Presented by: **Nicolaas Matthijs**

Supervised by:
Dr. Filip Radlinski (Microsoft Research)
Dr. Stephen Clark (University of Cambridge)

Personalized Search

Declaration of originality

I, Nicolaas Matthijs of Queens' College, being a candidate for M.Phil in Computer Speech, Text and Internet Technology, hereby declare that this dissertation and the work described in it are my own work, unaided except as may be specified below, and that the dissertation does not contain material that has already been used to any substantial extent for a comparable purpose.

Signed

Date: 17/06/2010

Word count: 14,983 (Excluding conference paper draft)

Personalized Search

Preface

This thesis presents a research project conducted in Easter Term 2010 in order to obtain the MPhil Computer Speech, Text and Internet Technology at the University of Cambridge. It investigates the use of profile based personalized web search using browsing history and thorough offline and online evaluation in a complete and self-contained research project. It involves the full research cycle and includes pre-research investigation, suggesting and implementing a new personalization strategy, capturing data, finding participants, two different evaluations, parameter investigation, comparison to previous best systems and making the developed Firefox add-on available for real-life usage.

All of the described work is original and was an inherent part of this research project. However, certain parts of the implemented code make use of 3rd party code (e.g. C&C Parser, OpenNLP Tools, etc.). When this is the case, this will be mentioned in a footnote. All of the code written for this project has been open sourced and can be found at http://github.com/nicolaasmatthijs/AlterEgo.

This project wouldn't have been possible without the help and time of others. I would like to thank my external supervisor, Dr. Filip Radlinski, for our inspirational weekly meetings, his willingness to help, his interesting suggestions and proofreading the content of this thesis. I would also like to thank his employer, Microsoft Research Cambridge, for making some of his time available. I would also like to thank my internal supervisor, Dr. Stephen Clark, for keeping an eye on the project and making sure it stayed on track.

This project couldn't have succeeded without the help of the 50 participants that have volunteered to install both versions of the developed Firefox add-on, which has provided me with real user data and evaluation results. I would especially like to thank the 6 people that were willing to spend an entire afternoon as participants in the offline evaluation session.

A special thanks also goes to the Center for Applied Research in Educational Technologies (CARET) at the University of Cambridge for providing me with a publicly visible server that was extensively used throughout the entire research project.

1. Abstract

In this thesis, various algorithms that use a person's complete browsing history as input data for web search personalization are studied. Using the specific characteristics of the web and more advanced NLP techniques, it is attempted to implicitly learn a user's interests and generate an interest profile. An end to end search personalization system is developed, being the first to give a thorough evaluation with both offline and online experiments. One of the additional goals of this thesis was to develop a scalable and user-friendly tool that is directly useful and applicable to end users, and can be downloaded and used as a Firefox add-on. In doing this, results that are significantly better than the default search engine ranking or previously published personalized search strategies were obtained.

2. Introduction

Although information retrieval systems like web search have become an essential part of our lives, there is still room for improvement. A major deficiency of current retrieval systems is that they generally lack user modeling and are not adaptive enough to a user's individual needs and interests [10]. This can be illustrated with the search query “ajax”. This query will return a wide range of results, including websites about Ajax based web development, websites about the Dutch football team Ajax Amsterdam and websites about the cleanser named Ajax. Web developers would most likely be interested in the websites about Ajax based web development, whilst sports fans would most likely be interested in the Dutch football team. People without these interests would most likely be looking for information about the cleanser. Without personalization however, all users will be presented the same ranking.

On top of that, previous research has noted that the vast majority of search queries are short [25, 11] and ambiguous [5, 22]. Often, different users will use the same query to express a completely different information need and goal [10, 11, 19, 23, 35]. Given all of these query characteristics and the ever increasing amount of information available on the web, the original “one ranking suits them all” idea behind early retrieval systems becomes less and less appropriate. Personalized search, which aims at custom tailoring the results returned by a retrieval system to an individual user's information need and general interests, is a potential solution to that problem.

A large range of personalization strategies have already been suggested [31, 19, 2, 3, 15, 18, 26, 28, 30, 7, 29, 23, 17, 4, 7, 16, 9, 24, 12, 8]; however this thesis manages to improve on these whilst still achieving a realistic and scalable implementation. In order to improve the retrieval quality, the user, his interests and his search and browsing behavior need to be modeled. This model can then be used to adjust search results to the user's specific individual needs.

In contrast, a non-personalized retrieval system only has access to a user's query to model its information need. Given the nature of queries, this representation of the user's information need can clearly be insufficient. Therefore, the retrieval system will need access to more information about the user and his information need.

The data used to construct such a user model can either be obtained in an explicit or an implicit manner. When gathered in an explicit way, the user is asked explicitly to indicate which of the retrieved documents are relevant to the user's information need, which is called *relevance feedback*. The user interests can then be modeled based on these relevant documents. Although it has been shown that relevance feedback can be quite effective for improving retrieval quality [1, 21], it is an unrealistic solution for real world applications. Users would be asked to put in extra effort, which is something they are usually very reluctant to do [14]. It is also an important goal of this thesis to produce a personalized output that does not require a change in the way people are searching, whilst still improving the quality of the retrieved results and thus achieve an optimal user experience.

Therefore, this thesis will limit itself to the use of implicitly collected information about the user. In previous research, query history and click-through data are often used to model the user’s interests, as for example in [27]. Although it is shown that

this can improve retrieval quality, such data is often sparse and additional information about the user can provide better understanding of the user's interests and information needs. Teevan et al [31] make use of a much richer user representation by utilizing a Desktop index which indexes files on the user's hard drive, e-mails, visited web pages and so on. However, this approach treats web documents as common documents and does not take advantage of the characteristics and structure encapsulated within a web page. This thesis makes use of the richness of a user's complete browsing history, but the specific characteristics and structure of web pages are also being exploited. Next to that, it is also attempted to apply more advanced NLP techniques to these web documents and investigate whether the noisy nature of web pages has a negative effect on this. It is found that this approach and taking advantage of web document structure both visibly improve retrieval quality.

Once a user's browsing behavior and interests have been learned, they can be used to re-rank the results returned by a search engine. For instance, a person who is a web developer issues the search query “Ajax”. It is likely that both the search results about web development and the user's profile will contain words that are indicative of web development. These results can then be promoted to the top of the ranking based on similarities between the user profile and the relevant search results. In other words, it is attempted to increase the chance of getting a relevant result near the top of the ranking.

Previous research [32] suggests that such profile based personalization may lack effectiveness on unambiguous queries like ”london weather forecast” and therefore no personalization should be attempted for these queries. However, if this or a related query has been issued by this user before, a preference for certain weather forecast websites could be detected by using the user's URL history, which can also be deduced from the browsing history. This thesis therefore expands upon a method which successfully incorporates a user profile and URL history into a personalized search framework [8]. However, there still exist scenarios in which search personalization might not help or even harm, for example when a query can not be personalized or when a user's information need changes over time. The method suggested in this thesis keeps this in mind by also assessing a personalization strategy that is not too aggressive and still allows potentially less relevant results in at a slightly lower rank.

In summary, this thesis describes a method that uses browsing history and URL history as input data, exploiting web pages' specific structure and characteristics and applying more advanced NLP techniques. A Firefox add-on, called AlterEgo[1], that makes this method available to end users is also developed. It is shown that it yields significant retrieval improvements over web search and other suggested personalization methods without any effort on the user's behalf and without changing the user's search environment or behavior.

The remainder of this thesis is organized as follows. Section 3 describes related work. Section 4 gives an overview of the different user profile generation and re-ranking strategies investigated. Section 5 describes the evaluation methods for personalized search. Section 6 discusses an offline evaluation method which is used to select the best personalization approaches. Section 7 describes an online evaluation approach that allows to more reliably evaluate the quality of the personalized rankings. Section 8 offers some concluding remarks and describes some open questions and possible directions for future research.

3. Related Work

Many search personalization strategies have been suggested over the last years. In this section, two groups of methods that have mainly been used in previous research are described.

Profile Based Approaches

Some personalization techniques [31, 19, 28] are based on a user profile that expresses the individual's interests and browsing behavior. This can be done explicitly through information provided by the user, which this thesis will not consider

[1] http://alterego.caret.cam.ac.uk

due to the extra effort involved on the user's behalf. Various methods and a wide range of information sources have also been proposed to learn a user's profile implicitly without any user effort.

In [7], the user profile is inferred from the entire search history and is used to model long term user interests. Shen et al [23] make use of the recent search history to model the short term user interests, in which session boundaries are used to define short term search history. These methods often suffer from data sparsity and very frequently re-rank results relying on a very limited amount of data.

Other methods have attempted to incorporate more information about the user by using the full browsing history [28, 17]. The Curious browser, a web browser developed to record a user's explicit relevance ratings of web pages and browsing behavior when viewing a page, such as dwell time, mouse clicks and scrolling behavior, is described in [4]. The most promising profile based approach was suggested by Teevan et al. [31]. They use a rich model of user interests, built from both search-related information, previously visited web pages and other information about the user (e.g. documents on their hard drive, e-mails etc.) to re-rank web search results within a relevance feedback framework. In doing this, they obtain a significant improvement over default web ranking. The method suggested in this thesis is compared to this approach in section 6 and significant improvements in retrieval performance are shown. The method described in [2] is based on solely using a user's desktop information.

Concerning the model used to describe a user, user interests can be represented as a set of keyword vectors [6], a set of concepts [16], an instance of a predefined ontology [9, 24, 18. 30] or a hierarchical category tree based on ODP[2] and corresponding keywords [15, 3]. This thesis will focus on modeling users through a vector of weighted terms.

Profile Based Approaches

A different range of personalization strategies utilize URL and click-through data from past queries. In [12], user click-through data is collected as training data to learn a retrieval function, which is used to produce a customized ranking of search results that suits a group of users' preferences. In [28, 26, 29], the click-through data collected over a long time period is exploited through query expansion to improve retrieval accuracy. The most promising URL and click-through based method seems to be PClick, or person-level re-ranking based on historical clicks, as suggested in [8]. If a query is issued by a user for the second time, pages that have been clicked during the first search for this query are promoted to the top of the ranking. A disadvantage of this approach is that it can only be applied to repeated queries. The method suggested in this thesis will incorporate both a user profile and a user's URL and click-through history. The approach described in this thesis is compared to the PClick method in section 6, and significant improvements are found.

Commercially Deployed Systems

Recently, personalized search has also been made available in some of the mainstream web search engines like Google[3] and Yahoo!. These appear to use a combination of explicitly and implicitly collected information about a user. They allow the user to build a profile for themselves by selecting categories of interests and custom tailor the results delivered to the user based on that. However, in practice few users are expected to provide this explicit information. Implicitly, a user's search and click-through history is used to bring previously visited pages and websites to the top. In addition, the results returned by the search are adapted to the geographical location of the user and websites closer to that geographical location will be favored, which is known as localized search[4]. However, as the details of these methods and algorithms are not publicly available, this thesis will only compare to the default search engine ranking and not the personalized version.

---

[2] The Open Directory Project (ODP) is a hierarchical ontology scheme for organizing site listings

[3] http://googleblog.blogspot.com/2009/12/personalized-search-for-everyone.html

[4] http://googleblog.blogspot.com/2009/04/google-becomes-more-local.html

4. Personalization Strategies

In this section, the personalization approach suggested in this thesis and the different parameters involved are described. The first step consists of implicitly learning a user's interests and constructing a user profile. In the second phase, this user profile is used to re-rank retrieved search results.

4.1 User Profile Generation

In this thesis, a user is represented as a list of terms and weights associated to those terms, a list of visited URLs and the number of visits to each of them and a list of past search queries and pages clicked for these search queries. This is a global profile, one-sidedly applied to all search queries, aimed at modeling a user's long term interests and browsing behavior.

The base assumption behind the suggested method is that it can be assumed that most of the pages within a user's browsing history are relevant to that user and can be used to model the user's interests. Inevitably, non-relevant pages and advertisements will be part of the browsing history as well, but it will be investigated in the evaluation phase whether this causes poor performance or whether these non-relevant pages are outweighed by the actual relevant pages.

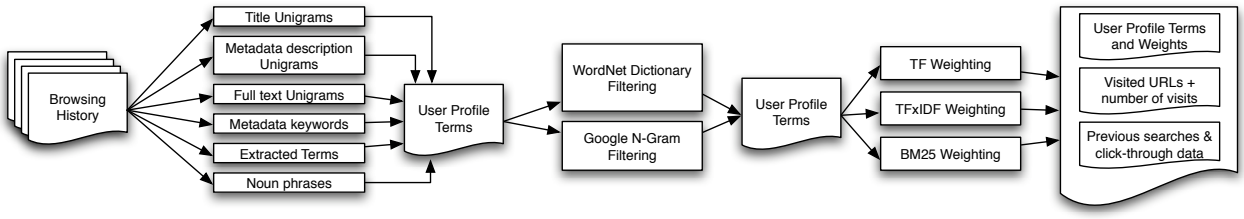


Figure 1: User Profile Generation Steps and Workflow

The generation of a user's profile is a multi-step process that is shown in Figure 1. First, a user's browsing history is collected and stored as a set of URLs with accompanying HTML content. In the next step, this browsing history is processed in order to produce a set of more meaningful data sources that can be used to generate the list of terms to be associated with the user in step 3. Next, the terms can be filtered through various mechanisms and finally term weights are generated using a range of weighting algorithms, producing the user's profile. Each of these steps will now be described in detail.

4.1.1 Data Capture

As the methods this thesis describes use a person's complete browsing history as the input data to personalize search, the first step was to capture people's browsing history as complete and accurately as possible, whilst keeping privacy issues in mind at the same time.

To achieve this, a Firefox add-on called AlterEgo[5] was developed. This approach was chosen as it gives access to all of the necessary data and was most straightforward to develop. In order to respect a user's privacy as much as possible, the user is not asked for their name or any other information that would allow the plug-in to identify the user and a random unique identifier is generated at installation time. This identifier is used for all data exchange between the Firefox add-on and the server recording the data. Note that this data is collected by the server to allow for research on search personalization. Once

[5] The source code for the AlterEgo Firefox add-on has been open sourced and is available for download from http://github.com/nicolaasmatthijs/AlterEgo

the add-on is made available for mass consumption, the entire personalization process can be moved to the client side without any implications for the algorithms, avoiding some of the privacy concerns that arise in the server based approach.

Every time a user leaves a web page during browsing, the add-on will record the current user's unique identifier, the URL of the page, the time that was spent on the page, the current date and time and the length of the source HTML. All of this data is sent to a RESTful PHP web service on the server using an Ajax request, which adds the record to a queue of items to process. However, when the add-on detects a secure URL starting with https://, no data is sent to the server at all.

The server attempts to fetch the source HTML of all pages in the queue. The reason for doing this on the server side is to ensure that no private data is transmitted and only publicly available data is saved. Once the source HTML is received, the server compares its length to the length of the HTML received from the Firefox add-on. If the length difference is smaller than 50 characters (determined through trial and error), the HTML is accepted and is saved in a compressed and encrypted way alongside the unique identifier, URL, duration and date and time into the database. If the difference is more than 50 characters, it is assumed the content probably came from a password protected but non-secure site (e.g. FaceBook, Hotmail etc.). In that case, the record is discarded and not added to the user's browsing history.

Metric	Total
Participants	50
Data Capturing Period (Days)	92

Table 1: Participant Statistics

In order for people to install this add-on, a website[6] explaining the purpose and consequences to potential users was built and disseminated through various e-mail lists causing a total of 50 people to install the add-on. Whilst it is expected that most of these participants are employed in the IT industry due to the recruitment process and are thus very proficient in web search, a number of people outside of the IT industry without significant web search experience participated as well.

Metric	Total	Min	Max	Mean
Page Visits	530,334	51	53,459	10,607
Unique Page Visits	218,228	36	26,756	4,365
Google Searches	39,838	0	4,203	797
Bing Searches	186	0	53	4
Yahoo Searches	87	0	29	2
Wikipedia Pages	1,728	0	235	5

Table 2: Captured Data Statistics

The add-on captured data for three months during March to May 2010, in which it saved 6.8 GB of compressed HTML for the 50 participating users. As shown in Table 2, a total of 530,334 page visits or an average of 10,607 page visits per user was gathered, which boils down to an average of 171 page visits per user per day. The results also show that 218,228 unique page visits were saved, indicating that 58% of internet traffic are visits to previously visited web pages. The add-on

[6] http://alterego.caret.cam.ac.uk

also recorded 39,838 Google[7] search queries, compared to 186 Bing[8] searches and 87 Yahoo![9] searches. This indicates that this project's user audience is very much biased towards Google as their search engine; hence Google is used as the base search engine in the experiments described in section 6 and 7. An average user issued 797 queries over the three months, indicating that 7.5% of all web traffic is search related.

4.1.2 Data Capture

The user's recorded browsing history was not used directly for personalization, but was first processed into a more meaningful set of different kinds of data types and formats. One of the goals of this thesis is to investigate how best to build a user profile, and what elements of browsed web pages are most useful for search personalization. Therefore, an XML schema, as shown in Appendix A, that tries to incorporate as much of the characteristics and structure of web pages as possible is developed.

Full Text Unigrams

Every page in a user's browsing history will generate a record within their personal XML file. Each record consists of the web page's unique identifier within the database, the URL of the page, the date and time the page was visited and the amount of time spent on this webpage. The complete body of text that was present on the page and stripped from redundant whitespace was added into a tag as well.

HTML Extraction

Some more specific data is also extracted out of the HTML structure. The page title is extracted and stored into its own tag. The same is done for the metadata description tag. The metadata keywords, split on the ',' delimiter, are inserted as well. For each image present on the page, a tag containing its alt value (text displayed if the image can not be found and thus a reasonable attempt at describing the image) and its title value (tooltip) is added. Every anchor tag receives a separate tag in the final XML as well, containing the URL it is linking out to and its title value. Every header (h1, ..., h6) and textual (paragraph and span) tag also get their own entry in the XML. Lists and their accompanying list items and tables and the content and type (header or regular) of all table cells are incorporated as well.

Term Extraction

Next to getting all of this information directly out of the HTML structure, it was also anticipated that applying some more advanced NLP techniques to this source data might prove to be beneficial as well. Keywords describing a web page are thought to be very useful and helpful in describing the topics of a page and linking that to a user's personal interests. The metadata keywords extracted out of the HTML structure can be helpful in that respect, especially as they are supposed to be a machine readable summary of the page and might describe the page's content very accurately. However, these metadata keywords suffer from the fact that they are not obligatory in a web page and are thus not always provided. They are also used in search engine's crawling algorithms and are often abused by website authors by putting in keywords that are not related to the actual page content. Therefore, it is expected that an automatically extracted set of keywords for a web page would usefully supplement the manually provided ones.

---

[7] http://www.google.com

[8] http://www.bing.com

[9] http://www.yahoo.com

To achieve this, a Term Extraction algorithm based on Unithood And Termhood Unification, as presented in [34], is re-implemented and is fed the full text of each visited web page. This algorithm uses the C/NTC method, which uses a combination of linguistic and statistical information to evaluate the weight of each term. Given the OpenNLP[10] Part of Speech tagger, this method uses three linguistic patterns to extract the term candidates, as shown in Table 3.

Linguistic Pattern
Noun + Noun
(Adj|Noun) + Noun
(Adj|Noun) + ((Adj|Noun)*(NounPrep)?)(Adj|Noun)*)Noun

Table 3: Linguistic patterns used for Term Candidate extraction

The *CValue* is calculated for these terms and is based on the frequency of the term and its subterms, as shown in equation (1). *f(a)* represents the frequency of term *a* with |*a*| words. T_a is the set of extracted candidate terms that contain *a* and $P(T_a)$ is the total number of longer candidate terms that contain *a*.

$$CValue(a) = log_2(|a|) \times (f(a) - \frac{1}{P(T_a)} \sum_{b \in T_a} f(b)) \quad (1)$$

The *NTCValue* combines the context information of a term together with the *CValue*, as shown in equation 3. The weight of a context word[11] *b*, as shown in equation 2, is defined by the number of terms *t(b)* in which it appears over the total number of terms considered, *n*. C_a is the set of distinct context words and $f_a(b)$ is the frequency of *b* as context word of *a.* Equation 4 indicates that the *NTCValue* is mainly weighted by the *CValue.*

$$weight(b) = \frac{t(b)}{n} \quad (2)$$

$$NValue = \sum_{b \in C_a} f_a(b) \times weight(b) \quad (3)$$

$$NTCValue(a) = 0.8 \times CValue(a) + 0.2 \times NValue(a) \quad (4)$$

This method is supplemented with a term re-extraction process that utilizes the term weights using dynamic programming (the Viterbi algorithm) to augment Term Extraction. This algorithm is outlined in Algorithm 1, where $T_{i,j}$ is the word chain formed by the words from *i* to *j* of the term $T=w_1,w_2,...w_n$. *MaxNTC(1,i)* is the maximum *NTCValue* value from 1 to *i* of the term T and $NTC(T_{1,i})$ is the NTCValue of $T_{i,j}$. An exmaple of the outcome for this algorithm run on 2 web pages can be seen in Table 3.

Despite the noisy nature of web pages, some quick testing has shown that the extracted terms are in general fairly descriptive for the web page they are generated from, although common noisy words like "javascript" and "advertisement" are often present in the generated list. These extracted terms are added to the XML file as well.

[10] http://opennlp.sourceforge.net/

[11] A context of 3 is used in this implementation

AlterEgo	Mallorca
add-ons	majorca
addons	island
Nicolaas	palma
Matthijs	islands
CSTIT	spanish
Nicolaas Matthijs	balearic
Cambridge	mallorca
Language Processing	cathedral
Google	Palma de Mallorca
keyword extraction	port

Table 3: Examples of extracted terms from the AlterEgo website and the Wikipedia page about Mallorca

Algorithm: Term re-extraction for a document

Input: L ← global term list with $NTCValue$
T ← input for TREM T $= w_1 w_2 ... w_n$

1: **For** $i = 2 \rightarrow n$
2: **If** $(T_{1,i} = w_1 ... w_i) \in L$
3: $MaxNTC(1,i) = NTC(T_{1,i})$
4: **Else** $MaxNTC(1,i) = 0$
5: **End If**
6: **For** $j = 1 \rightarrow i-1$
7: **If** $(T_{j+1,i} = w_{j+1} ... w_i) \in L$
8: $MaxNTC(1,i) = \max \{MaxNTC(1,j) + NTC(T_{j+1,i}); MaxNTC(1,i)\}$
9: **End If**
10: **End For**
11: **End For**

Output: Updated term list for a document

Algorithm 1: Term Re-extraction algorithm

Noun Phrase Extraction

It is also thought that extracting nouns and noun phrases could be helpful in generating a user's profile, as nouns that are repeated often across a user's browsing history might be related to that user's interests and be beneficial in modeling the user profile and browsing behavior. A possible way to achieve this is to use a dictionary on the source text and filter out all of the words that are not recognized as a noun. Although quite an easy approach in terms of implementation, this might suffer from proper nouns not being in the dictionary. Another possible issue is a word with multiple part-of-speech possibilities being allowed in even though it's not being used as a noun in the current context. It also does not succeed in extracting noun phrases from the text, which might be useful. For example "web development" is more accurate in describing a person's interests than "web" and "development". Extracting noun phrases was achieved by taking all of the text available on a web page and splitting it into sentences using a sentence splitter from the OpenNLP Tools. The OpenNLP tokenization

script was run on all of these sentences. The sentences coming out of this were fed to the Clark & Curran Statistical Natural Language Parser[12] [4], which assigns a constituent tree to the sentence and Part of Speech tags to all of the words within the sentence. When this failed for a particular sentence, the sentence was omitted. Noun phrases were then extracted from this constituent tree and were added to the XML structure.

Over a month worth of browsing history had already been collected when it was parsed for the first time. Therefore, it was necessary to set up a cluster of 10 computers to speed up the processing. In order to do this, a Java application was written which divided the text files to be parsed into 1MB blocks. Every connected computer was fed one of these blocks, which was then parsed on that machine. After this had finished, the result was stored on the main server and the next block of text was parsed.

All of the above approaches end up generating a large XML file per user containing raw data, data extracted from the HTML structure and processed data. On average, a user's XML file is about 714 Megabyte in size. These files will be used as input to the user profile generation algorithms described in section 4.1.3.

4.1.3 Term List Generation

To generate the list of terms associated with a user, a wide range of options in terms of input data sources, filtering and weighting can be utilized. A user interface, shown in Figure 2, that allows for testing and configuring all of these different options was developed.

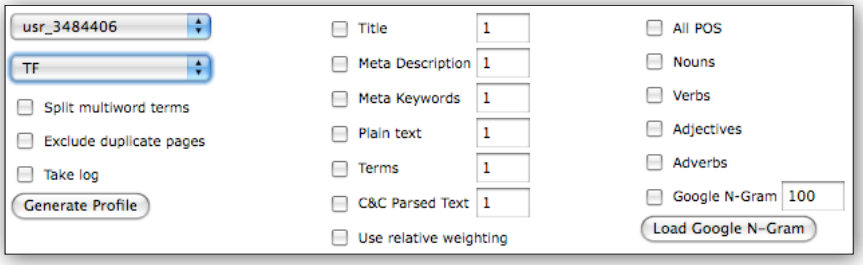


Figure 2: User Interface allowing Configurable User Profile Generation

In the initial implementation suggested in this thesis, a total of 6 different sources of input data can be used. As the user XML file gives access to more than that, this can be extended in future research. The first possible source of input data are the unigrams associated with the the title of the pages in a user's browsing history. These titles are split on whitespace, stripped of punctuation, and added to the list of terms for the user profile. The same can be done for all unigrams associated with the pages' metadata description and all unigrams associated with the text available in the user's browsing history. Metadata keywords, extracted terms and noun phrases can be utilized as well. These can be unigrams or higher order N-grams (bigram, trigram, etc.) and can be used as such in the profile. The user interface also has an option called "Split multiword terms" which will split these higher order N-grams on whitespace and transforms them into a set of unigrams. A combination of all of these different possible sources of input data can be configured and used to generate the list of terms associated with the user.

4.1.4 Term List Filtering

The list of terms accumulated through a configuration of different input sources has the potential of containing non-sensical or non-useful terms, like stop words or determiners, which may be unhelpful or even harmful in the later stage of result re-

[12] http://svn.ask.it.usyd.edu.au/trac/candc/wiki

ranking. This is especially true if the full text as present on all of the web pages is used as a source of input data. Therefore, the user interface allows for filtering the list of gathered terms. If no filtering is required, the “All POS” checkbox option should be checked.

Filtering is offered through 2 different approaches. The first one is dictionary filtering of the list of terms based on the lexical database WordNet 3.0[13] and the Java WordNet Library API[14]. The user interface allows for specifying whether only recognized nouns, verbs, adjectives and adverbs or a combination of these should be kept. Terms that are not recognized as one of the selected Part of Speech tags are filtered out of the list of terms associated with the user. The advantage of this approach is that nearly all of the non-sensical or non-helpful terms present in the list of terms will be filtered out. However, WordNet is far from complete and also filters out relevant terms like proper and common nouns that are appropriate but are not present in WordNet. Thus WordNet filtering has the potential of being too aggressive.

An alternative way in which terms can be filtered is based on how often a term occurs on the internet. The idea is that words that occur less than a certain number of times on the web can be filtered out. This is implemented by using the Google N-Gram corpus[15] to estimate the number of occurrences for each word and filter out the terms with a number of occurrences that is lower than some threshold specified in the user interface. This approach is in general successful for filtering out non-sensical words and still allows less common nouns like proper nouns in. However, as Part of Speech tags are not a part of the Google N-Gram corpus, it will be unsuccessful in filtering out stop words and determiners as they are very common on the internet and thus Google N-Gram filtering has the potential of not being aggressive enough.

4.1.5 Term Weighting

After the list of terms has been accumulated and potentially filtered through one of the proposed mechanisms, three ways are suggested in which weights can be calculated for each of the terms.

TF Weighting

The most straightforward implementation considered is called Term Frequency. A frequency vector $\vec{F}$ that contains the frequency counts of a given term t_i for all of the input data sources is defined, as shown in equation (5). For example, f_{title} is the number of times a given term t_i occurs in all of the titles in the user's browsing history. A term weight is calculated based on the dot product of these frequencies with the input data weight vector $\vec{\alpha}$ as shown in equation (6).

$$\vec{F}_{t_i} = \begin{bmatrix} f_{title_{t_i}} \\ f_{mdesc_{t_i}} \\ f_{text_{t_i}} \\ f_{mkeyw_{t_i}} \\ f_{terms_{t_i}} \\ f_{nphrases_{t_i}} \end{bmatrix} \quad (5)$$

$$w_{TF}(t_i) = \vec{F}_{t_i} \cdot \vec{\alpha} \quad (6)$$

[13] http://wordnet.princeton.edu/

[14] http://sourceforge.net/projects/jwordnet/

[15] http://googleresearch.blogspot.com/2006/08/all-our-n-gram-are-belong-to-you.html

Terms are given a default weight of 1 for each of the input sources, but the user interface allows for specifying a separate weighting factor in the input data weight vector in order to give a higher weight to certain input sources and thus make them relatively more important. For example, it might make sense to give a keyword describing a page a higher weight than all of the words available in the full page text. Not making use of one or more input data sources is equivalent to changing its weighting factor to 0. In the experiments conducted for this thesis, it was chosen to give each of the input data sources a binary value (0 or 1).

On top of weighting that is dependent on the type of input data source being used, it is also possible to check “Use relative weighting” in the User Interface. This means that the frequency counts within the frequency vector $\vec{F}$ will be normalized by the total number of terms available in a given data source across the complete browsing history *(N)*, as shown in equation (7).

$$\vec{F}_{rel_{t_i}} = \begin{bmatrix} \frac{f_{title_{t_i}}}{N_{title}} \\ \frac{f_{mdesc_{t_i}}}{N_{mdesc}} \\ \frac{f_{text_{t_i}}}{N_{text}} \\ \frac{f_{mkeyw_{t_i}}}{N_{mkeyw}} \\ \frac{f_{terms_{t_i}}}{N_{terms}} \\ \frac{f_{nphrases_{t_i}}}{N_{nphrases}} \end{bmatrix} \qquad (7)$$

$$w_{TF_{rel}}(t_i) = N_{total} \times (\vec{F}_{rel_{t_i}}.\vec{\alpha}) \qquad (8)$$

In this case, metadata keywords will automatically receive a higher weight than full text terms, as there are less of them within the data set, without having to manually tune the input data source weighting factor. The final weight is then multiplied by the total number of terms available in all of the data sources *(N_{total})*, as shown in equation (8).

TF-IDF Weighting

The second possibility for calculating term weights is called TF-IDF or Term Frequency - Inverse Document Frequency. In this scenario, the term's Term Frequency weight is normalized by its Document Frequency, representing the number of documents in which the current term appears, as shown in equation (9).

$$w_{TFIDF}(t_i) = \frac{1}{log(DF_{t_i})} \times w_{TF}(t_i) \qquad (9)$$

It is however questionable whether the browsing history should be used as the data set to calculate the Document Frequency. It is perfectly possible that the term "NLP" occurs in a lot of the documents within one's browsing history, despite it being quite a rare word, and it is still desirable for the term to have a high weight in the profile as it is descriptive of the user's interests and will prove to be helpful in re-ranking search results. Therefore, the Inverse Document Frequency of a given term for all pages on the web (Inverse Internet Frequency) is used. The Google N-Gram corpus is utilized to estimate this number. Due to the large dynamic range of term frequencies across the web, the log of the Document Frequency is taken. Using MemCached and a Redis Server to store the Google N-Gram corpus in memory, these numbers can be retrieved very efficiently. Again, it is possible to use relative weights for each of the different input sources as shown in equation (10).

$$w_{TFIDF_{rel}}(t_i) = \frac{N_{total}}{log(DF_{t_i})} \times w_{TF_{rel}}(t_i) \quad (10)$$

BM25 Weighting

The final weight calculation method is taken from Teevan et al [31] in which web search personalization is explored through modifying the probabilistic weighting scheme BM25. As they have no access to explicit relevance judgements, they also assume that all of the data related to the user is relevant. In their extension of BM25 weighting, they had to incorporate the fact that the corpus includes an unknown amount of outside documents.They use the equation presented in (11) to calculate the weight for a given term.

$$w_{BM25}(t_i) = log\frac{(r_{t_i} + 0.5)(N - n_{t_i} + 0.5)}{(n_{t_i} + 0.5)(R - r_{t_i} + 0.5)} \quad (11)$$

As the domain of the algorithm is Web search, the corpus is the Web. The parameter *N* represents the number of documents on the web. To obtain an estimate for this, the frequency of the most frequent word in English according to the Google N-Gram corpus is used (220,680,773). n_{ti} represents the number of documents in the corpus that contain the term i, which is estimated by using the Google N-Gram corpus as well. *R* is the number of documents in the user's browsing history and r_{ti} is the number of documents in the browsing history that contains term i within the selected input data sources. Implementing this method allows us to compare the obtained results to the Teevan method and assess wether their method is more effective when different input data sources and filtering methods are utilized. Note that the suggested method has no access to users' full Desktop index and are limited to their browsing history, making this implementation simpler but potentially less effective than Teevan's.

4.1.6 Additional Options

The User Interface also presents some additional options. Every page in a user's browsing history will contribute to a term's weight if the term is present in any of the chosen data sources. When the same web page has been visited many times, every visit will contribute towards the weight of the terms mentioned on that page. If “Exclude duplicate pages” is checked, every unique web page is only considered once. Uniqueness is determined by looking at the page's URL. This potentially avoids giving too high a weight to certain terms on websites the user visits often of which the content is not necessarily completely relevant, like online newspapers. The option "Take Log" will take the log value of all of the obtained term weights before saving the profile. This can be used as a way to normalize the term weights, especially when Term Frequency is chosen as the weighting scheme.

The User Interface shown in Figure 2 will generate a user profile, consisting of a list of terms and associated weights describing a user's interests, a list of visited URLs and the number of visits and a list of previous search queries and click-through data, for the selected user with the selected set of configuration parameters. This is all highly configurable and allows for easy exploration of the different possible combinations. After the user profile has been generated, it is printed onto the screen, which can help in understanding which configurations produce sensible results. The generated user profile is written to a file and is used for re-ranking search results coming out of a search engine in section 4.2.

4.2 Re-ranking Strategies

Like previous work, the user profile is used to re-rank the top results returned by a search engine in order to bring up results that are more relevant to the user. This allows us to take advantage of the data search engines use to come up with their initial ranking, like Universal PageRank, to which no access can be gained, to first obtain a very limited set of results which

are then personalized. According to [31], chances are high that even for an ambiguous query the search engine will be quite successful in returning pages for the different meanings of an ambiguous search query. In this thesis, it is opted to retrieve and re-rank the first 50 results retrieved for a query via the Google Rest API and aim to move more relevant results to the top of the ranking and push down less relevant results. Each of these 50 search results are given a weight and are re-ranked accordingly. The received weight reflects the chance of the result being relevant to the user given his profile and interests. Whilst the user profile is global across all queries, result re-ranking is done on a query per query basis. In this work, Google is used as the base search engine as the users that have installed the AlterEgo add-on are heavily biased towards using Google for their web searching needs. A User Interface, as shown in Figure 3, is developed which allows for testing and experimenting with the different re-ranking methods suggested.

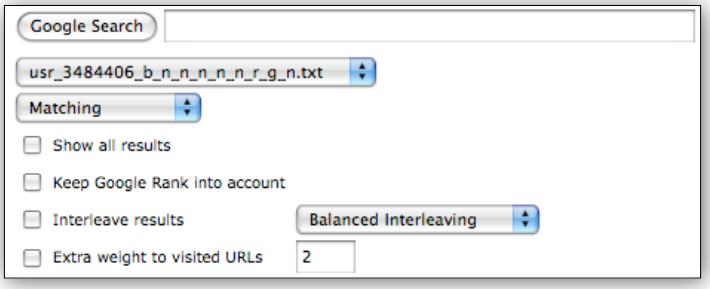


Figure 3: User Interface allowing Configurable Search Re-ranking

4.2.1 Weighting Methods

Weights can either be assigned to the full web documents returned by the search engine or to the search snippets associated with those documents displayed on the search page. The focus of this thesis lies on assigning weights to the search snippets, as it was found to be more effective for re-ranking search results by Teevan et al [31]. Search snippets tend to be query biased, are keyword focussed and are less noisy than the actual web pages they are referring to. On top of that, using search snippets also allows for a straightforward and realistic client-side implementation of search personalization. In this thesis, four different methods and additional options in which the weight for a search snippet can be calculated are presented and implemented.

Matching

For each word in the search snippet's title and summary that is also in the user's list of profile terms, the weight associated with that term will be added to the snippet's weight, as shown in equation (12). Words present in the snippet's title or summary but not in the user's profile will not contribute towards the snippet's final weight. N_{si} represents the total number of unique words within the snippet's title and summary, and f_{ti} represents the number of occurrences of t_i within the snippet. This method is equivalent to taking the dot product between the user profile vector and the snippet vector.

$$score_M(s_i) = \sum_{z=0}^{N_{s_i}} f_{t_i} \times w(t_z) \qquad (12)$$

Unique Matching

A second search snippet weighting option (called Unique Matching) is very similar to Matching. The difference is that if a certain word occurs more than once in the search snippet title or summary, it only contributes to the snippet's weight once as shown in equation (13).

$$score_{UM}(s_i) = \sum_{z=0}^{N_{s_i}} w(t_z) \qquad (13)$$

A motivating example for this approach is the query "cambridge pub" for a user living in the UK. One search snippet about a pub in Cambridge, MA might mention "cambridge" 3 times and the word "pub" once. A different search snippet about a pub in Cambridge, UK might mention the words "cambridge", "pub" and "uk" once. Clearly, the second result is more relevant to the current user; however snippet 1 will be weighted heavier when using the Matching approach. Unique Matching will only assign the weight of the term "cambridge" within the user profile once to the search snippet.

Language Model

The third weight calculation method attempts to generate a unigram Language Model from the user profile in which the weights associated to the terms are used as the frequency counts for the Language Model generation, as shown in equation (14). *N* represents the total number of words in the snippet's title and summary, and w_{total} stands for the sum of all the weights within the user profile. The Language Model calculates the probability of a snippet given a user's profile. As the profile describes the user's interests in keywords, this probability is a good indication of how well the snippet models the user. In order to avoid a zero probability for snippets that contain a word that is not part of the user profile, Add1 smoothing is applied to the Language Model, adding 1 to the frequency count of each term. This will end up producing a deficient LM assigning quite a large amount of probability mass to the unseen words.

$$\begin{aligned} score_{LM}(s_i) &= log(\prod_{z=0}^{N_{s_i}} \frac{w(t_z)+1}{w_{total}}) \\ &= \sum_{z=0}^{N_{s_i}} log(\frac{w(t_z)+1}{w_{total}}) \end{aligned} \qquad (14)$$

PClick

The final snippet weighting method is an implementation of the PClick or “Person-level Re-ranking Based on Historical Clicks” method as presented in [8]. It assumes that for a query *q* submitted by a user *u*, the web pages frequently clicked by *u* in the past are more relevant to *u* than those seldom clicked by *u*. The personalized score for a snippet is computed by equation (15). |*Clicks(q,p,u)*| is the number of clicks on web page *p* by user *u* on query *q* in the past, |*Clicks(q,.,u)*| is the total click number on query *q* by *u*, and β is a smoothing factor set to 0.5.

$$score_{PC}(s_i) = \frac{|Clicks(q,p,u)|}{|Clicks(q,\bullet,u)|+\beta} \qquad (15)$$

PClick makes no use of the terms and weights associated to the user's profile and is solely based on click-through data for a given query. A disadvantage of this method is that it can only be applied to repeated queries.

4.2.2 Additional Options

The re-ranking User Interface also offers a number of additional options that can be utilized to enhance personalization. First, one can opt to give additional weight to URLs that have already been visited, making use of the assumption that the

user has visited that page before because it was relevant. This is different to PClick in that it boosts all URLs that have previously been visited, whilst PClick will only boost URLs that have directly been clicked for the current search query. Chances of pushing up a page that is relevant for the user, but not relevant for the user's current information need, are small, as the results returned for the query are already very query focussed. A factor *v* can be set in the User Interface and the snippet weight will be multiplied by this factor and the number of previous visits to that web page (*n*) using equation (16). Thus pages visited frequently are likely to be moved to the top of the ranking and a combination of user profile and URL history based personalization is accomplished.

$$finalscore(s_i) = (score(s_i) + 1) \times v \times n \quad (16)$$

In the re-ranking framework discussed so far, the original ranking is not taken into account. The Google rank can be incorporated into the final snippet weight by normalizing the snippet weight by the log value of the snippet's original rank r_{si} as shown in equation (17).

$$finalscore(s_i) = score(s_i) \times \frac{1}{log(r_{s_i})} \quad (17)$$

Intuitively, it makes sense to consider the original ranking of documents and this approach causes re-ranking to happen less frequently. In practice, the first couple of results will become less sensitive to re-ranking, unless the calculated weight for a snippet is very high compared to the other snippet weights due to remarkable similarities with the user profile or due to encountering a previously visited web page. This has the potential of still obtaining a reasonably good ranking for non-ambiguous or non-personalizable queries by putting more confidence in the default search engine ranking, taking the need away to differentiate between personalizable and non-personalizable search queries.

4.2.3 Re-ranking User Interface

The User Interface displayed in Figure 3 makes it possible to submit any search query to Google and select a user profile to use for result re-ranking. As shown in Figure 4, the original Google ranking is displayed side by side with the re-ranked list, displayed both as actual common searches with results embedded in the Google layout. If the option "Show all results" is checked, a full list with all 50 results will be shown on the screen instead of the default paged version.

This personalization framework makes it possible to generate a user profile using various parameters and re-rank and display the search results for a given query using various parameters in a single screen, allowing for easy exploration of different parameter configurations.

The User Interface also offers the possibility of interleaving the re-ranked search results with the original results using either the Balanced or Team Draft interleaving algorithms as suggested in [20]. This will be utilized in the evaluation experiments presented in section 7.

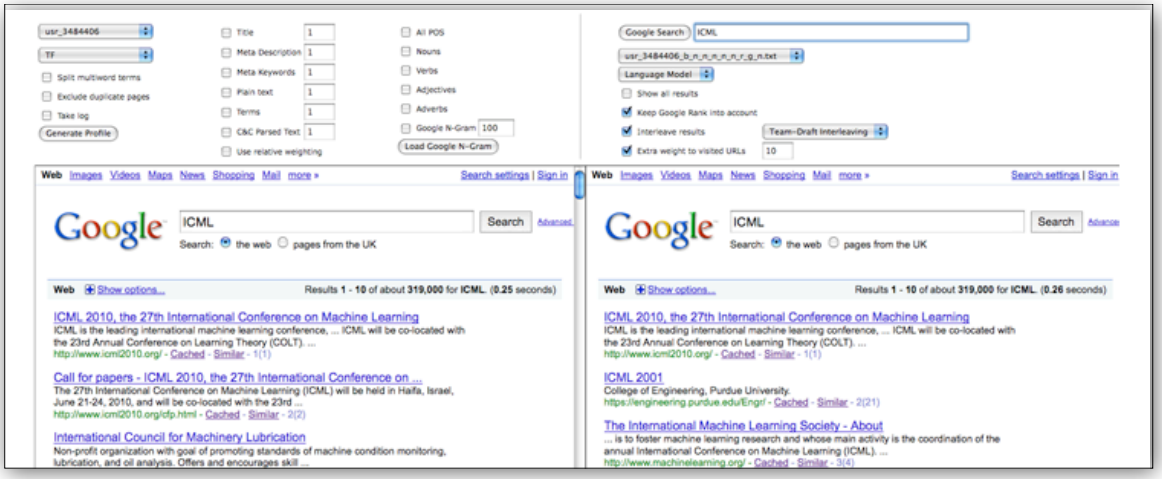


Figure 4: Full User Profile and Search Re-Ranking User Interface

5. Evaluation Approach

Comparing the quality of two different rankings has recently turned into a very active research area on its own and thus it is not surprising that evaluating personalized search is quite a difficult problem that often suffers from the lack of thorough or comparable evaluation methods. This thesis aims at using a reliable evaluation method which assesses whether personalized search makes an actual difference in a user's day to day web search activities.

Of the many evaluation methods that have been used in the past, four, of which two are offline and two are online evaluation strategies, are considered in detail.

Relevance judgements

The first offline evaluation approach, used by for example Teevan et al [31], is based on assembling a group of people that judge the relevance of the top *k* documents or search snippets for a set of queries. Given these relevance judgements, an (N)DCG or (Normalized) Discounted Cumulative Gain score can be calculated for a given query and ranking, which reflects the quality of the presented ranking for that user. This approach has the advantage that once the relevance judgements are made, it allows for testing a large number of different user profile and re-ranking parameter configurations. However, due to the large amount of time it takes to judge *k* documents on relevance, this can only be done for a small number of search queries. As volunteers need to be found to sit through this slow and tedious evaluation process, it is also hard to gather a large group of evaluators. The evaluation process also does not reflect a user's normal browsing and searching behavior, which might influence the final results. Moreover, this approach assumes that (N)DCG is the right way to combine a set of relevance judgements into a rank quality score.

Side-by-side evaluation

The next offline evaluation method, as previously used by for example [33], consists of presenting users with two alternative rankings side-by-side and ask which they consider to be best. The advantage of this method is that an actual judgement is made of which ranking is the best one. However, users evaluate the entire presented ranking whilst in real life situations they might only look at the first couple of results which might bias the results. Judging two rankings next to each other is considerably faster than judging *k* documents per query, but it still requires a long offline evaluation exercise. In addition, an evaluator has to provide a new assessment for each distinct ordering of documents that is investigated. This makes it hard to scale to large numbers of parameters as they lead to a large number of different rankings. It also does not succeed in reflecting a user's normal behavior which might thus influence the results.

Clickthrough-based evaluation

One online evaluation approach involves looking at the query logs and click-through data from a search engine on large scale (e.g. used by [8]). The logs record which search result was clicked for a given query, which allows the evaluation framework to check where the clicked result would be positioned in the personalized version of the search results. This allows for testing a large number of parameter configurations and also does not require any user effort as their actions are recorded as they go, reflecting their natural environment and browsing behavior. However, the method can have difficulties in assessing whether a search personalization strategy actually works. Users are more likely to click a search result presented at a high rank, although these are not necessarily most or more relevant [15]. It is also unsuccessful in assessing whether the results that have been brought up from a page lower down would have been relevant. On top of that, this project had no access to such large scale usage and user profile data for this experiment.

Interleaved evaluation

The final online evaluation option, which has not been used before for the evaluation of personalized search, is interleaved evaluation [20, 13]. Interleaved evaluation combines the results of two search rankings by alternating between results from the two rankings while omitting duplicates, and the user is presented with the produced interleaved ranking. The users' clicks indicate a relative preference comparing the quality of two retrieval functions. The ranking that contributed the most clicks over a large number of queries and a large number of users is considered to be better. Radlinski et al. [20] show that this interleaving approach is much more sensitive to changes in ranking quality than other click-based metrics. It has also been shown to correlate highly with offline evaluations with large numbers of queries [22]. In addition, this method does not require any time from the user as they provide an evaluation over time as they pursue their real life information needs, reflecting their normal browsing behavior. However, one evaluation only provides an assessment for 1 set of parameters and thus an explicit evaluation is required for each parameter configuration being investigated.

1: **Input**: Rankings $A = (a_1, a_2, \dots)$ and $B = (b_1, b_2, \dots)$
2: **Init**: $I \leftarrow (); TeamA \leftarrow \emptyset; TeamB \leftarrow \emptyset;$
3: **while** $(\exists i : A[i] \notin I) \wedge (\exists j : B[j] \notin I)$ **do**
4: **if** $(|TeamA| < |TeamB|) \vee$
$((|TeamA| = |TeamB|) \wedge (RandBit() = 1))$ **then**
5: $k \leftarrow \min_i\{i : A[i] \notin I\}$ ... *top result in A not yet in I*
6: $I \leftarrow I + A[k]$; ..... *append it to I*
7: $TeamA \leftarrow TeamA \cup \{A[k]\}$ ..... *clicks credited to A*
8: **else**
9: $k \leftarrow \min_i\{i : B[i] \notin I\}$ ... *top result in B not yet in I*
10: $I \leftarrow I + B[k]$ ..... *append it to I*
11: $TeamB \leftarrow TeamB \cup \{B[k]\}$ .... *clicks credited to B*
12: **end if**
13: **end while**
14: **Output**: Interleaved ranking I, $TeamA$, $TeamB$

Algorithm 2: Team-Draft Interleaving

In the conducted experiments, the Team-Draft interleaving algorithm, as described in Algorithm 2, is used to interleave the original Google ranking and the re-ranked personalized version. Let A and B be retrieval functions. Given the results for a query *q*, *A(q) = (a1, a2, ..., a_n)* and *B(q) = (b1, b2, ..., b_m)*, Team-Draft interleaving combines these results into a single ranking. This algorithm is motivated by how sports teams are often assigned in friendly games. Given a pool of available players (ranking A and B), two captains (one for Team A and one for Team B) take turns picking their next favorite player in the set of remaining players, subject to a coin toss every turn deciding which team captain gets to pick first. This approach treats *A(q)* and *B(q)* as the team captains' preference orders. An example ranking produced by Team-Draft interleaving, along with team assignments, is shown in Table 4. More details about this approach can be found in [20]. The combined

ranking is presented to the user and clicks on any of the results are recorded. If results from the personalized ranking end up being clicked more often, it is a strong indicator that the personalization is successful.

RandBit	Presented Ranking	Ranking A	Ranking B
1	(A) Ajaxian	(@1) Ajaxian	(@1) Ajaxian
N/A	(B) Ajax Programming - Wikipedia	(@2) Ajax Programming - Wikipedia	(@2) Ajax Programming - Wikipedia
1	(A) GWT: Google Code	(@3) GWT: Google Code	(@3) Ajax Tutorial
N/A	(B) Ajax Tutorial	(@4) Ajax Tutorial	(@4) Ajax - MDC
0	(B) Ajax - MDC	(@5) Ajax.org - RT Collaboration	(@5) GWT: Google Code
N/A	(A) Ajax.org - RT Collaboration	(@6) Ajax - MDC	(@6) Ajax.org - RT Collaboration

Table 4: Example of Team-Draft Interleaving for the query “ajax”

This last option seems like the evaluation method that is most reliable and most reflective of real user experience. Therefore, this project wanted to use this for evaluation. Quite a number of people would then be able to evaluate the different personalization techniques in real life. It is also the hardest evaluation technique for showing improvements as, opposed to NDCG based evaluation, bringing a slightly relevant page up from for example rank 8 to rank 5 will not help if the most relevant page is at rank 1. This effect is even stronger when few re-ranking is done at high ranks.

However, the user profile generation and re-ranking steps both have a large number of possible parameters and it is unfeasible to do an online evaluation for all of them. Hence, to pick the sets of parameter configurations to properly evaluate using the interleaved evaluation method, offline evaluation is used. In the first phase, a relevance judgements evaluation session is organized for a number of users that have installed the AlterEgo add-on and an average NDCG score is calculated for all possible profile and re-ranking parameter configurations. This is used to trim down the list of possible combinations to the 3 most promising ones which are then used in an online interleaved evaluation phase.

6. Offline Evaluation

This section first describes the organized offline evaluation session. Next, the investigated parameters are described and an overview of the obtained results and the most promising discovered personalization strategies is given.

6.1 Relevance Judgements

For the relevance judgements evaluation session, which was conducted on the 30th of April 2010 and can be seen in Figure 6, a total of 6 participants (5 of which are employed in the IT industry and 1 in the education industry) were recruited. At that point in time, a total of 2 months of browsing history had been recorded and stored for each of them. The process and format of the evaluation session was kept very similar to the relevance judgement session conducted by Teevan et al [31].

(a) Select *Irrelevant* if the document is not useful and not interesting to you. (b) Select *Relevant* if the document is interesting to you, but is not directly about what you were hoping to find or if the document is somewhat useful to you, meaning that it touches on what you were hoping to find (maximum 1 paragraph), but not very extensively. (c) Select *Very Relevant* if the document is useful or very interesting to you, i.e. it is what you were hoping to find.

Table 5: Relevance judgements guidelines

Each participant was asked to judge the relevance of the top 50 web pages returned by Google for 12 queries according to the criteria presented in Table 5. The documents were presented in a random order. Full web pages were evaluated instead of snippets because the user is only interested in the actual web page being relevant for his information need. An evaluation user interface, depicted in Figure 5, was developed to guide the participants through the process.

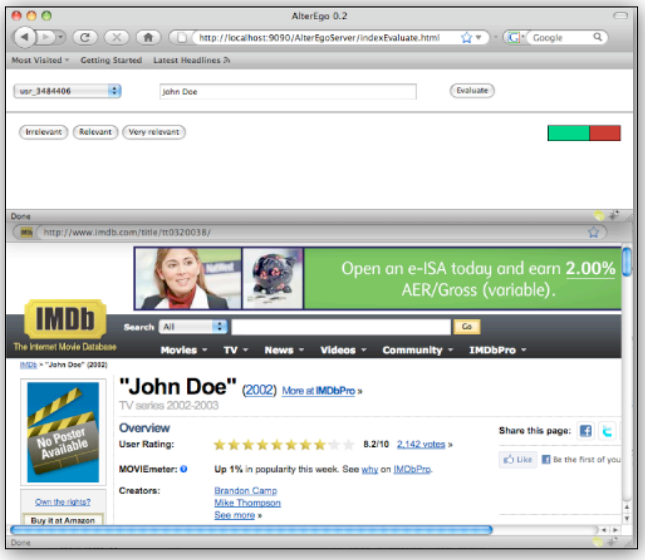


Figure 5: Relevance Judgements Evaluation Environment

Every user had to provide 50 judgements for 12 queries, adding up to a total of 600 relevance judgements per user. The first query they had to provide judgements for was their own name (Firstname Lastname), as a warm-up exercise. Next, all users were presented with a list of 25 general search queries in a random order. These queries consisted of 16 queries taken from the TREC 2009 web search track. However, as most of these are US centric, 9 self-defined UK focussed queries such as "pub", "football" and "cambridge" were added. All users were asked to select 6 queries out of this list and provide relevance judgements for each of them. Examples of some of the queries that were chosen by multiple people can be seen in Table 6. As a final exercise, each participant was presented with a list of their most recent 40 search queries from their browsing history and were asked to pick 5 for which they remembered the returned results could have been better and provide relevance judgements for all of these.

Query	Users
Cambridge	6
GPS	4
Website design hosting	3
Volvo	3

Table 6: Common queries used by different participants

On average, it took each participant about 2.5 hours to complete this exercise for 12 search queries. Interestingly, all users mentioned that they came across really interesting websites during to the exercise which they did not know existed before, indicating that there is a potential for search personalization to enrich the set of returned results. The provided relevance judgements were saved in the database and later used to calculate an average NDCG score for all possible parameter configurations from section 5, which was utilized to determine the set of most promising combinations to be evaluated in the interleaved evaluation.

Figure 6: Offline Evaluation Session

6.2 Results and Discussion

Using the judgements described above, the three most promising user profile and re-ranking personalization strategies were selected.

The relevance judgements methodology allows testing all possible parameter combinations once the judgements are given. However, a number of parameters have by trial and error been found to make little or no difference, or performed poorly. Therefore, these parameters are not evaluated in this experiment, which simplifies the procedure. Duplicate pages in a user's browsing history, based on the URL, were always included in the term and term weight generation script as visiting a web page very often is an indication of relevance, which is also the intuition behind URL boosting. Multiword terms found in the extracted noun phrases, metadata keywords, etc. were always split into a set of unigrams in order to be able to generate a full unigram user profile. Early experiments indicated no real improvements for treating them as multiword terms. Splitting them into unigrams also yields a more straightforward implementation in terms of filtering, difference in dynamic range for frequency counts of unigrams and higher order N-Grams, etc.

In summary, the following parameters were investigated for profile generation, representing the different steps shown in Figure 1: All possible combinations of the 6 different input data sources in which all of the input sources used for a given configuration could either be given a weight of 1 or a relative weight, 3 filtering methods including No Filtering, WordNet noun

filtering and Google N-Gram filtering using a threshold of 1,000, which through trial and error seemed most successful in filtering out non-sensical terms and allowing in relevant terms, and three term weighting methods being TF, TF-IDF and BM25. All possible combinations of these parameters were investigated, which adds up to a total of 1,134 profiles per user.

In terms of re-ranking the search results, the following set of parameters was investigated: 4 snippet weighting methods, being Matching, Unique Matching, Language Model and PClick, normalizing the snippet weight by the original Google rank or not and being able to either give no extra weight to previously visited URLs or multiply the obtained weight by 10 times the number of visits.

This added up to a total of 15,878 different parameter configurations for profile generation and result re-ranking being evaluated. All of these profiles were generated for all of the participating users based on 2 months of browsing history, followed by generating a re-ranked ranking for all users, all queries and all combinations whilst calculating the Normalized Discounted Cumulative Gain, as shown in equation (18), for each of these rankings. DCG is a rank quality metric that puts a strong emphasis on getting relevant results high in the ranking.

$$DCG_p = \sum_{i=1}^{p} \frac{2^{rel_i} - 1}{log_2(1+i)} \qquad (18)$$

The *rel* value is determined from the relevance judgements, in which Irrelevant is 0, Relevant is 1 and Very Relevant is 2. In these experiments, the DCG score up to rank 10 is calculated, which represents one page of search results, meaning that *p* is set to 10. As shown in equation (19), the Normalized DCG score is calculated for each of the rankings, which allows easier comparison of the quality of 2 different rankings. To calculate the NDCG score, the DCG score is divided by the ideal DCG score in which all Very Relevant documents come first, followed by all Relevant documents which on their turn are followed by all Irrelevant documents.

$$NDCG_p = \frac{DCG_p}{IDCG_p} \qquad (19)$$

In all of the following results, NDCG scores that have been averaged across all queries and all users are reported.

6.2.1 Baseline Comparisons

In order to assess how well the different suggested personalization strategies performed, the outcome of the experiment is compared with several baselines as shown in Figures 7 and 8. The reported scores are normalized DCG scores for 72 queries, and all statistical analyses were performed using a paired T-Test.

The baseline methods compared are the default Google ranking, the Teevan method and the PClick method. The results obtained for these are nicely in line with the results reported in [31] and [8], where the Teevan approach is significantly ($p < 0.05$) better than Google and PClick is significantly ($p < 0.01$) better than Google. Note that in this thesis' implementation of the Teevan algorithm, only the browsing history is used as input data, as this project has no access to the user's files or e-mails, which might disadvantage the Teevan approach. PClick also performs better than the Teevan approach, which is consistent with the results reported in [8].

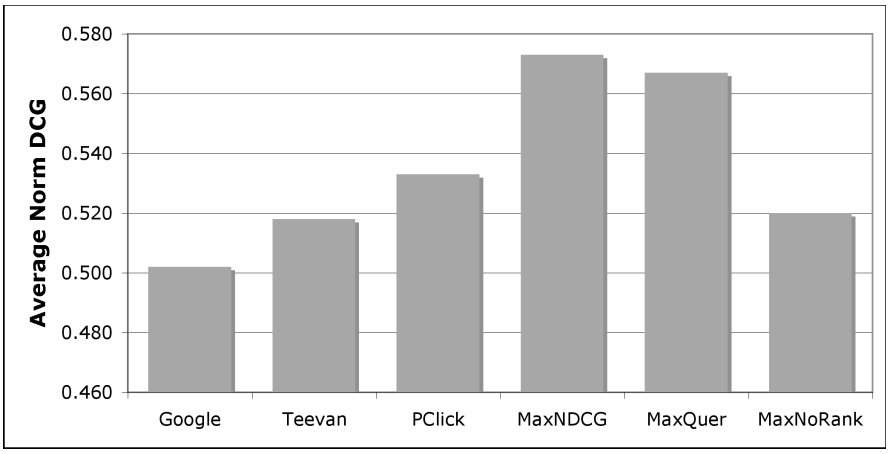


Figure 7: Average Normalized DCG scores comparing baseline system performance to best personalization strategies

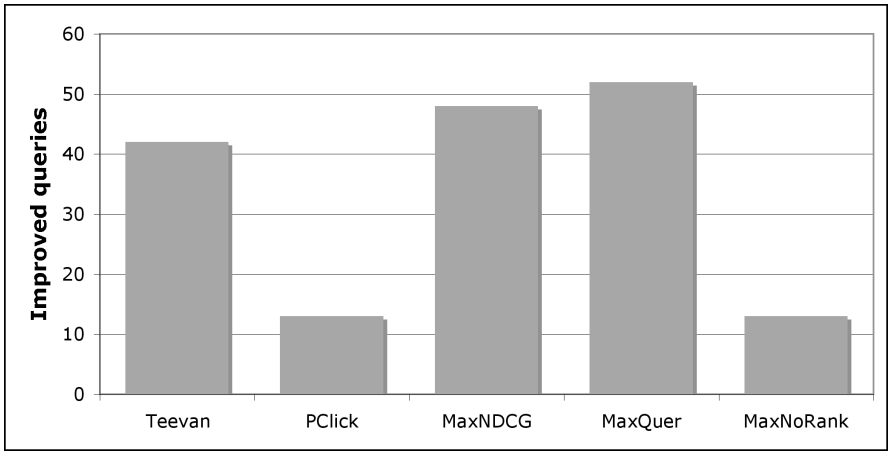


Figure 8: Comparing baseline system performance to best personalization strategies in terms of number of improved queries

As shown in Table 7, out of the 15,878 user profile and re-ranking combinations that have been investigated, a total of 4,455 combinations performed better than Google. 1,556 combinations performed consistently better than Google, meaning that the average NDCG score was higher than Google for each user, indicating that the personalization strategy worked for all six users. A total of 3,335 strategies performed better than the Teevan approach, of which 413 were consistently better. Compared to PClick, 1,580 combinations performed better of which 359 were consistently better.

Four interesting user profile and re-ranking combinations, which can be found in Table 8, were selected to further evaluate. The first selected profile parameter configuration, which yielded the highest average Normalized DCG score in the offline experiment, involves using Title, Metadata Keywords and Extracted noun phrases as input data using relative weighting, No Filtering and TF-IDF as the weighting method. This profile will be referred to as MaxNDCG in the following sections. The second selected profile was chosen because it had the highest number of improved queries and consists of using Extracted Terms and Extracted noun phrases as input data using relative weighting, No Filtering and TF as the weighting method, which will be referred to as MaxQuer. The third selected method was the best performing method that did not take the original Google ranking into account. It used Metadata keywords as input data, No Filtering and TF as the weighting method. The final selected method was obtained by a greedy learning strategy in which one parameter was selected at a time and then fixed. The outcome to this was also very similar to what is on average the combination of the best or most useful parameters. It consists of using Title, Metadata Keywords and Extracted Terms as input data, making it heavily focussed on describing the user using keywords. It deploys relative weighting, No Filtering and uses BM25 for term weight calculation and will be referred to as MaxBestPar. All of these selected profiles used Language Model snippet weighting, the Google rank was taken into account and the calculated weights were multiplied by 10 and the number of visits for pages that have been visited before by that user.

Baseline	Better	Consistently Better
Google	4,455	1,556
Teevan	3,335	413
PClick	1,580	359

Table 7: Comparing baseline system performance to best personalization strategies in terms of number of improved queries

MaxNDCG and MaxQuer are both significantly ($p < 0.01$) better than default Google, Teevan and PClick. MaxNDCG, with an average NDCG of 0.573, yields a 14.1% improvement over Google, and MaxQuer, with an average NDCG of 0.567, yields a 12.9% improvement over Google. These results are, according to previous papers, better than any results obtained so far for personalized search.

Despite MaxNoRank ignoring the Google rank, it still obtains a result (NDCG of 0.520) that is significantly ($p < 0.05$) better than Google and better than Teevan. A ranking that outperforms web ranking solely based on user data has not been achieved before.

Name	Method	Avg NDCG	Improved
MaxNDCG	TF-IDF, RTitle, RMKeyw, RCCParse, NoFilt - LM, Look At Rank, Visited	0.573	48/72
MaxQuer	TF, RTerms, RCCParse, NoFilt - LM, Look At Rank, Visited	0.567	52/72
MaxNoRank	TF, RMKeyw, NoFilt - LM, Look At Rank, Visited	0.520	13/72
MaxBestPar	BM25, RTitle, RMKeyw, RTerms, NoFilt - LM, Look At Rank, Visited	0.566	45/72

Table 8: Investigated profiles for interleaved evaluation

A different metric that can be used for comparing methods is to look at the number of queries for which the NDCG score improved due to personalization, which is shown in Figure 8. PClick only improves 13 out of 72 queries compared to the 44 improved queries for the Teevan approach, yet it still ends up with a higher NDCG score. This can be explained due to the fact that the PClick method only works on repeated queries by bringing up pages of which the user thought were relevant before. Although this will only occasionally cause personalization to make a difference, it does manage to make bigger improvements in the pages it brings up. The Teevan approach, in which the user profiles are quite noisy because web pages are being treated as normal documents, is much more aggressive in personalization, which causes it to have a negative effect on some of the queries as well. MaxNDCG achieves an improvement for 48 queries, MaxQuer improves 52 queries. Although the difference with the Teevan approach is quite small, they obtain a much higher NDCG score, which can be explained by the rich set of input data and filtering methods used for these profiles that cause the profiles to be less noisy and thus harm less on queries that are more challenging to personalize.

6.2.2 Parameter Configuration

Given all the investigated user profile and re-ranking combinations, it was assessed how all of these parameters alter the overall strategy performance. The one-way effects of all individual parameters are summarized and their influence on the average NDCG score is explored. Firstly, as shown in Figure 9, the average NDCG score is presented for all personalization techniques using a given parameter. In Figure 10, a certain parameter λ (e.g. filtering, term weighting, etc.) is considered. For every other parameter setting, it was counted how often the different possible values of λ were most helpful. This experiment was repeated for every parameter group. These approaches do not examine interaction effects, so at the end of this section an overview is given of which parameter combinations achieved the best performance.

The trends shown in Figure 10 can definitely be recognized in Figure 9, although the difference in Normalized DCG scores between different parameter values is a lot smaller than the difference in the number of times a given parameter value yielded an improvement. This indicates that using an approach that works consistently better and is a better personalization strategy, might in practice only give a small increase in NDCG. The NDCG scores shown are averaged over a large number of parameter configurations for each investigated parameter, including all personalization strategies that performed worse than Google, which might have an influence on the difference in NDCG score for different parameter values.

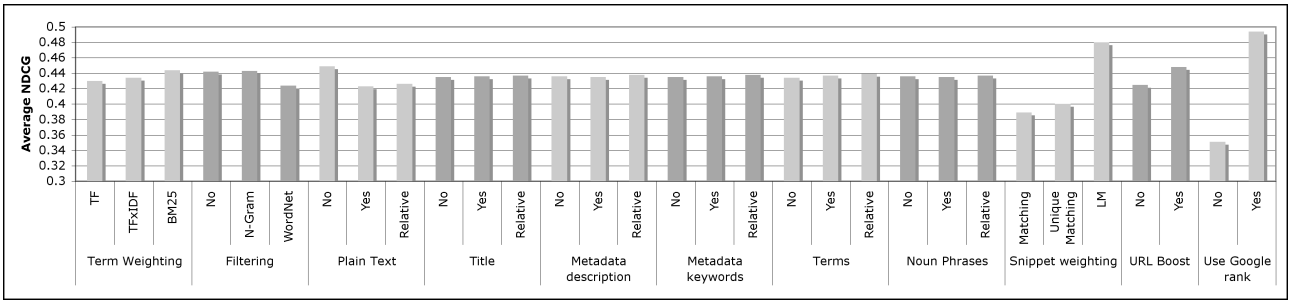


Figure 9: Average normalized DCG score for different variables

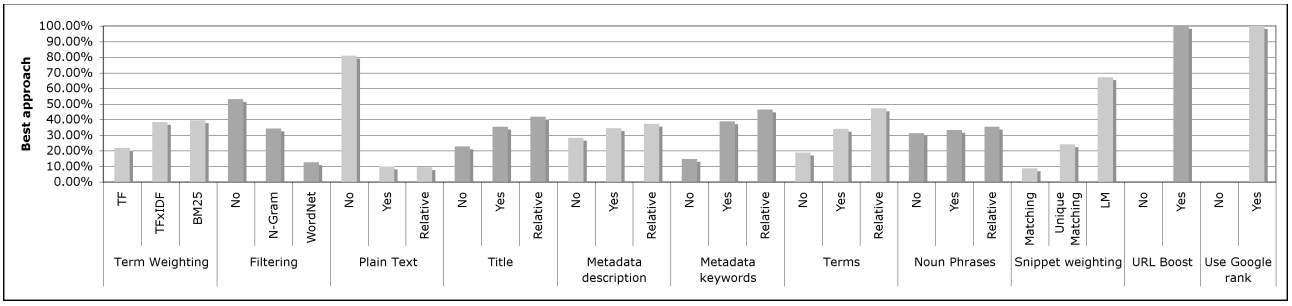


Figure 10: Percentage-wise number of times an investigated parameter performs best for a given parameter configuration

In terms of re-ranking strategies, it is very clear that there is one approach that significantly outperforms all the others, indicating that this approach should be used for all generated profiles. Using the Matching approach for snippet weight calculation is significantly worse ($p < 0.01$) than using Unique Matching. Using a Language Model based on the user profile performs significantly better ($p < 0.01$) than both Matching and Unique Matching and is in 67% of the investigated profiles the best choice. Unique Matching only occasionally performs better than a Language Model when the user profile is very much keyword focussed and does not use Extracted Nouns, Title, etc. as input data. Except for a single case, multiplying the snippet weight by 10 for URLs that have previously been visited performs significantly better ($p < 0.01$) than not keeping this into account. This proves that a combination of a user profile based and click based personalization strategy can be used successfully. Not normalizing the obtained snippet weights by their original rank always performs worse than when the Google rank is taken into account.

When looking at the input data used to generate a user's profile, it can be seen that all data sources are helpful in improving personalization performance, except when the full web page text is used. Not using the full text performs significantly better ($p < 0.01$) than when this is included. This indicates that treating web pages like a normal document or a bag of words does not work, presumably due to its noisy nature. Using metadata keywords, extracted terms and the page title all yield significant ($p < 0.01$) improvements over not including them. Metadata description and extracted noun phrases also give a significant ($p < 0.05$), but less strong, improvement. The different input data sources can be ranked in terms of helpfulness for personalization: Metadata keywords, extracted terms, title, metadata description and extracted noun phrases. However, a combination of the most helpful data sources does not necessarily achieve the best performance, as strong parameter interactions exists between the different sources. It can also be seen that using relative weighting performs consistently

better than giving every term of every data source a weight of 1. This can be explained by the fact that the most helpful data sources are the ones that contain the lowest number of terms, becoming more numerous as the data sources become less helpful.

The charts show that in general no term filtering performs significantly better ($p < 0.01$) than using Google N-Gram based filtering, which in turn performs significantly better ($p < 0.01$) than WordNet noun filtering. WordNet filtering seems to filter out too many actual relevant terms, which is reflected in its lower average NDCG score. Even though no filtering performs significantly better than Google N-Gram filtering, the actual difference in NDCG score is very small (0.001), which can be explained by the fact that N-Gram filtering only filters out non-sensical terms, which are rare in snippets, and not the words that harm in personalization.

When looking at term weighting methods, it seems that Term Frequency performs significantly worse ($p < 0.01$) than TF-IDF and BM25. BM25 performs on average significantly better ($p < 0.05$) than TF-IDF. However, when looking at the number of times a given parameter value is the best option for a parameter configuration, it can be seen that TF-IDF and BM25 are the best option for roughly the same number of approaches. BM25 seems to work better in general when the input data is richer and thus noisier, as it normalizes the term weights. TF-IDF is in general more successful when the selected set of input sources are more keyword focussed.

6.2.3 Relevance Distribution

Out of the 3,600 relevance judgements collected in the evaluation session, 9% is classified as Very Relevant, 32% is classified as Relevant and 58% as Irrelevant. With more than half of the returned results being judged as Irrelevant, it is clear that it is a challenge to get a lot of relevant results in the top of the ranking if no personalization is attempted.

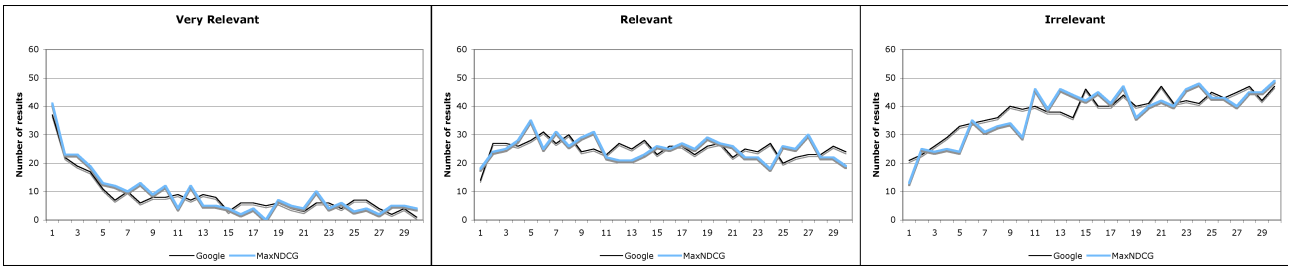


Figure 11: Distribution of relevance at rank for the Google and MaxNDCG rankings

The relevance judgements distribution for the default Google rank and the MaxNDCG re-ranking approach shown in Figure 11 indicates that web rank already manages to place the biggest part of the Very Relevant results in the top 5 results. However, the relevance distribution for the MaxNDCG strategy shows that personalized search manages to add more Very Relevant results into the top 5 of the ranking, but it mainly succeeds in adding Very Relevant results in between rank 5 and 10. This is expected as the personalization strategy keeps the Google rank into account and is thus less aggressive at very high ranks. However, this is also positive as it will cater for less personalizable queries or when a user's information need changes.

The Relevant results are evenly spread over all ranks, with slightly more weight towards the first half of the ranking. The personalization strategy manages to add a few more Relevant results to the top 5 of the different rankings. Irrelevant results are present at all ranks, but become more dominant after rank 10. Personalized search manages however to remove quite a large number of Irrelevant results out of the top 10 results and pushes them down into the lower regions of the rankings.

6.2.4 Selected Profiles

Given the discussed results, 3 promising profiles were selected to carry on with in the interleaved evaluation experiment. The chosen profiles are MaxNDCG, as it yielded the highest normalized NDCG, MaxQuer, as it improved on most queries and MaxBestPar as it uses the best parameters according to the greedy learning approach. All of these selected profiles use Language Model snippet weighting, take into account the Google rank, and multiply the weight of previously visited URLs by 10 and the number of visits.

7. Online Evaluation

In this section, the details of the large scale online interleaved evaluation are described first. Next, the obtained results are discussed and it is determined what the best personalization strategy is and whether it improves over Google.

7.1 Large Scale Interleaved Evaluation

The second stage in the evaluation process is a large scale online interleaved evaluation. It is very important that this evaluation is run over a long enough period of time to get a sufficient number of queries per user, and that a reasonable amount of users participate in order to get quantitative and reliable results. It is crucial that participants are able to do the evaluation during their day-to-day lives as realistic information needs come up. This attempts to assess whether any of the 3 personalization strategies selected in section 6 yield an actual improvement in a user's search experience. It is also critical that in doing this evaluation exercise, users' search experience is not altered and that no noticeable differences can be detected in the user interface.

Therefore, an updated version of the AlterEgo Firefox add-on was developed and all of the volunteers who downloaded and installed the initial version were asked to upgrade to this second version. The add-on detects when a user submits a web search via Google and sends the search query, the unique user identifier and the current page number over to the server. Next, the server requests the first 50 search results from Google using the Google Rest API for the given query and picks one of the three personalization strategies at random. The selected strategy is used to calculate the snippets weights and re-ranking is done accordingly.

The Team-Draft interleaving algorithm, as described in Algorithm 2, is then utilized to interleave the original Google ranking with the new personalized re-ranked version. In doing this, 50% of the search results are assigned to Team A, which represents the default Google ranking, and 50% is assigned to Team B, which represents the personalized ranking. A click that is recorded for either of these teams is considered a vote for the ranking they represent. The interleaved results are sent back to the user's browser, where they replace the original list of results, but keeping the surrounding page elements and thus not altering the search environment.

To avoid presenting slightly different rankings for a given query every time a search page is refreshed or paging is used, both the random personalization strategy selector and the random bit inside the interleaving algorithm were seeded with a combination of unique identifier, query and the current hour. Therefore, the same query submitted by the same user in the same hour would always produce the same ranking and thus not influence the user's search experience.

An additional user experience consideration that had to be taken into account was re-ranking performance. A user profile had an average size of about 3.4 Megabyte, and all of them were kept in memory for fast access. The Google Rest API is also limited to retrieving 8 results per request, and thus all 7 requests required to accumulate 50 results had to be done in an asynchronous fashion using a separate thread per Google REST API request. A script was also set up which kept all of the user profiles up-to-date and made them available to the server on an hourly basis. The technical architecture for this can be found in Appendix B.

7.2 Results and Discussion

In this section, the results from the interleaved evaluation experiment are presented. It is expected to be harder to show actual improvements for this evaluation since bringing a relevant result up to a rank beneath the most relevant result will likely not make a difference in a user's behavior. This experiment should however give a clearer idea of whether the suggested personalization methods yield an improvement in real life. The previous evaluation phase also had a strong emphasis on ambiguous queries, whilst the share of ambiguous queries should be lower in this evaluation, making it harder to successfully deploy search personalization, as relevance becomes more a matter of which page is higher quality content-wise.

This interleaved evaluation experiment was run over a 3 week period and 50 users, of which most are IT professionals, participated in it. A total of 2,715 queries were submitted, of which 1,847 received an actual click on a search result. 76% of these search queries were unique and 24% were repeated queries.

Method	Queries	Google Votes	Re-ranked Votes
MaxNDCG	590	220 (37.3%)	370 (62.7%)
MaxQuer	621	295 (47.5%)	326 (52.5%)
MaxBestPar	636	290 (45.7%)	346 (54.3%)

Table 9: Search Queries and Team votes per personalization strategy

Each click on a search result represented a vote for either the original Google ranking or the re-ranked personalized ranking and the user profile that was used to do the actual re-ranking. The total number of votes for each strategy can be seen in Table 9, showing that all of the investigated personalization approaches yield an actual improvement over the default Google ranking. MaxNDCG significantly outperforms ($p < 0.01$) web ranking and is considerably better than both MaxQuer and MaxBestPar. MaxQuer is significantly ($p < 0.05$) better and MaxBestPar also outperforms web ranking, although its improvements are not significant. The collected votes suggest that MaxNDCG is the most convincing and best performing personalization strategy, which is in line with the findings from the first evaluation experiment. MaxBestPar performs better than MaxQuer in the current experiment, which is in contrast with the NDCG scores obtained in the relevance judgements session. This can potentially be explained by the fact that MaxQuer uses Term Frequency as the term weighting algorithm, which due to its much higher dynamic range for term weights, might be re-ranking too aggressively compared to the normalized BM25 weights in MaxBestPar.

Method	Unchanged	Improved	Deteriorated
MaxNDCG	429	152 (23.9%)	55 (8.7%)
MaxQuer	444	112 (18.1%)	64 (10.3%)
MaxBestPar	403	128 (21.6%)	59 (10.1%)

Table 10: Rank quality implications for each Personalization Strategy

The effect personalization has on the set of search queries for the different strategies can be seen in Table 10. The Unchanged column indicates the number of queries for which personalization did not make a difference, i.e. the clicked result was at the same rank for both the default ranking and the personalized ranking. The Improved column shows the number of occasions in which the clicked result was brought up due to search personalization, whilst the Deteriorated

column shows the amount of queries for which the clicked result was actually lower in the ranking due to personalization and thus was worse than the original ranking. The trends that can be seen from the voting results are consistent with these numbers, however the actual differences between the different approaches seem smaller. On average, about 70% of the search queries are untouched by search personalization, 20% of the queries are improved and 10% becomes worse. MaxQuer seems like the least effective approach, having the highest percentage of unchanged queries, the highest percentage of deterioration and the lowest percentage of improvements. However, MaxQuer still improves more queries than it deteriorates, indicating that it is still a successful approach. MaxBestPar generally makes about the same number of queries worse, however its improvement percentage is 3.5% higher than for MaxQuer and it also leaves fewer search queries unchanged. MaxNDCG again clearly seems to be the most successful personalization approach, having the highest change and improvement rate and the lowest harming rate, improving 2.7 times more queries than it harms.

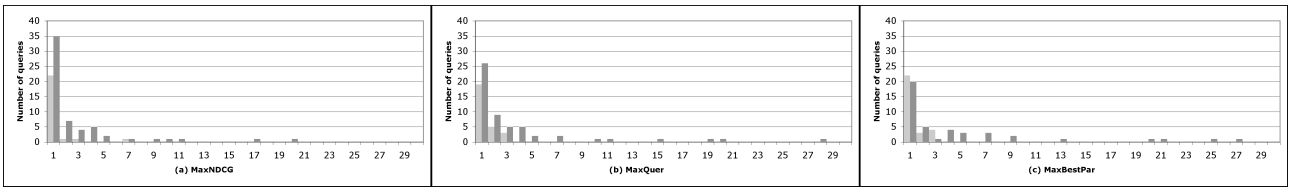


Figure 12: Rank differences for deteriorated (light grey) and improved queries (dark grey) for MaxNDCG, MaxQuer and MaxBestPar

Figure 12 shows, for each of the personalization strategies, the distribution of rank changes for all queries that were improved or became worse. It can be seen that for the vast majority of deteriorated queries, especially for MaxNDCG, the clicked result only loses 1 rank compared to the original ranking. The majority of clicked results that improved a query gain 1 rank as well, however there are quite a few relevant results that are bumped up 2, 3, 4 or even 5 ranks. For each of the personalization strategies, the average rank deterioration is about 1.38 and the average rank improvement is around 3.5, indicating that the gains achieved by personalization are on average about twice as high as the losses experienced on a query that was originally better. In other words, if personalization was unsuccessful for a query, a user could expect to find the result he was looking for at a rank which is on average only 1 lower than were it originally was. However, when personalization is successful, the result the user is looking for will on average be 3 or 4 ranks higher than original. Given this and the fact that there are a lot more improved than harmed queries, it very strongly indicates that the suggested approaches are very likely to not considerably harm non-personalizable or hard to personalize queries.

Figure 14 shows how much personalization is achieved at each rank through the suggested approaches. It can be seen that most re-ranking is done after rank 10, having little or no influence on a user's search experience. Less actual re-ranking is done in the first five ranks due to the rank normalization going on, which has already proven to be a worthwhile approach. MaxQuer does the least re-ranking in general, whilst MaxBestPar does most re-ranking after rank 10. It can be seen that MaxNDCG does considerably more re-ranking at ranks 1-5. This could explain why it is in practice the most successful approach, given that, according to the distribution of clicked ranks shown in Figure 13, it targets the range of ranks that is clicked most often. This is only true if MaxNDCG does not harm the quality of the ranking due to re-ranking and actually improves it, which seems to be the case for this approach, otherwise it would make the search experience worse than the original one.

The clicked rank distribution also strongly indicates that paging through search results is hardly ever used in practice, making it necessary for personalized search to both bring in relevant results that were originally lower than rank 10 into the top 10 and to get the order of the results as good as possible given the user's interests. Ideally, personalized search should target the top 5 of the search ranking as much as possible in order to make a real difference, however only if it can ensure that non-relevant web pages are not being inserted into the top 5.

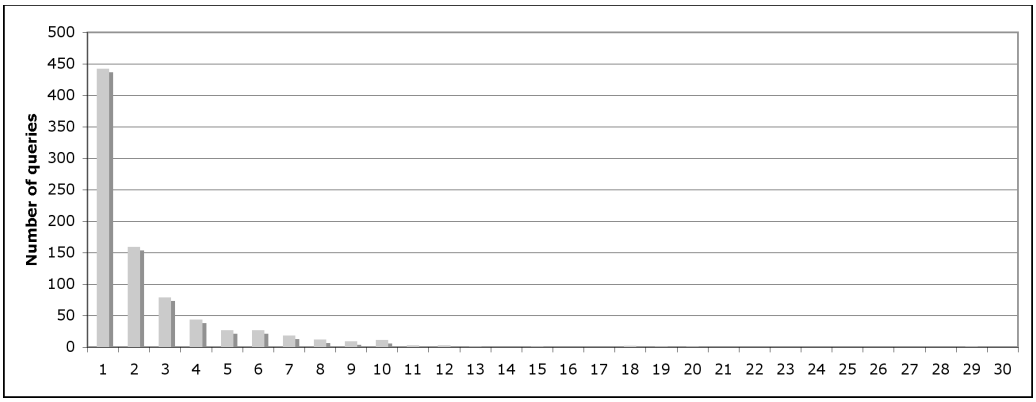


Figure 13: Distribution of clicked ranks

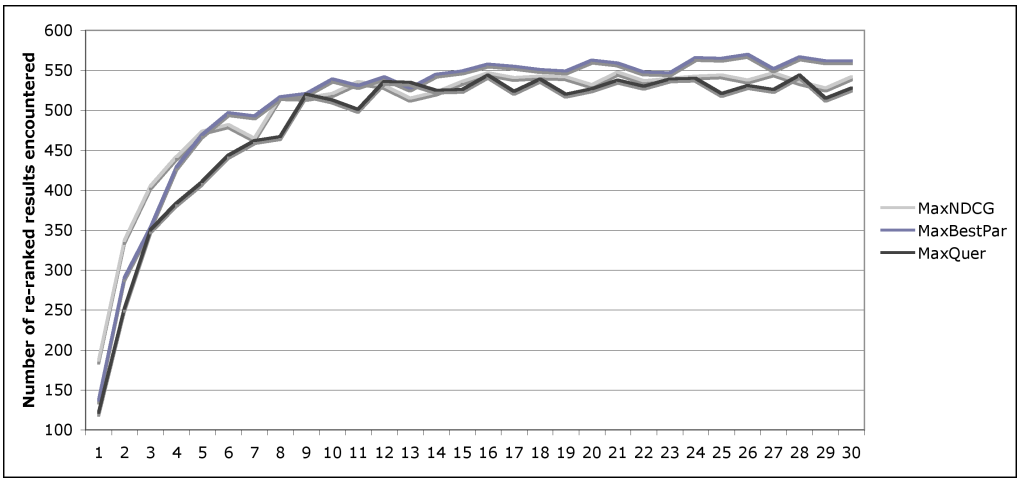


Figure 14: Degree of personalization per rank

In summary, it can be seen that the personalization strategy that uses Title, Metadata keywords and Extracted noun phrases as input data using relative weighting, no filtering and TF-IDF as the term weighting scheme, is the most effective according to both the voting method and the number of improved/deteriorated queries, yielding significant improvements over default web search and previous suggested personalization methods.

8. Conclusion and Future Work

In this thesis, personalized web search has been investigated in which first a user's long-term interests were learned and then it was attempted to re-rank the first 50 search results returned by a search engine in a user profile and click history based approach using full browsing history as the input data. A set of personalization techniques are proposed that significantly outperform both default Google ranking and the best previous personalization methodologies, which are also compared to each other for the first time in both small scale offline and large scale online experiments. The suggested methods also seem to be the first profile based approaches applied to all queries that manage to successfully personalize queries that have a potential for personalization and do not harm queries that are non-personalizable or are caused by a change in information need. This is also the first large scale personalized search and online evaluation work for general web search that was not carried out at a search company.

A method is presented in which user data and personalization performance results can be collected on large scale in a straightforward way using a browser plug-in based approach. A comprehensive evaluation framework is suggested which consists of an offline session allowing to determine the best approaches and a large scale online evaluation which assesses whether personalization can improve search in real life, showing that personalized search can actually achieve that.

It is discovered that the key to using web pages to model a user is to not treat a web page as a normal document, but to treat is as a structured document out of which several types of data can be extracted. It is also found that applying advanced NLP techniques like term extraction and shallow parsing can be beneficial for search personalization.

The suggested methods can be implemented straightforwardly and are feasible at large scale. One of the outcomes of this thesis is a personalized search Firefox add-on that can be publicly downloaded[16] and used without altering the user's browsing experience. The source code[17] is also available for download for the research community.

As the results obtained in this thesis are better than the default Google ranking and best previously suggested personalization strategies, a conference paper will be assembled out of it. The conference this paper will be submitted to is the WSDM (International Conference on Web Search and Data Mining) 2011 conference, which is a Tier 1 conference in web search, organized in Hong Kong. A first draft of that paper can be found in Appendix C.

There are a number of directions that can still be investigated. A first option would be to check whether the suggested methods and set of parameters can still be expanded and improved. Quite a few input data sources that were extracted from the HTML structure and have been included in the user XML files have not yet been used, like headings etc., and experiments could be carried out to evaluate whether any of these yield improvements. On top of that, a Firefox add-on has got access to a large number of user behavior and implicit relevance data, like time spent on a page, amount of scrolling, text selection, mouse activity, and so on. These could be collected and incorporated in the formula used to determine the weight given to a term.

The method suggested in this thesis relies on using a user's browsing history to learn his long-term interests. However, some strategies, like the Teevan approach that this thesis compares to, also make use of more personal data like files on their hard drive, e-mails, ... It would be worth investigating whether personalization methods would benefit further from this richer set of documents related to the user. However, it is worth noticing that it is a lot harder to extract all of the data sources used in the suggested methods from these documents as they are not as structured as HTML documents.

About 10 of the 50 users that participated in the experiments described in this thesis were native speakers of a language other than English. The results obtained suggest that the personalization strategies are quite robust to this, even though some of the more advanced NLP techniques are language specific. However, the influence of the presence of foreign languages in a user's browsing history has not been formally investigated. It is potentially interesting to investigate whether running language detection on a web page before it is processed and adjust the NLP techniques accordingly improves overall performance.

In all of the experiments described, the baseline system used to request the first 50 search results for a given query was the default Google search. However, Google also offers personalized search using geo-location and click-through based information. As the details of these algorithms are not publicly known and because it allowed for a more straightforward implementation, it was decided not to compare to the personalized version. In future experiments, a more thorough comparison could be done against better baselines that were beyond the scope of the presented research.

The current implementation of the search personalization browser add-on stores user data and profiles on a central server, making users potentially reluctant to use it due to privacy reasons. A useful next step would be to move this data storage and processing to the user's browser or a program installed on the user's computer, avoiding some of the currently existing privacy issues.

---

[16] http://alterego.caret.cam.ac.uk

[17] http://github.com/nicolaasmatthijs/AlterEgo

Finally, one could also look into using the extracted profiles for purposes other than personalized search. After the experiments described had passed, all participants were presented with one of their keyword based profiles. Most users indicated that they were stunned by how well these profiles described them and that they would use a similar set of keywords to describe themselves if asked. This indicates that there is a potential of using these keyword centric profiles in different areas. They could be used for personalizing advertisements, suggesting interesting news articles to read, interesting FaceBook groups that can be joined, and so on.

9. References

[1] Z. Chen and B. Fu. On the complexity of Rocchio's similarity-based relevance feedback algorithm. J. Am. Soc. Inf. Sci. Technol., 58(10):1392–1400, 2007.

[2] P.-A. Chirita, C. S. Firan, and W. Nejdl. Summarizing local context to personalize global web search. In CIKM '06: Proceedings of the 15th ACM international conference on Information and knowledge management, pages 287-296, New York, NY, USA, 2006. ACM.

[3] P. A. Chirita, W. Nejdl, R. Paiu, and C. Kohlschutter. Using ODP metadata to personalize search. In SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval, pages 178–185, New York, NY, USA, 2005. ACM.

[4] S. Clark and J. R. Curran. Wide-coverage efficient statistical parsing with CCG and log-linear models. Comput. Linguist., 33(4):493–552, 2007.

[5] M. Claypool, P. Le, M. Wased, and D. Brown. Implicit interest indicators. In IUI '01: Proceedings of the 6th international conference on Intel ligent user interfaces, pages 33–40, New York, NY, USA, 2001. ACM.

[6] S. Cronen-Townsend and W. B. Croft. Quantifying query ambiguity. In Proceedings of the second international conference on Human Language Technology Research, pages 104–109, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

[7] M. Daoud, L. Tamine-Lechani, and M. Boughanem. Learning user interests for a session-based personalized search. In IliX '08: Proceedings of the second international symposium on Information interaction in context, pages 57–64, New York, NY, USA, 2008. ACM.

[8] M. Daoud, L. Tamine-Lechani, M. Boughanem, and B. Chebaro. A session based personalized search using an ontological user profile. In SAC '09: Proceedings of the 2009 ACM symposium on Applied Computing, pages 1732–1736, New York, NY, USA, 2009. ACM.

[9] Z. Dou, R. Song, and J.-R. Wen. A large-scale evaluation and analysis of personalized search strategies. In WWW '07: Proceedings of the 16th international conference on World Wide Web, pages 581–590, New York, NY, USA, 2007. ACM.

[10] S. Gauch, J. Chaffee, and A. Pretschner. Ontology-based personalized search and browsing. Web Intel li. and Agent Sys., 1(3-4):219–234, 2003.

[11] T. Jaime, D. Susan, and H. Eric. Potential for personalization. ACM Trans. Comput.-Hum. Interact., 17(1):31, March 2010.

[12] B. J. Jansen, A. Spink, and T. Saracevic. Real life, real users, and real needs: a study and analysis of user queries on the web. Inf. Process. Manage., 36(2):207–227, 2000.

[13] T. 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.

[14] T. 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.

[15] T. Joachims, L. Granka, B. Pan, H. Hembrooke, F. Radlinski, and G. Gay. Evaluating the accuracy of implicit feedback from clicks and query reformulations in web search. ACM Transactions on Information Systems (TOIS), 25(2), April 2007.

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

[17] F. Liu, C. Yu, and W. Meng. Personalized web search by mapping user queries to categories. In CIKM '02: Proceedings of the eleventh international conference on Information and knowledge management, pages 558–565, New York, NY, USA, 2002. ACM.

[18] F. Liu, C. Yu, and W. Meng. Personalized web search for improving retrieval effectiveness. IEEE Trans. on Knowl. and Data Eng., 16(1):28–40, 2004.

[19] M. Morita and Y. Shinoda. Information filtering based on user behavior analysis and best match text retrieval. In SIGIR '94: Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval, pages 272–281, New York, NY, USA, 1994. Springer-Verlag New York, Inc.

[20] A. Pretschner and S. Gauch. Ontology based personalized search. In ICTAI '99: Proceedings of the 11th IEEE International Conference on Tools with Artificial Intel ligence, page 391, Washington, DC, USA, 1999. IEEE Computer Society.

[21] F. Qiu and J. Cho. Automatic identification of user interest for personalized search. In WWW '06: Proceedings of the 15th international conference on World Wide Web, pages 727–736, New York, NY, USA, 2006. ACM.

[22] F. Radlinski and N. Craswell. Comparing the sensitivity of information retrieval metrics. In To appear in Proceedings of SIGIR 2010. Association for Computing Machinery, Inc., 2010.

[23] F. Radlinski, M. Kurup, and T. Joachims. How does clickthrough data reflect retrieval quality? In CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge management, pages 43–52, New York, NY, USA, 2008. ACM.

[24] G. Salton and C. Buckley. Improving retrieval performance by relevance feedback. pages 355–364, 1997.

[25] M. Sanderson. Ambiguous queries: test collections need more sense. In SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval, pages 499–506, New York, NY, USA, 2008. ACM.

[26] X. Shen, B. Tan, and C. Zhai. Implicit user modeling for personalized search. In CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management, pages 824–831, New York, NY, USA, 2005. ACM.

[27] A. Sieg, B. Mobasher, and R. Burke. Web search personalization with ontological user profiles. In CIKM '07: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management, pages 525–534, New York, NY, USA, 2007. ACM.

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

[29] M. Speretta and S. Gauch. Personalized search based on user search histories. In WI '05: Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intel ligence, pages 622–628, Washington, DC, USA, 2005. IEEE Computer Society.

[30] S. Sriram, X. Shen, and C. Zhai. A session-based search engine. In SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval, pages 492–493, New York, NY, USA, 2004. ACM.

[31] K. Sugiyama, K. Hatano, and M. Yoshikawa. Adaptive web search based on user profile constructed without any effort from users. In WWW '04: Proceedings of the 13th international conference on World Wide Web, pages 675–684, New York, NY, USA, 2004. ACM.

[32] J.-T. Sun, H.-J. Zeng, H. Liu, Y. Lu, and Z. Chen. Cubesvd: a novel approach to personalized web search. In WWW '05: Proceedings of the 14th international conference on World Wide Web, pages 382–390, New York, NY, USA, 2005. ACM.

[33] F. Tanudja ja and L. Mui. Persona: A contextualized and personalized web search. In HICSS '02: Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS'02)-Volume 3, page 67, Washington, DC, USA, 2002. IEEE Computer Society.

[34] J. Teevan, S. T. Dumais, and E. Horvitz. Personalizing search via automated analysis of interests and activities. In SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval, pages 449–456, New York, NY, USA, 2005. ACM.

[35] J. Teevan, S. T. Dumais, and D. J. Liebling. To personalize or not to personalize: modeling queries with variation in user intent. In SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval, pages 163–170, New York, NY, USA, 2008. ACM.

[36] P. Thomas and D. Hawking. Evaluation by comparing result sets in context. In CIKM '06: Proceedings of the 15th ACM international conference on Information and knowledge management, pages 94–101, New York, NY, USA, 2006. ACM.

[37] A. T. A. Thuy Vu and M. Zhang. Term extraction through unithood and termhood unification. In Proceedings of the 3rd International Joint Conference on Natural Language Processing (IJCNLP-08), Hyderabad, India, 2008.

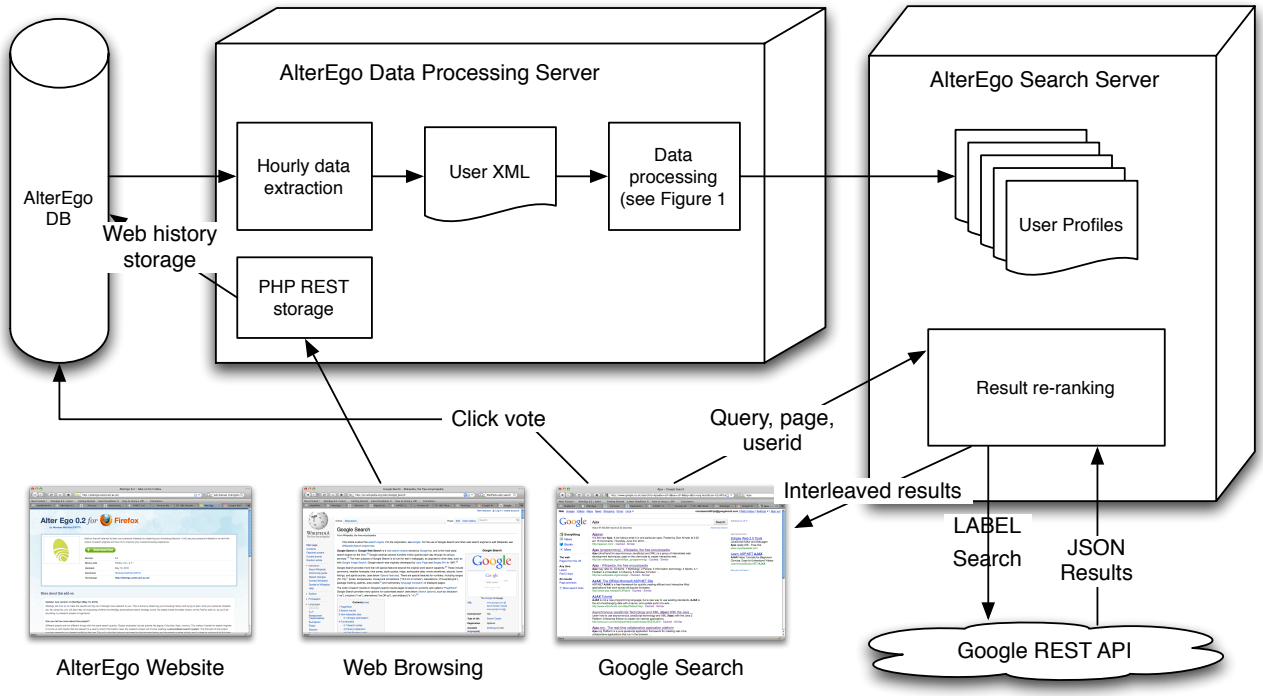
[38] R. W. White and S. M. Drucker. Investigating behavioral variability in web search. In WWW '07: Proceedings of the 16th international conference on World Wide Web, pages 21–30, New York, NY, USA, 2007. ACM.

Appendix A: User XML Schema

```
<?xml>
<user id="">
<entries="">
<documents>
<document>
  <url/>
  <duration/>
  <plaintext/>
  <structured>
    <head>
      <metadescription/>
      <metakeywords/>
      <title/>
    </head>
    <images>
      <img alt="" title=""/>
      <img alt="" title=""/>
    </images>
    <anchors>
      <a title="" href=""/>
      <a title="" href=""/>
    </anchors>
    <headings>
      <heading type=""/>
      <heading type=""/>
    </headings>
    <textual>
      <text type="p"/>
      <text type="span"/>
    </textual>
    <lists>
      <list type="ol">
        <item/>
        <item/>
      </list>
      <list type="ul">
        <item/>
        <item/>
      </list>
    </lists>
    <tables>
      <table>
        <cell type=""/>
        <cell type=""/>
      </table>
      <table>
        <cell type=""/>
        <cell type=""/>
      </table>
    </tables>
  </structured>
</document>
…
</documents>
```

Appendix B: Technical Schema

This Appendix gives an overview of the technical architecture and infrastructure used to offer a working version of AlterEgo to end users. It involves a Firefox add-on which can be downloaded from the AlterEgo website. This add-on captures the web pages a user visits and saves them into the database. An hourly process processes all of this data (term extraction, parsing, structure extraction, etc.) and saves the outcome as a user XML file. This user XML file is then processed into the required set of user profiles. Every time the user performs a Google search, the query, the page number and user's unique identifier are sent to the server. This server will take the user profile for that user, request the first 50 search results from Google and generate a personalized ranking. This ranking is then interleaved with the original Google ranking. The interleaved ranking is sent back to the user and clicks on any of the presented links are stored in the AlterEgo database.



Appendix C: WSDM Paper Draft

Personalizing Web Search Using Long Term Browsing History

Nicolaas Matthijs
University of Cambridge
15 JJ Thomson Avenue
Cambridge CB3 0FD, UK
nm417@cam.ac.uk

Filip Radlinski
Microsoft Research
7 J J Thomson Ave
Cambridge CB3 0FB, UK
filiprad@microsoft.com

Stephen Clark
University of Cambridge
15 JJ Thomson Avenue
Cambridge CB3 0FD, UK
sc609@cam.ac.uk

ABSTRACT

In this paper, we study various algorithms that use a person's complete browsing history as input data for web search personalization. Using the specific characteristics of the web and more advanced NLP techniques, we attempt to implicitly learn a user's interests and generate an interest profile. We develop an end to end search personalization system, being the first to give a detailed evaluation with both offline and online experiments. One of the additional goals of the paper was to develop a scalable and user-friendly tool that is directly useful and applicable to end users, and can be downloaded and used as a Firefox add-on. In doing this, we succeeded in obtaining results that are significantly better than the default search engine ranking or previously published personalized search strategies.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms

Algorithms, Experimentation, Measurement

Keywords

AlterEgo, Browsing History, Evaluation, Personalized Web Search, Interleaving, Searching, Ranking, User Profile

1. INTRODUCTION

Although information retrieval systems such as web search have become an essential part of our lives, there is still room for improvement. A major deficiency of current retrieval systems is that they are not adaptive enough to a user's individual needs and interests [10]. This can be illustrated with the search query "ajax". This query will return results about Ajax based web development, about the Dutch football team Ajax Amsterdam and websites about the cleanser named Ajax. Meanwhile, different users would clearly prefer different results. Without personalization, however, all users will be presented with the same ranking.

Additionally, previous research has noted that the vast majority of search queries are short [25, 11] and ambiguous [5, 22]. Often, different users will use the same query to express a completely different information need [10, 11, 19, 23, 35]. Personalized search is a potential solution to this problem.

A large range of personalization strategies have previously been suggested, including [31, 19, 1, 2, 15, 18, 26, 28, 30, 7, 29, 23, 17, 4, 16, 9, 24, 12, 8]. Our approach improves on these approaches with a realistic and scalable implementation.

Specifically, we model users with an automatically collected profile of all their browsing behavior. The content of the pages visited, along with the users particular behavior on web search results, is used to build a user model. This data is collected using a Firefox add-on, called AlterEgo[1], that we developed and made available to users. This profile was used to build a model then used to rerank the top search results returned by a non-personalized search engine. The key differences in our approach from previous work is that we parse the web pages structure, also using part of speech tagging and other filtering approaches to refine the user model. We show that it yields significantly retrieval improvements over web search and other personalization methods without requiring any effort on the user's behalf, and without changing the user's search environment.

The remainder of this paper is organized as follows. After presenting related work in Section 2, we give an overview of the different user profile generation and re-ranking strategies investigated in Section 3. Section 4 describes our evaluation approach, with results from our offline evaluation in Section 5, and online evaluation in Section 6. Section 7 offers concluding remarks, and future directions.

2. RELATED WORK

Many search personalization strategies have been suggested over the last years. In this section, we describe two groups of methods that have mainly been used in previous research.

The data used to construct such a user model can either be obtained in an explicit or an implicit manner. As explicit information, or *relevance feedback*, requires additional user effort, therefore we limit ourselves to the use of implicitly collected information about the user. In previous research,

[1]http://alterego.caret.cam.ac.uk

WSDM '11 Hong Kong

query history and click-through data are often used to model the users interests, as for example in [27]. Although it is shown that this can improve retrieval quality, such data is often sparse and additional information about the user can let us better understand the user's interests and information needs. Teevan et al [31] make use of a much richer user representation by utilizing a Desktop index which indexes files on the user's hard drive, e-mails, visited web pages and so on. However, this approach treats web documents as common documents and does not take advantage of the characteristics and structure encapsulated within a web page. In this paper, we make use of the richness of a user's complete browsing history, but we also exploit the specific characteristics and structure of web pages. Next to that, we also attempt to apply more advanced NLP techniques to these web documents and investigate whether the noisy nature of web pages influence has a negative effect on this. We find that this approach and taking advantage of web document structure both visibly improve retrieval quality.

Once a user's browsing behavior and interests have been learned, they can be used to re-rank the results returned by a search engine. For instance, a person who is a web developer issues the search query "Ajax". It is likely that both the search results about web development and the user's profile will contain words that are indicative of web development. These results can then be promoted to the top of the ranking based on similarities between the user profile and the relevant search results. In other words, we attempt to increase the chance of getting a relevant result near the top of the ranking.

Previous research [32] suggests that such profile based personalization may lack effectiveness on unambiguous queries like "london weather forecast" and therefore no personalization should be attempted for these queries. However, if this or a related query has been issued by this user before, we could detect any preference for certain weather forecast websites by using the user's URL history, which can also be deducted from the browsing history. We therefore expand upon a method which successfully incorporates a user profile and URL history into a personalized search framework [8]. However, there still exist scenarios in which search personalization might not help or even harm, for example when a query can not be personalized or when a user's information need changes over time. Our method keeps this in mind by also assessing a personalization strategy that is not too aggressive and still allows potentially less relevant results in at a slightly lower rank.

Profile Based Approaches

Some personalization techniques [31, 19, 28] are based on a user profile that expresses the individual's interests and browsing behavior. This can be done explicitly through information provided by the user, which we will not consider due to the extra effort involved on the user's behalf. Various methods and a wide range of information sources have also been proposed to learn a user's profile implicitly without any user effort.

In [7], the user profile is inferred from the entire search history and is used to model long term user interests. Shen et al [23] make use of the recent search history to model the short term user interests, in which session boundaries are used to define short term search history. These methods often suffer from data sparsity and very frequently re-rank results relying on a very limited amount of data.

Other methods have attempted to incorporate more information about the user by using the full browsing history [28, 17]. The Curious browser, a web browser developed to record a user's explicit relevance ratings of web pages and browsing behavior when viewing a page, such as dwell time, mouse clicks and scrolling behavior, is described in [4]. The most promising profile based approach was suggested by Teevan et al. [31]. They use a rich model of user interests, built from both search-related information, previously visited web pages and other information about the user (e.g. documents on their hard drive, e-mails etc.) to re-rank web search results within a relevance feedback framework. In doing this, they obtain a significant improvement over default web ranking. We compare our method to this approach in section 5 and show significant improvement in retrieval performance. The method described in [1] is based on solely using a user's desktop information.

Concerning the model used to describe a user, user interests can be represented as a set of keyword vectors [6], a set of concepts [16], an instance of a predefined ontology [9, 24, 18, 30] or a hierarchical category tree based on ODP and corresponding keywords [15, 2]. In this paper, we will focus on modeling users through a vector of weighted terms.

Click-through Based Approaches

A different range of personalization strategies utilize URL and click-through data from past queries. In [12], user click-through data is collected as training data to learn a retrieval function, which is used to produce a customized ranking of search results that suits a group of users' preferences. In [28, 26, 29], the click-through data collected over a long time period is exploited through query expansion to improve retrieval accuracy. The most promising URL and click-through based method seems to be PClick, or person-level re-ranking based on historical clicks, as suggested in [8]. If a query is issued by a user for the second time, pages that have been clicked during the first search for this query are promoted to the top of the ranking. A disadvantage of this approach is that it can only be applied to repeated queries. The method suggested in this paper will incorporate both a user profile and a user's URL and click-through history. We compare our approach to the PClick method in section 5, and find obtain significant improvements.

Commercial Personalization Systems

Recently, personalized search has also been made available in some of the mainstream web search engines including Google[2] and Yahoo!. These appear to use a combination of explicitly and implicitly collected information about a user. They allow the user to build a profile of themselves by selecting categories of interests and custom tailor the results delivered to the user based on that. However, in practice we expect few users to provide this explicit information. Implicitly, users' search and click-through history is also used personalize results, with results closer to that geographical location of the user additionally favored. However, as the details of these methods and algorithms are not publicly available, we only compare our approach to the default search engine ranking and not the personalized version.

[2]http://googleblog.blogspot.com/2009/12/personalized-search-for-everyone.html

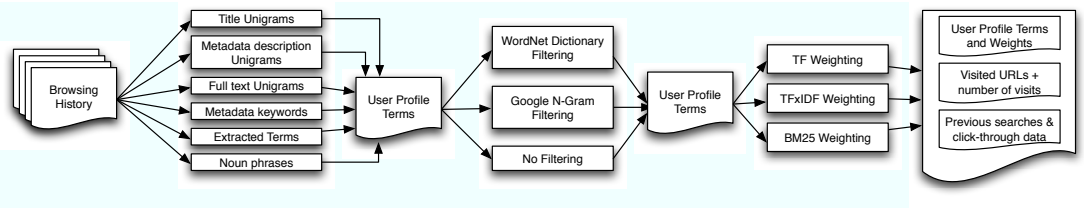


Figure 1: User Profile Generation Steps and Workflow

3. PERSONALIZATION STRATEGIES

In this section, we describe our personalization approach. The first step consists of constructing a user profile, that is then used in a second phase to re-rank search results.

3.1 User Profile Generation

A user is represented by a list of terms and weights associated to those terms, a list of visited URLs and the number of visits to each, and a list of past search queries and pages clicked for these search queries. This profile is generated as shown in Figure 1. First, a user's browsing history is collected and stored as (URL, HTML content) pairs. Next, this browsing history is processed into six different summaries consisting of term lists. Next, the terms can be filtered, and finally term weights are generated using three different weighting algorithms. We now describe each of these steps in detail.

3.1.1 Data Capture

To obtain user browsing histories, a Firefox add-on called AlterEgo[3] was developed. To respect a user's privacy as much as possible, a random unique identifier is generated at installation time. This identifier is used for all data exchange between the add-on and the server recording the data[4]. Participants for this study were recruited via a website explaining the purpose and consequences to potential users, publicized on various e-mail lists, resulting in 50 participants taking part. Whilst we expect that most of these participants are employed in the IT industry due to the recruitment process, a number of people outside of the IT industry without significant web search experience participated as well.

Every time a user with this add-on installed leaves a non-secure (non-https) web page, the add-on transmits the current user's unique identifier, the URL of the page, the time that was spent on the page, the current date and time, and the length of the source HTML to the server. The server then adds the record to a queue of items to process. The server attempts to fetch the source HTML of all pages in the queue. This is performed server-side to ensure that only publicly-visible data is used. Once the source HTML is received, the server compares its length to the length received from AlterEgo. If the length difference is smaller than 50 characters, the HTML is accepted and saved along with the unique identifier, URL, duration and date and time into the database. Otherwise, we assume the content probably came from a password protected but non-secure site (e.g. FaceBook, Hotmail etc.) and the record is discarded.

[3]The source code for this add-on can be downloaded from http://github.com/nicolaasmatthijs/AlterEgo

[4]Note that it is necessary for this data to be collected by our server for research purposes, but our approach does not require this data to be centralized. Our entire method can execute client-side, avoiding the privacy concerns that otherwise arise with server-based approaches.

Table 1: Captured Data Statistics

Metric	Total	Min	Max	Mean
Page Visits	530,334	51	53,459	10,607
Unique Page Visits	218,228	36	26,756	4,364.56
Google Searches	39,838	0	4,203	797
Bing Searches	186	0	53	3.72
Yahoo Searches	87	0	29	1.74
Wikipedia Pages	1,728	0	235	34.56

The add-on captured data for three months from March to May 2010. As shown in Table 1, a total of 530,334 page visits (or an average of 10,607 page visits per user) were recorded. 58% of the visits were to unique pages. The add-on also recorded 39,838 Google searches, 186 Bing searches and 87 Yahoo! searches, indicating that our users are strongly biased towards Google as their search engine, hence Google is used as the baseline in our experiments. An average user issued 797 queries over the three months, indicating that at least 7.5% of all web requests are search related.

3.1.2 Data Extraction

We considered the following summaries of the content viewed by users in building the user profile:

Full Text Unigrams
The body text of each web page, stripped of html tags.

Title Unigrams
The words inside any <*title*> tag on the html pages.

Metadata Description Unigrams
The content inside any <*meta name="description"*> tag on the html pages.

Metadata Keywords Unigrams
The content inside any <*meta name="keywords"*> tag on the html pages.

Extracted Terms
We implemented the Term Extraction algorithm based on Unithood And Termhood Unification as presented in [34], running it on the full text of each visited web page. This algorithm uses the C/NC method, which uses a combination of linguistic and statistical information to score each term. Term candidates are found using a number of linguistic patterns and are assigned a weight based on the frequency of the term and its subterms. This is supplemented with term

Table 2: Extracted terms from the AlterEgo website and the Wikipedia page about Mallorca

AlterEgo	Mallorca
add-ons	majorca
addons	island
Nicolaas	palma
Matthijs	islands
CSTIT	spanish
Nicolaas Matthijs	balearic
Cambridge	mallorca
Language Processing	cathedral
Google	Palma de Mallorca
keyword extraction	port

re-extraction using the Viterbi algorithm. The outcome of this algorithm run on two sample web pages can be seen in Table 3.

Noun Phrases

Noun phrases were extracted by taking the text from each web page and splitting it into sentences using a sentence splitter from the OpenNLP Tools[5]. The OpenNLP tokenization script was then run on each sentences. The tokenized sentences were then tagged using the Clark & Curran Statistical Language Parser[6] [3], which assigns a constituent tree to the sentence, part of speech tags to each word. Noun phrases were then extracted from this constituent tree.

3.1.3 Term List Filtering

To reduce the number of noisy terms in our user representation, we also tried filtering terms using two different approaches.

The first is dictionary filtering using the lexical database WordNet 3.0[7], retaining only words marked as nouns. However, WordNet is far from complete and hence tends to remove many nouns words, in particular most proper nouns.

The alternative way we tried was removing uncommon words using the Google N-Gram corpus[8]. This approach is general successfully remove non-sensical word, while allowing less common nouns such as proper nouns. However, as part of speech tags are not a part of the Google N-Gram corpus, it could not filter out stop words and determiners.

3.1.4 Term Weighting

After the list of terms has been accumulated and potentially filtered, we computed weights for each term using three methods.

TF Weighting

The most straightforward implementation we consider is Term Frequency (TF) weighting. We define a frequency vector $\vec{F}$ that contains the frequency counts of a given term t_i for all of the input data sources, as shown in equation (1). For example, f_{title} is the number of times a given term t_i occurs in all of the titles in the user's browsing history. We calculate a term weight based on the dot product of these frequencies with a weight vector $\vec{\alpha}$:

$$\vec{F}_{t_i} = \begin{bmatrix} f_{title_{t_i}} \\ f_{mdesc_{t_i}} \\ f_{text_{t_i}} \\ f_{mkeyw_{t_i}} \\ f_{terms_{t_i}} \\ f_{nphrases_{t_i}} \end{bmatrix} \quad (1)$$

$$w_{TF}(t_i) = \vec{F}_{t_i} \cdot \vec{\alpha} \quad (2)$$

For simplicity, in our experiments we limit ourselves to three possible values for each weight α_i: 0, ignoring the particular data field, 1, including the particular data field, and $\frac{1}{N_i}$, where N_i is the total number of terms in field i. This gives more weight to terms in shorter fields (such as the meta keywords or title fields). We call the latter *relative weighting*.

TF-IDF Weighting

The second possibility for term weights we consider is TF-IDF (or Term Frequency, Inverse Document Frequency) weighting. Here, words appearing in many documents are down-weighted by the inverse document frequency of the term:

$$w_{TFIDF}(t_i) = \frac{1}{log(DF_{t_i})} \times w_{TF}(t_i) \quad (3)$$

To obtain IDF estimates for each term, we use the inverse document frequency of the term on all web pages using the Google N-Gram corpus.

Personalized BM25 Weighting

The final weight method we consider is taken from Teevan et al [31]. They propose a personal term weight similar to how BM25 weights terms. Specifically, they propose a weight

$$w_{BM25}(t_i) = log\frac{(r_{t_i}+0.5)(N-n_{t_i}+0.5)}{(n_{t_i}+0.5)(R-r_{t_i}+0.5)}, \quad (4)$$

where N represents the number of documents on the web (estimated from the Google N-Gram corpus, 220,680,773), n_{t_i} is the number of documents in the corpus that contain the term t_i (estimated using the Google N-Gram corpus), R is the number of documents in the user's browsing history and r_{t_i} is the number of documents in the browsing history that contains this term within the selected input data source.

While this method allows us to compare our results against the approach proposed by Teevan et al., note that we do not have access to users' full Desktop index, and are limited to their browsing history, making our implementation potentially less effective.

3.2 Re-ranking Strategies

Like previous work, we use the user profile to re-rank the top results returned by a search engine to bring up results that are more relevant to the user. This allows us to take advantage of the data search engines use to obtain their initial ranking to first obtain a small set of results that can then be personalized. In particular, [31] noted that chances are high that even for an ambiguous query the search engine will be quite successful in returning pages for the different meanings of the query. We opt to retrieve and re-rank the

[5]http://opennlp.sourceforge.net/
[6]http://svn.ask.it.usyd.edu.au/trac/candc/wiki
[7]http://wordnet.princeton.edu/
[8]http://googleresearch.blogspot.com/2006/08/all-our-n-gram-are-belong-to-you.html

first 50 results retrieved for a query. Each of these 50 search results is given a weight and are re-ranked accordingly.

3.2.1 Weighting Methods

When reranking, each candidate document can either be scored, or just the snippets can be scored. We focus on assigning scores to the search snippets as it was found to be more effective for re-ranking search results by Teevan et al. [31]. Also, using search snippets allows a straightforward client-side implementation of search personalization. We implemented the following four different weighting methods:

Matching

For each word in the search snippet's title and summary that is also in the user's list of profile terms, the weight associated with that term will be added to the snippet's weight:

$$score_M(s_i) = \sum_{z=0}^{N_{s_i}} f_{t_i} \times w(t_z) \quad (5)$$

where N_{s_i} represents the total number of unique words within the snippet's title and summary, and f_{t_i} represents the number of occurrences of t_i within the snippet. Words in the snippet title or summary but not in the user's profile do not contribute towards the final weight. This method is equivalent to taking the dot product between the user profile vector and the snippet vector.

Unique Matching

A second search snippet weighting option we consider involves counting each unique word just once:

$$score_{UM}(s_i) = \sum_{z=0}^{N_{s_i}} w(t_z) \quad (6)$$

Language Model

The third weight calculation method attempts to generate a unigram language model from the user profile in which the weights associated to the terms are used as the frequency counts for the language model:

$$\begin{aligned} score_{LM}(s_i) &= log(\prod_{z=0}^{N_{s_i}} \frac{w(t_z)+1}{w_{total}}) \\ &= \sum_{z=0}^{N_{s_i}} log(\frac{w(t_z)+1}{w_{total}}) \end{aligned} \quad (7)$$

where N is the total number of words in the snippet's title and summary, and w_{total} stands for the sum of all the weights within the user profile. The language model estimates the probability of a snippet given a user's profile. To avoid a zero probability for snippets that contain words not in the user's profile, we use add-1 smoothing.

PClick

As a final snippet weighting method we use the PClick algorithm proposed by Dou et al. [8]. It assumes that for a query q submitted by a user u, the web pages frequently clicked by u in the past are more relevant to u. The personalized score for a snippet is:

$$score_{PC}(s_i) = \frac{|Clicks(q,p,u)|}{|Clicks(q,\bullet,u)| + \beta} \quad (8)$$

where $|Clicks(q,p,u)|$ is the number of clicks on web page p by user u for query q in the past, $|Clicks(q,\bullet,u)|$ is the total click number on query q by u, and β is a smoothing factor set to 0.5. Note that PClick makes no use of the terms and weights associated to the user's profile and is solely based on click-through data for a given query. As such, it only affects repeated queries.

3.2.2 Additional Options

Finally, we consider two adjustments to the snippet scores. First, we consider giving additional weight to URLs that have been visited previously. This extends PClick in that it boosts *all* URLs that have previously been visited, while PClick only boosts URLs that have directly been clicked for the current search query. The snippet weight will be boosted by the number of previous visits to that web page (n) times a factor v:

$$\text{finalScore}(s_i) = score(s_i) * (1 + v \times n_i) \quad (9)$$

Second, in the re-ranking framework discussed so far, the original ranking is not taken into account. The original rank can be incorporated into the final snippet weight by multiplying the snippet weight by the inverse log of the snippet's original rank r_{s_i}:

$$\text{finalScore}(s_i) = score(s_i) \times \frac{1}{log(r_{s_i})} \quad (10)$$

We expect both these extensions to be very beneficial.

4. EVALUATION APPROACH

We now consider potential evaluations for personalized search strategies. On the one hand, offline approaches allow the creation of a standard dataset that can be used to optimize personalization parameters. On the other hand, only an online test with actual users can truly reflect how changes to rankings affect user behavior. We now explore the available alternatives, and describe our final strategy.

Relevance judgements

The first offline evaluation approach (e.g. used by Teevan et al. [31]) is based on assembling a group of people that judge the relevance of the top k documents or search snippets for a set of queries. Given these relevance judgements, a standard metric such as (N)DCG or (Normalized) Discounted Cumulative Gain [**?**] can be calculated for a given query and ranking, reflecting the quality of the presented ranking for that user. This approach has the advantage that once the relevance judgements are made, it allows for testing many different user profile and re-ranking parameter configurations. However, due to the long time it takes to judge k documents, this can only be done for a small number of search queries. As volunteers need to be found to sit through this slow and tedious evaluation process, it is also hard to gather a large group of evaluators. The evaluation process also does not reflect a user's normal browsing and searching behavior, which might influence the final results. Moreover, this approach assumes that (N)DCG is the right way to combine a set of relevance judgements into a rank quality score. Finally, the queries evaluated must be representative of a true query load, or offline results may not reflect perhaps poorer performance for non-personalizable queries.

Side-by-side evaluation

An alternative offline evaluation method, previously used for example by [33], consists of presenting users with two alternative rankings side-by-side and ask which they consider best. The advantage of this method is that an actual judgement is made of which ranking is the best one, although users evaluate the entire presented ranking whilst in real life situations they might only look at the first couple of results, potentially biasing the results. Judging two rankings next to each other is considerably faster than judging k documents per query, but it still requires a long offline evaluation exercise. Additionally, an evaluator has to provide a new assessment for each distinct ordering of documents that is investigated. This makes it hard to use such judgments to tune reranking parameters.

Clickthrough-based evaluation

One online evaluation approach involves looking at the query logs and click-through data from a search engine on large scale (e.g. used by [8]). The logs record which search result was clicked for a given query, allowing the evaluator to check if the clicked result would be positioned higher in a personalized ranking. This allows for testing many parameter configurations and also does not require any user effort as their actions are recorded as they go, reflecting their natural environment and browsing behavior. However, the method can have difficulties in assessing whether a search personalization strategy actually works. First, users are more likely to click a search result presented at a high rank, although these are not necessarily most or more relevant [14]. It is also unsuccessful in assessing whether the results that have been brought up from a page lower down would have been relevant as they would rarely have been clicked if originally shown at low rank. On top of that, we also have no access to such large scale usage and user profile data for this experiment.

Alternatively, both personalized and non-personalized rankings can be shown online to users, with metrics such as mean clickthrough rates and positions being computed. However, [21] showed that such an approach is not sensitive enough to detect small differences in relevance with thousands of query impressions as we could obtain in an online experiment.

Interleaved evaluation

The final online evaluation option we consider, which to our knowledge has not been used before for the evaluation of personalized search, is interleaved evaluation [13, 21]. Interleaved evaluation combines the results of two search rankings by alternating between results from the two rankings while omitting duplicates, and the user is presented with this interleaved ranking. The ranking that contributed the most clicks over many queries and users is considered better. Radlinski et al. [21] showed that this approach is much more sensitive to changes in ranking quality than other click-based metrics. It has also shown to correlate highly with offline evaluations with large numbers of queries [20]. On top of that, this method does not require any additional effort from the user, providing an evaluation users naturally use search engines. However, one evaluation only provides an assessment for one particular ranking, and thus an evaluation is required for each parameter configuration being investigated. It is also the hardest evaluation technique for showing improvements as, opposed to other metrics, bringing a slightly relevant page up from rank 8 to rank 5 will not help if the most relevant page is at rank 1.

4.1 Evaluation Plan

This last approach, interleaved evaluation, best reflects real user experience, and would be preferred for evaluating a personalized search system. However, the user profile generation and re-ranking steps both have a large number of possible parameters and it is infeasible to perform an online evaluation for all of them. Hence, we start with an offline NDCG based evaluation to pick the optimal parameter configurations that we then evaluate with the more realistic online interleaved evaluation.

5. OFFLINE EVALUATION

This section first describes how we collected relevance judgments for offline evaluation. Next, we describe how this was used to identify the most promising personalization strategies.

5.1 Relevance Judgements

To obtain personalized relevance judgments, six participants who had installed the AlterEgo plugin recording their browsing behavior were recruited. At that point, two months of browsing history had been recorded and stored for each of them.

The process and format of the evaluation sessions was kept very similar to the relevance judgment session conducted by Teevan et al [31]. Each participant was asked to judge the relevance of the top 50 web pages returned by Google for 12 queries according to the criteria presented in Table 3. The documents were presented in a random order. Full web pages were evaluated instead of snippets because the user is only interested in the actual web page being relevant for his information need.

Every participant was asked to provide 600 judgments, 50 relevance judgements for each of 12 queries. The first query participants were asked to judge was their own name (Firstname Lastname) as a warm-up exercise. Next, each participant was presented with a list of 25 general search queries in a random order, consisting of 16 queries taken from the TREC 2009 Web Search track queries and 9 other UK focussed queries such as "pub", "football" and "cambridge". All users were asked to judge 6 of these queries. Examples of some of the queries chosen by multiple people can be seen in Table 4. Finally, each participant was presented with their most recent 40 search queries (from their browsing history) and were asked to judge 5 for which they remembered the returned results could have been better.

On average, it took each participant about 2.5 hours to complete this exercise. Particularly interestingly, all users mentioned that during to the exercise they came across really interesting websites which they did not know existed before, indicating that there is a potential for search personalization to enrich the set of returned results.

5.2 Results and Discussion

The following parameters were investigated for profile generation, representing the different steps shown in Figure 1:

- All combinations of the six different input data sources with three possible values for α (namely 0, 1 and normalized)

Table 3: Relevance judgements guidelines

(a) Select *Irrelevant* if the document is not useful and not interesting to you.
(b) Select *Relevant* if the document is interesting to you, but is not directly about what you were hoping to find or if the document is somewhat useful to you, meaning that it touches on what you were hoping to find (maximum 1 paragraph), but not very extensively.
(c) Select *Very Relevant* if the document is useful or very interesting to you, i.e. it is what you were hoping to find.

Table 4: Common queries used by different participants

Query	Users
Cambridge	6
GPS	4
Website design hosting	3
Volvo	3

- Three filtering methods: No Filtering, WordNet noun filtering and Google N-Gram filtering using a threshold of 1,000
- The three term weighting methods: TF, TF-IDF and BM25

In terms of re-ranking the search results, the following set of parameters were investigated:

- The four snippet weighting methods: Matching, Unique Matching, Language Model and PClick
- Whether or not to consider the original Google rank
- Whether or not to give extra weight to previously visited URLs (using $v = 10$)

For every profile and ranker combination, the mean personalized NDCG was measured as follows:

$$NDCG@p = \frac{1}{Z} \sum_{i=1}^{p} \frac{2^{rel_i} - 1}{\log_2(1+i)} \tag{11}$$

where rel_i is determined from the relevance judgements (non-relevant = 0, relevant = 1 and very relevant = 2) and Z is such that the maximum NDCG for each query is 1. In all of these following results, we compare NDCG scores that have been averaged across all queries and all users.

5.2.1 Baseline Comparisons

To assess how well our different personalization strategies performed, we compared the outcome of the experiment with several baselines as shown in Figures 2 and 3. The reported scores are normalized DCG scores for all 72 queries, and all statistical analyses were performed using a paired T-Test.

The baseline methods we compare to are the default Google ranking, the Teevan method and the PClick method. The results obtained appear to agree with those reported in [31] and [8]. We find the Teevan approach is significantly ($p < 0.05$) better than Google and PClick is significantly ($p < 0.01$) better than Google. Note that in our implementation of the Teevan algorithm we only make use of the browsing history as input data, as we do not have access to the user's files or e-mails, which may disadvantage it.

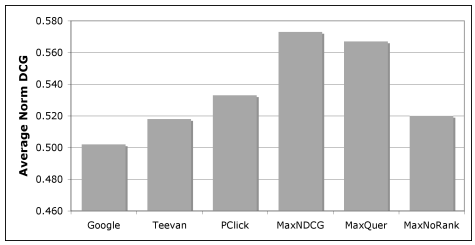


Figure 2: Average Normalized DCG scores comparing baseline system performance to best personalization strategies

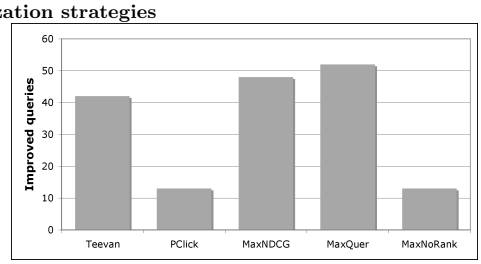


Figure 3: Comparing baseline system performance to best personalization strategies in terms of number of improved queries

We selected four user profile and re-ranking parameter settings, summarized in Table **??**, to further evaluate. The first, `MaxNDCG`, yielded the highest average NDCG score on the offline dataset. It involves using Title, Metadata Keywords and Extracted noun phrases as input data using relative weighting, no filtering, and TF-IDF as the weighting method. The second, `MaxQuer`had the highest number of improved queries. It consisted of using Extracted Terms and Extracted noun phrases as input data using relative weighting, no filtering and TF as the weighting method. The third, `MaxNoRank`, was the best method that does not take the original Google ranking into account. It uses Metadata keywords as input data, no filtering and TF as the weighting method. Finally, `MaxBestPar`was obtained by a greedy selection strategy in each parameter was selected one at a time and then locked. It is very similar to `MaxNDCG`, consisting of using Title, Metadata Keywords and Extracted Terms as input data using relative weighting, no filtering and BM25 for term weight calculation. All of these selected profiles used Language Model snippet weighting, the Google rank was taken into account and previously visited URLs received addition weight per Equation 9 with $v = 10$.

`MaxNDCG`and `MaxQuer`are both significantly ($p < 0.01$) better than default Google, Teevan and PClick. `MaxNDCG`, with an average NDCG of 0.573, yields a 14.1% improvement over Google, and `MaxQuer`, with an average NDCG of 0.567, yields a 12.9% improvement over Google.

Interestingly, despite `MaxNoRank`ignoring the Google rank, it obtaining an NDCG score of 0.520 that is significantly ($p < 0.05$) better than Google and better than Teevan. A

Table 5: Investigated profiles for interleaved evaluation

Name	Method	Avg NDCG	Improved
`MaxNDCG`	TF-IDF, RTitle, RMKeyw, RCCParse, NoFilt - LM, Look At Rank, Visited	0.573	48/72
`MaxQuer`	TF, RTerms, RCCParse, NoFilt - LM, Look At Rank, Visited	0.567	52/72
`MaxNoRank`	TF, RMKeyw, NoFilt - LM, Look At Rank, Visited	0.520	13/72
`MaxBestPar`	BM25, RTitle, RMKeyw, RTerms, NoFilt - LM, Look At Rank, Visited	0.566	45/72

ranking that outperforms web ranking based on user data has to our knowledge not yet been achieved. While this may be a result of overfitting the parameters given our small offline dataset, we observed this effect often that we do not believe this to be the case.

A different metric that can be used for comparison of personalization methods is to look at the number of queries for which the NDCG score improved, as shown in Figure 3. PClick improves fewest, 13 out of 72 queries, compared to the 44 improved queries for the Teevan approach, despite obtaining a higher NDCG score. This is because the PClick method only works on repeated queries by bringing up pages on which the user previously clicked. While this will only occasionally cause personalization, it makes bigger improvements when it reranks results. Also, the Teevan approach has a negative effect on some of the queries as well. `MaxNDCG`improves performance on 48 queries, `MaxQuer`improves 52.

5.2.2 Parameter Configuration

Given all the investigated user profile and re-ranking combinations, we look at how all of these parameters alter the overall strategy performance. We summarize the one-way effects of all individual parameters and explore their influence on the average NDCG score. In Figure , we consider a certain parameter λ (e.g. filtering, term weighting, etc.). For every other parameter setting, we counted how often the different possible values of λ were most helpful. We repeated this experiment for every parameter group. These approaches do not examine interaction effects, so at the end of this section we give an overview of which parameter combinations achieved the best performance. Note that most parameters only make a small difference in NDCG, even if they are preferred in most cases.

In terms of re-ranking strategies, it is clear that there is one approach that significantly outperforms all other ones, indicating that this approach should be used for all generated profiles. Using the Matching approach for snippet weight calculation is significantly worse ($p < 0.01$) than using Unique Matching. Using a Language Model based on the user profile performs significantly better ($p < 0.01$) than both Matching and Unique Matching and is in 67% of the investigated profiles the best choice. Unique Matching only occasionally performs better than a Language Model when the user profile is very much keyword based and does not use Extracted Nouns, Title, etc. as input data. Except for a single case, multiplying the snippet weight by 10 for URLs that have previously been visited performs significantly better ($p < 0.01$) than not keeping this into account. This proves that a combination of a user profile based and click based personalization strategy can be used successfully. Not normalizing the obtained snippet weights by their original rank always performs worse than when the Google rank is taken into account.

When looking at the input data used to generate a user's profile, it can be seen that all data sources are helpful in improving personalization performance, except when the full web page text is used. Not using the full text performs significantly better ($p < 0.01$) than when this is included. This indicates that treating web pages like a normal document or a bag of words does not work, presumably due to its noisy nature. Using metadata keywords, extracted terms and the page title all yield significant ($p < 0.01$) improvements over not including them. Metadata description and extracted noun phrases also give a significant ($p < 0.05$), but less strong, improvement. The different input data sources can be ranked in terms of helpfulness for personalization: metadata keywords, extracted terms, title, metadata description and extracted noun phrases. However, a combination of the most helpful data sources does not necessarily achieve the best performance, as strong parameter interactions exists between the different sources. It can also be seen that using relative weighting performs consistently better than giving every term of every data source a weight of 1. This can be explained by the fact that the most helpful data sources are the ones that contain the lowest number of terms, becoming more numerous as the data sources become less helpful.

The charts show that in general no term filtering performs significantly better ($p < 0.01$) than using Google N-Gram based filtering, which in its turns performs significantly better ($p < 0.01$) than WordNet noun filtering. WordNet filtering seems to filter out too many actual relevant terms, which is reflected in its lower average NDCG score. Even though no filtering performs significantly better than Google N-Gram filtering, the actual difference in NDCG score is very small (0.001), which can be explained by the fact that N-Gram filtering only filters out non-sensical terms, which are rare in snippets, and not the words that harm in personalization.

When looking at term weighting methods, it seems that Term Frequency performs significantly worse ($p < 0.01$) than TF-IDF and BM25. BM25 performs on average significantly better ($p < 0.05$) than TF-IDF. However, when looking at the number of times a given parameter value is the best option for a parameter configuration, we can see that TF-IDF and BM25 are the best option for roughly the same number of approaches. BM25 seems to work better in general when the input data is richer and thus noisier, as it normalizes the term weights. TF-IDF is in general more successful when the selected set of input sources are more keyword focussed.

5.2.3 Relevance Distribution

Of the 3,600 relevance judgements collected in the evaluation session, 9% were Very Relevant, 32% Relevant and 58% as Non-Relevant. The relevance judgements distribution for the default Google rank and the `MaxNDCG` re-ranking approach shown in Figure 5 indicates that web rank already manages to place the biggest part of the Very Relevant results in the top 5 results. However, the relevance distribu-

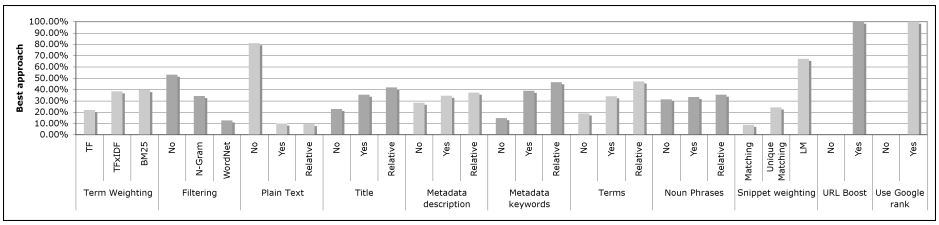


Figure 4: Percentage-wise number of times an investigated parameter performs best for a given parameter configuration

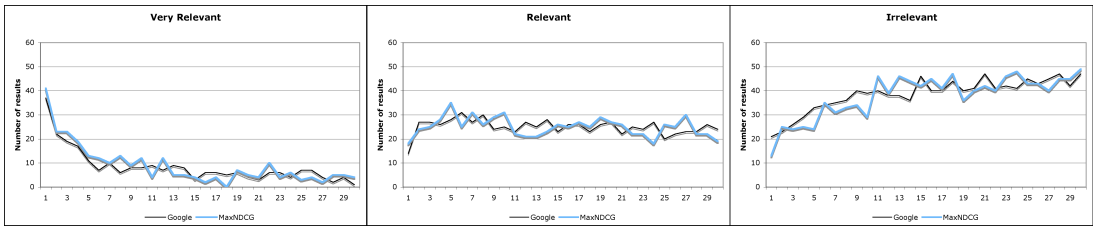


Figure 5: Distribution of relevance at rank for the Google and `MaxNDCG` rankings

tion for the `MaxNDCG` strategy shows that personalized search manages to add more Very Relevant results into the top 5 of the ranking, but it mainly succeeds in adding Very Relevant results in between rank 5 and 10. This is expected as the personalization strategy keeps the Google rank into account and is thus less aggressive at very high ranks.

The Relevant results are evenly spread over all ranks, with slightly more weight towards the first half of the ranking. The personalization strategy manages to add a few more Relevant results to the top 5 of the different rankings. Irrelevant results are present at all ranks, but become more dominant after rank 10. Personalized search manages however to remove quite a large number of Irrelevant results out of the top 10 results and pushes them down into the lower regions of the rankings.

5.2.4 Selected Profiles

Given the discussed results, we selected 3 promising profiles to carry on with in the interleaved evaluation experiment. The chosen profiles are `MaxNDCG`, as it yielded the highest normalized NDCG, `MaxQuer`, as it improved on most queries and `MaxBestPar` as it uses the best parameters according to our greedy learning approach. All of these selected profiles use Language Model snippet weighting, take into account the Google rank, and multiply the weight of previously visited URLs by 10 and the number of visits.

6. ONLINE EVALUATION

The second stage in the evaluation process is a large scale online interleaved evaluation. It is very important that this evaluation is run over a long enough period of time to get a sufficient number of queries per user, and that a reasonable amount of users participate in order to get quantitative and reliable results. It is crucial that participants are able to do the evaluation during their day-to-day lives as realistic information needs come up. This attempts to assess whether any of the 3 personalization strategies selected in section 5 yield an actual improvement in a user's search experience. It is also critical that in doing this evaluation exercise, users' search experience is not altered and they no noticeable differences can be detected in the user interface.

Algorithm 1 Team-Draft Interleaving

1: **Input**: Rankings $A = (a_1, a_2, \dots)$ and $B = (b_1, b_2, \dots)$
2: **Init**: $I \leftarrow (); TeamA \leftarrow \emptyset; TeamB \leftarrow \emptyset;$
3: **while** $(\exists i : A[i] \notin I) \wedge (\exists j : B[j] \notin I)$ **do**
4: **if** $(|TeamA| < |TeamB|) \vee$
$((|TeamA| = |TeamB|) \wedge (RandBit() = 1))$ **then**
5: $k \leftarrow \min_i \{i : A[i] \notin I\}$ ... *top result in A not yet in I*
6: $I \leftarrow I + A[k];$ ..... *append it to I*
7: $TeamA \leftarrow TeamA \cup \{A[k]\}$ ..... *clicks credited to A*
8: **else**
9: $k \leftarrow \min_i \{i : B[i] \notin I\}$ ... *top result in B not yet in I*
10: $I \leftarrow I + B[k]$ ..... *append it to I*
11: $TeamB \leftarrow TeamB \cup \{B[k]\}$ .... *clicks credited to B*
12: **end if**
13: **end while**
14: **Output**: Interleaved ranking I, $TeamA$, $TeamB$

In this section, we first describe the details of the large scale online interleaved evaluation. Next, we discuss the results obtained and find out what the best personalization strategy is and whether it improves over Google. ADD: Given our limited offline relevance data, if we didn't do this, we might overfit.

6.1 Interleaving Implementation

Therefore, an updated version of the AlterEgo Firefox add-on was developed and all of the volunteers who downloaded and installed the initial version were asked to upgrade to this second version. The add-on detects when a

Table 6: Search Queries and Team votes per personalization strategy

Method	Queries	Google Vote	Re-ranked Vote
MaxNDCG	590	220 (37.3%)	370 (62.7%)
MaxQuer	621	295 (47.5%)	326 (52.5%)
MaxBestPar	636	290 (45.7%)	346 (54.3%)

Table 7: Rank quality implications for each Personalization Strategy

Method	Unchanged	Improved	Deteriorated
MaxNDCG	429 (67.5%)	152 (23.9%)	55 (8.7%)
MaxQuer	444 (71.6%)	112 (18.1%)	64 (10.3%)
MaxBestPar	403 (68.3%)	128 (21.6%)	59 (10.1%)

user submits a web search via Google and sends the search query, the unique user identifier and the current page number over to the server. Next, the server requests the first 50 search results from Google for the given query and picks one of the three personalization strategies at random. The selected strategy is used to calculate the snippets weights and re-ranking is done accordingly.

In our experiments, we use the Team-Draft interleaving algorithm, as described in Algorithm 1, to interleave the original Google ranking and the re-ranked personalized version. This algorithm is motivated by how sports teams are often assigned in friendly games. Given a pool of available players (ranking A and B), two captains (one for TeamA and one for TeamB) take turns picking their next favorite player in the set of remaining players, subject to a coin toss every turn deciding which team captain gets to pick first. More details about this approach can be found in [21]. The combined ranking is presented to the user and clicks on any of the results are recorded. If results from the personalized ranking end up being clicked more often, it is a strong indicator that the personalization is successful.

The Team-Draft interleaving algorithm, as described in Algorithm 1, is then utilized to interleave the original Google ranking with the new personalized re-ranked version. In doing this, 50% of the search results are assigned to Team A, which represents the default Google ranking, and 50% is assigned to Team B, which represents the personalized ranking. A click that is recorded for either of these teams is considered a vote for the ranking they represent. The interleaved results are sent back to the user's browser, where they replace the original list of results, but keeping the surrounding page elements and thus not altering the search environment.

To avoid presenting slightly different rankings every time a search page is refreshed, both the random personalization strategy selector and the random bit inside the interleaving algorithm were seeded with a combination of unique identifier, query and the current hour.

An additional user experience consideration that had to be taken into account was re-ranking performance. A user profile had an average size of about 3.4 Megabyte, and all of them were kept in memory for fast access. Also, the user profiles were automatically updated on an hourly basis.

6.2 Results and Discussion

In this section, we present the results from the interleaved evaluation experiment. We expect it to be harder to show

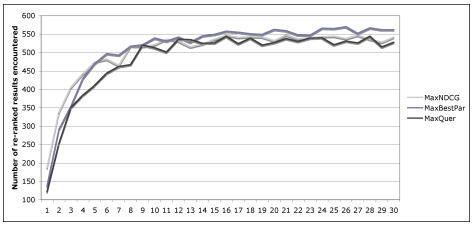


Figure 7: Degree of personalization per rank

actual improvements for this evaluation since bringing a relevant result up to a rank beneath the most relevant result will likely not make a difference in a user's behavior. This experiment should however give a clearer idea of whether our suggested personalization methods yield an improvement in real life. The previous evaluation phase also had a strong emphasis on ambiguous queries, whilst the share of ambiguous queries should be lower in this evaluation, making it harder to successfully deploy search personalization.

This interleaved evaluation experiment was run over three weeks, for the 41 users who updated the plugin. A total of 2,715 queries were submitted, of which 1,847 received an actual click on a search result. 76% of these search queries were unique and 24% were repeated queries.

Each click on a search result represented a vote for either the original Google ranking or the re-ranked personalized ranking and the user profile that was used to do the actual re-ranking. The total number of votes for each and each strategy can be seen in Table 6, showing that all of the investigated personalization approaches yield an actual improvement over the default Google ranking. MaxNDCGsignificantly outperforms ($p < 0.01$) web ranking by more than both MaxQuerand MaxBestPar. MaxQuerand MaxBestParboth outperform web ranking as well, although their improvements are not statistically significant. The collected votes suggest that MaxNDCGis the most convincing and best performing personalization strategy, which is in line with the findings from the first evaluation experiment. Although MaxBestParperforms better than MaxQuerin the current experiment, which is in contrast with the NDCG scores obtained in the relevance judgements session, the difference between these algorithms is not statitically significant.

The effect personalization has on the set of search queries for the different strategies can be seen in Table 7. The Unchanged column indicates the number of queries for which personalization did not make a difference, i.e. the clicked result was at the same rank for both the default ranking and the personalized ranking. The *Improved* column shows the number of occasions in which the clicked result was brought up due to search personalization, whilst the *Deteriorated* column shows the amount of queries for which the clicked result was lower in the ranking due to personalization. The trends that can be seen from the voting results are consistent with these numbers, however the actual differences between the different approaches seem smaller. On average, about 70% of the search queries are untouched by search personalization, 20% of the queries are improved and 10% becomes worse. MaxQuerseems like the least effective approach, having the highest percentage of unchanged queries, the highest percentage of deterioration and the lowest percentage of im-

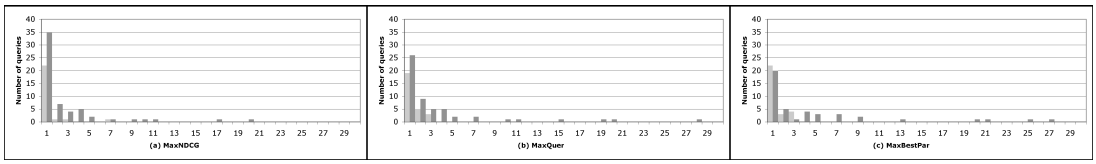


Figure 6: Rank differences for deteriorated (light grey) and improved queries (dark grey) for `MaxNDCG`, `MaxQuer`and `MaxBestPar`

provements. However, `MaxQuer`still improves more queries than the ones it makes worse, indicating that it is still a successful approach. `MaxBestPar`generally makes about the same number of queries worse, however its improvement percentage is 3.5% higher than for `MaxQuer`and it also leaves fewer search queries unchanged. `MaxNDCG`again clearly seems to be the most successful personalization approach, having the highest change and improvement rate and the lowest harming rate, improving 2.7 times more queries than it harms.

Figure 6 shows, for each of the personalization strategies, the distribution of rank changes for all queries that were improved or became worse. It can be seen that for large majority of the deteriorated queries, especially for `MaxNDCG`, the clicked result only loses 1 rank compared to the original ranking. The majority of clicked results that improved a query gain 1 rank as well, however there are quite a few relevant results that are bumped up 2, 3, 4 or even 5 ranks. For each of the personalization strategies, the average rank deterioration is about 1.38 and the average rank improvement is around 3.5, indicating that the gains achieved by personalization are on average more than twice as high as the losses experienced on a query that was originally better. In other words, if personalization was unsuccessful for a query, a user could expect to find the result he was looking for at a rank which is on average only 1 lower than were it originally was. However, when personalization is successful, the result the user is looking for will on average be 3 or 4 ranks higher than original.

Figure 7 shows how much personalization is achieved through the suggested approaches at each rank. It can be seen that most re-ranking is done after rank 10, having little or no influence on user's search experience. Less actual re-ranking is done in the first 5 ranks due to the rank normalization going on, which has already proven to be a worthwhile approach. `MaxQuer` does the least re-ranking in general, whilst `MaxBestPar` does most re-ranking after rank 10.

In summary, it can be seen that the personalization strategy that uses Title, Metadata keywords and Extracted noun phrases as input data using relative weighting, no filtering and TF-IDF as the term weighting scheme, is the most effective according to both the voting method and the number of improved/deteriorated queries, yielding significant improvements over default web search.

7. CONCLUSION & FUTURE WORK

In this paper, we have investigated personalized web search in which we first try to learn a user's long-term interests and then attempt to re-rank the first 50 search results returned by a search engine in a user profile and click history based approach using full browsing history as the input data. We propose a set of personalization techniques that significantly outperform both default Google ranking and the best previous personalization methodologies, which are also compared to each other for the first time in both small scale offline and large scale online experiments. Our methods also seem to be the first profile based approaches, applied to all queries that manage to successfully personalize queries that have a potential for personalization and does not harm queries that are non-personalizable. This is also the first large scale personalized search and online evaluation work for general web search that was not carried out at a search company.

We present a method in which user data and personalization performance results can be collected on large scale in a straightforward way using a browser plug-in based approach.

We discover that the key to using web pages to model a user is to not treat a web page as a normal document, but to treat is as a structured document out of which several types of data can be extracted. We also find that applying advanced NLP techniques like term extraction and parsing can be beneficial for search personalization. The suggested methods can be implemented straightforwardly and are feasible at large scale. One of the outcomes of this paper is a personalized search Firefox add-on that can be publicly downloaded[9] and used without altering the user's browsing experience. The source code is also available for download for the research community[10].

There are a number of directions that can still be investigated. A first option would be to check whether the suggested methods and set of parameters can still be expanded and improved and whether they can benefit from having more data available. Using other field, such as headings in HTML, were not explored. On top of that, a Firefox add-on has access to other behavioral information, such as time spent on a page, amount of scrolling, text selection and mouse activity, that we do not explore. Similarly to the Teevan approach, we could also make use of more personal data like files on their hard drive and e-mails.

In all of the experiments described, the baseline system used the first 50 search results return by the default Google ranker. However, Google also offers personalized search using geo-location and click-through based information. As the details of these algorithms are not publicly known and because it allowed for a more straightforward implementation, we decided not to compare to the personalized version. In future experiments, a more thorough comparison could be done against better baselines that were beyond the scope of our research.

Finally, one could also look into using the extracted profiles for purposes other than personalized search. After the

[9]http://alterego.caret.cam.ac.uk
[10]http://github.com/nicolaasmatthijs/AlterEgo

experiments described had passed, all participants were presented with one of their keyword based profiles. Most users indicated that they were stunned by how well these profiles described them and that they would use the same set of keywords to describe themselves if asked. This indicates that there is a potential of using these keyword centric profiles in different areas. They could be used for personalizing advertisements, suggesting interesting news articles to read, interesting FaceBook groups that can be joined, and so on.

8. REFERENCES

[1] P.-A. Chirita, C. S. Firan, and W. Nejdl. Summarizing local context to personalize global web search. In *CIKM '06: Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 287–296, New York, NY, USA, 2006. ACM.

[2] P. A. Chirita, W. Nejdl, R. Paiu, and C. Kohlschütter. Using odp metadata to personalize search. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 178–185, New York, NY, USA, 2005. ACM.

[3] S. Clark and J. R. Curran. Wide-coverage efficient statistical parsing with ccg and log-linear models. *Comput. Linguist.*, 33(4):493–552, 2007.

[4] M. Claypool, P. Le, M. Wased, and D. Brown. Implicit interest indicators. In *IUI '01: Proceedings of the 6th international conference on Intelligent user interfaces*, pages 33–40, New York, NY, USA, 2001. ACM.

[5] S. Cronen-Townsend and W. B. Croft. Quantifying query ambiguity. In *Proceedings of the second international conference on Human Language Technology Research*, pages 104–109, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

[6] M. Daoud, L. Tamine-Lechani, and M. Boughanem. Learning user interests for a session-based personalized search. In *IIiX '08: Proceedings of the second international symposium on Information interaction in context*, pages 57–64, New York, NY, USA, 2008. ACM.

[7] M. Daoud, L. Tamine-Lechani, M. Boughanem, and B. Chebaro. A session based personalized search using an ontological user profile. In *SAC '09: Proceedings of the 2009 ACM symposium on Applied Computing*, pages 1732–1736, New York, NY, USA, 2009. ACM.

[8] Z. Dou, R. Song, and J.-R. Wen. A large-scale evaluation and analysis of personalized search strategies. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 581–590, New York, NY, USA, 2007. ACM.

[9] S. Gauch, J. Chaffee, and A. Pretschner. Ontology-based personalized search and browsing. *Web Intelli. and Agent Sys.*, 1(3-4):219–234, 2003.

[10] T. Jaime, D. Susan, and H. Eric. Potential for personalization. *ACM Trans. Comput.-Hum. Interact.*, 17(1):31, March 2010.

[11] B. J. Jansen, A. Spink, and T. Saracevic. Real life, real users, and real needs: a study and analysis of user queries on the web. *Inf. Process. Manage.*, 36(2):207–227, 2000.

[12] T. 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.

[13] T. 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.

[14] T. Joachims, L. Granka, B. Pan, H. Hembrooke, F. Radlinski, and G. Gay. Evaluating the accuracy of implicit feedback from clicks and query reformulations in web search. *ACM Transactions on Information Systems (TOIS)*, 25(2), April 2007.

[15] F. Liu, C. Yu, and W. Meng. Personalized web search by mapping user queries to categories. In *CIKM '02: Proceedings of the eleventh international conference on Information and knowledge management*, pages 558–565, New York, NY, USA, 2002. ACM.

[16] F. Liu, C. Yu, and W. Meng. Personalized web search for improving retrieval effectiveness. *IEEE Trans. on Knowl. and Data Eng.*, 16(1):28–40, 2004.

[17] M. Morita and Y. Shinoda. Information filtering based on user behavior analysis and best match text retrieval. In *SIGIR '94: Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 272–281, New York, NY, USA, 1994. Springer-Verlag New York, Inc.

[18] A. Pretschner and S. Gauch. Ontology based personalized search. In *ICTAI '99: Proceedings of the 11th IEEE International Conference on Tools with Artificial Intelligence*, page 391, Washington, DC, USA, 1999. IEEE Computer Society.

[19] F. Qiu and J. Cho. Automatic identification of user interest for personalized search. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 727–736, New York, NY, USA, 2006. ACM.

[20] F. Radlinski and N. Craswell. Comparing the sensitivity of information retrieval metrics. In *To appear in Proceedings of SIGIR 2010*. Association for Computing Machinery, Inc., 2010.

[21] F. Radlinski, M. Kurup, and T. Joachims. How does clickthrough data reflect retrieval quality? In *CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge management*, pages 43–52, New York, NY, USA, 2008. ACM.

[22] M. Sanderson. Ambiguous queries: test collections need more sense. In *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 499–506, New York, NY, USA, 2008. ACM.

[23] X. Shen, B. Tan, and C. Zhai. Implicit user modeling for personalized search. In *CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 824–831, New York, NY, USA, 2005. ACM.

[24] A. Sieg, B. Mobasher, and R. Burke. Web search personalization with ontological user profiles. In *CIKM '07: Proceedings of the sixteenth ACM conference on Conference on information and*

knowledge management, pages 525–534, New York, NY, USA, 2007. ACM.

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

[26] M. Speretta and S. Gauch. Personalized search based on user search histories. In *WI '05: Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 622–628, Washington, DC, USA, 2005. IEEE Computer Society.

[27] S. Sriram, X. Shen, and C. Zhai. A session-based search engine. In *SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 492–493, New York, NY, USA, 2004. ACM.

[28] K. Sugiyama, K. Hatano, and M. Yoshikawa. Adaptive web search based on user profile constructed without any effort from users. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, pages 675–684, New York, NY, USA, 2004. ACM.

[29] J.-T. Sun, H.-J. Zeng, H. Liu, Y. Lu, and Z. Chen. Cubesvd: a novel approach to personalized web search. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 382–390, New York, NY, USA, 2005. ACM.

[30] F. Tanudjaja and L. Mui. Persona: A contextualized and personalized web search. In *HICSS '02: Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS'02)-Volume 3*, page 67, Washington, DC, USA, 2002. IEEE Computer Society.

[31] J. Teevan, S. T. Dumais, and E. Horvitz. Personalizing search via automated analysis of interests and activities. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 449–456, New York, NY, USA, 2005. ACM.

[32] J. Teevan, S. T. Dumais, and D. J. Liebling. To personalize or not to personalize: modeling queries with variation in user intent. In *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 163–170, New York, NY, USA, 2008. ACM.

[33] P. Thomas and D. Hawking. Evaluation by comparing result sets in context. In *CIKM '06: Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 94–101, New York, NY, USA, 2006. ACM.

[34] A. T. A. Thuy Vu and M. Zhang. Term extraction through unithood and termhood unification. In *Proceedings of the 3rd International Joint Conference on Natural Language Processing (IJCNLP-08)*, Hyderabad, India, 2008.

[35] R. W. White and S. M. Drucker. Investigating behavioral variability in web search. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 21–30, New York, NY, USA, 2007. ACM.