

[25] J. Lafferty and C Zhai. Document language models, query models, and risk minimization for information retrieval. In W.B. Croft, D.J. Harper, D.H. Kraft, and J. Zobel, editors, *Proceedings of the 24th annual international ACM-SIGIR conference on research and development in information retrieval*, pages 111–119, New Orleans, Louisiana, 2001. New York: ACM.

[26] V. Lavrenko, M. Choquette, and W. B Croft. Cross-lingual relevance models. In *Proceedings of the 25th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, 2001.

[27] Victor Lavrenko and W. Bruce Croft. Relevance-based language models. In *Research and Development in Information Retrieval*, pages 120–127, 2001.

[28] Henxi Lin., Gui-Rong Xue., Hua-Jun Zeng., and Yong Yu. Using probabilistic latent semantic analysis for personalized web search. In *Proceedings of APWEB'05*, 2005.

[29] Fang Liu, Clement Yu, and Weiyi Meng. Personalized web search by mapping

user queries to categories. In *Proceedings of the eleventh international conference on Information and knowledge management, ACM Press*, pages 558–565, 2002.

[30] C. D. Manning and H Schtze. *Foundations of Statistical Language Processing*. The MIT Press, Cambridge, Massachusetts, 1999.

[31] D. Miller, T. Leek, and R. Schwartz. A hidden markov model information retrieval system. In *Proceedings on the 22nd annual international ACM SIGIR conference*, pages 214–221, 1999.

[32] Tom Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[33] J. Mostafa, S. Mukhopadhyay, and M. Palakal. Simulation studies of different dimensions of users' interests and their impact on user modelling and information filtering. *Information Retrieval*, 6:199–223, 2003.

[34] Franz Josef Och and Hermann Ney. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51, 2003.

[35] Jay M. Ponte and W. Bruce Croft. A language modeling approach to information retrieval. In *Research and Development in Information Retrieval*, pages 275–281, 1998.

[36] Filip Radlinski and Thorsten Joachims. Evaluating the robustness of learning from implicit feedback. In *ICML Workshop on Learning In Web Search*, 2005.

[37] Vijay V. Raghavan and Hayri Sever. On the reuse of past optimal queries. *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 344–350, 1995.

[38] S. E. Robertson and K. Sparck Jones. Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27(3):129–146, May-June 1976.

[39] J. J. Rocchio. Relevance feedback in information retrieval, the smart retrieval system. *Experiments in Automatic Document Processing*, pages 313–323, 1971.

[40] U. Rohini, Vamshi Ambati, and Vasudeva Varma. Creating simulated feedback. Technical report, International Institute of Information Technology, 2007.

[41] U. Rohini, Vamshi Ambati, and Vasudeva Varma. Personalized search without relevance feedback. Technical report, International Institute of Information Technology, 2007.

[42] U. Rohini, Vamshi Ambati, and Vasudeva Varma. Statistical machine translation models for personalized search. Technical report, International Institute of Information Technology, 2007.

[43] U. Rohini and Ambati Vamshi. A collaborative filtering based re-ranking strategy for search in digital libraries. In *proceedings of 8th ICADL - 2005*, 2005.

[44] U. Rohini and Vasudeva Varma. A novel approach for re-ranking of search results using collaborative filtering. In *Proceeedings of International Conference on Computing: Theory and Applications (ICCTA'07)*, pages 491–495, Kolkota, India, March 2007.

[45] R Rosenfeld. Two decades of statistical language modeling: where do we go from here? In *Proceedings of the IEEE*, 2000.

[46] C. E Shannon. Prediction and entropy of printed english. *Bell System Technical Journal*, 30:50–64, 1951.

[47] X. Shen., B. Tan., and C. Zhai. Context-sensitive information retrieval using implicit feedback. In *Proceedings of SIGIR 2005*, page 4350, 2005.

[48] X. Shen and C. Zhai. Exploiting query history for document ranking in interactive information retrieval (poster). In *Proceedings of SIGIR 2003*, pages 377–378, 2003.

[49] L. Si, R. Jin, J. Callan, and P Ogilvie. Language modeling framework for resource selection and results merging. In *Proceedings of the Eleventh International Conference on Information and Knowledge Management (CIKM'02)*, 2002.

[50] Craig Silverstein, Monika Henzinger, Hannes Marais, and Michael Moricz. Analysis of a very large altavista query log. Technical Report 1998-014, Digital SRC,

1998.

[51] Barry Smyth, Balfe, Oisin Boydell, Keith Bradley, Peter Briggs, Maurice Coyle, and Jill Freyne. A live-user evaluation of collaborative web search. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI'05), Edinburgh, Scotland*, 2005.

[52] Barry Smyth, Evelyn Balfe, Peter Briggs, Maurice Coyle, and Jill Freyne. Collaborative web search. In *In Proceedings of the 18th International Joint Conference on Artificial Intelligence, IJCAI-03*, pages 1417–1419. Morgan Kaufmann, 2003.

[53] Barry Smyth, Evelyn Balfe, Jill Freyne, Peter Briggs, Maurice Coyle, and Oisin Boydell. Exploiting query repetition & regularity in an adaptive community-based web search engine. *User Modeling and User-Adapted Interaction: The Journal of Personalization Research*, pages 383–423, 2004.

[54] F. Song and W. B. Croft. A general language model for information retrieval. In *Proceedings on the 22nd annual international ACM SIGIR conference*, page 279280, 1999.

[55] Micro Speretta and Susan Gauch. Personalizing search based on user search histories. In *Thirteenth International Conference on Information and Knowledge Management (CIKM 2004)*, 2004.

[56] K. Sugiyama, K. Hatano, and M. Yoshikawa. Adaptive web search based on user profile constructed without any effort from users. In *Proceedings of WWW 2004*, pages 675 – 684, 2004.

[57] B. Tan, X. Shen, and C. Zhai. Mining long term search history to improve search accuracy. In *Proceedings of 2006 ACM Conference on Knowledge Discovery and Data Mining (SIGKDD'2006)*, pages 718–723s, 2006.

[58] J. Teevan, S. T. Dumais, and E. Horvitz. Personalizing search via automated analysis of interests and activites. In *Proceedings of SIGIR 2005*, 2005.

[59] Rohini Uppuluri and Vamshi Ambati. Improving search results using collaborative

filtering. In *Proceedings of* 4^{th} *Intelligent Techniques for Web Personalization (ITWP), Workshop held at AAAI 2006*, Boston Massauchusetts, 2006.

[60] C. J van Rijsbergen. A theoretical basis for the use of co-occurrence data in information retrieval. *Journal of Documentation*, 33(2):106–119, 1977.

[61] E. M Voorhees. The trec-8 question answering track report. In *Proceedings of the eighth Text REtrieval Conference (TREC-8)*, page 7782. NIST Special Publication 500-246, 2000.

[62] Ryen W. White, Ian Ruthven, Joemon M. Jose, and C. J. van Rijsbergen. Evaluating implicit feedback models using searcher simulations. *ACM Transactions on Information Systems (ACM TOIS)*, 23(3):325–361, 2005.

[63] J. Xu and W. B. Croft. Cluster-based language models for distributed retrieval. In *Proceedings of the 22nd Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pages 254–261. New York: ACM, 1999.

[64] J. Xu, R Weischedel, and C Nguyen. Evaluating a probabilistic model for cross-lingual retrieval. In *Proceedings of the 24th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pages 105–110, 2001.

[65] G. K Zipf. Relative frequency as a determinant of phonetic change. *Harvard Studies in Classical Philology*, 40, 1929.

[66] G. K Zipf. *Selected Studies of the Principle of Relative Frequency in Language*. Harvard University Press., 1932.

[67] G. K Zipf. *Human Behavior and the Principle of Least Effort*. Addison Wesley Press, 1949.

[68] G. K Zipf. *The Psycho-Biology of Language: An introduction to Dynamic Philology*. MIT Press, Cambridge, 1965.