Journal of Web Engineering, Vol. 12, No. 1&2 (2013) 065–092

PERSONALIZING SEARCH USING SOCIALLY ENHANCED INTEREST MODEL, BUILT FROM THE STREAM OF USER'S ACTIVITY

TOMÁŠ KRAMÁR
Faculty of Informatics and Information Technologies, Slovak University of Technology, Ilkovičova 3, 842 16 Bratislava, Slovakia
kramar@fiit.stuba.sk

MICHAL BARLA
Faculty of Informatics and Information Technologies, Slovak University of Technology, Ilkovičova 3, 842 16 Bratislava, Slovakia
barla@fiit.stuba.sk

MÁRIA BIELIKOVÁ
Faculty of Informatics and Information Technologies, Slovak University of Technology, Ilkovičova 3, 842 16 Bratislava, Slovakia
bielik@fiit.stuba.sk

Received March 31, 2011
Revised August 7, 2012

Older studies have proved that when searching information on the Web, users tend to write short queries, unconsciously trying to minimize the cognitive load. However, as these short queries are very ambiguous, search engines tend to find the most popular meaning – someone who does not know anything about cascading stylesheets might search for a music band called css and be very surprised about the results. In this paper we propose a method which can infer additional keywords for a search query by leveraging a social network context and a method to build this network from the stream of user's activity on the Web. The approach was evaluated on real users using a personalized proxy server platform. The query expansion method was integrated into Google search engine and where possible, the original query was expanded and additional search results were retrieved and displayed. 70% of the expanded results were clicked and we observed a significant increase of time that the users spent on the expanded results when compared to the time spent on standard results.

Keywords: search personalization, implicit feedback, social networks, query expansion, metadata, user activity, personalized proxy
Communicated by: D. Schwabe & F. Vitali

1 Introduction

In our daily contact with the Web, we are faced with a vast amount of documents and their number increases every moment. To facilitate a systematic navigation in this chaotically linked large information space, search engines were created. They allow us to search for documents by entering a set of keywords – if these keywords are found in the text of the document, it is retrieved and presented to the user.

However, the fulltext-only search is often incapable of handling user queries satisfactorily and has been gradually shifting towards becoming more personalized. Many keywords are ambiguous, their meaning varies from context to context and from person to person. Some words are ambiguous by nature, e.g., a coach might be a bus or a person; other words became ambiguous only after being adopted for a particular purpose. Many English nouns, apart from their natural meaning, often name software, music band or any other entity. There are also words whose meaning depends on the person who is using them; clearly, architecture means different things to a processor designer than to an architect.

Ambiguity naturally disappears if more keywords are added to a query – together, they form a context, which defines the meaning of particular words, which would be ambiguous otherwise. For instance, if we extend our "coach" query with a word "repair", it is immediately clear that we mean a bus not a person meaning of the "coach". Based on the previous observations, we might conclude that using short queries is not a good idea. Unfortunately, this is how we search. We unconsciously try to minimize the cognitive load [1] and use short queries, with average length of about two keywords [2].

In the common demonstration of the problem, a user is searching using a query "jaguar". According to Wikipedia,[a] this word has 31 documented meanings, ranging from the well-known car or operating system, to the more obscure meanings such as a music band, game console or German battle tank. Traditional search engines do not have the knowledge of the meaning the user is looking for; they treat all documents the same, looking for the textual matches and ordering the resulting documents by a ranking function. As a consequence, the search results contain a mixed set of documents, covering all possible meanings of the word and the disambiguation is left to the user herself. Search engines work like databases: they crawl and index documents and respond to queries with a list of results. The order of documents depends on the adopted relevance function; the most widely used search engine today – Google – uses a PageRank relevance function: the more links to a document, the more likely it is to appear at the top positions. This ordering is however not always compatible with user's information needs: a programmer searching for "cucumber" probably does not want to make a salad.

In [3], authors suggest two ways to personalize search results. The first way is to modify the original query for each user and add, remove or replace keywords to move the search closer to her needs. The other way is to keep the query unchanged, but modify the ranking scheme and thus incorporate user's interests.

We tackle the problem by implicitly inferring the context and modifying the user's query to include it. The original query is enriched with additional keywords which capture the user's focus. In case of the said programmer, the resulting query might be "cucumber testing" which provides much more valuable and relevant documents than the original query. We select additional keywords following the social network, or rather the virtual community the user belongs to in this network. The search thus becomes personalized – the same query for another user from another community might be "cucumber salad".

Our key results include:

1. We build a metadata based model of user's interests. The metadata (keywords, terms, tags etc.) are automatically extracted from each visited page. We use a combination of

[a] Wikipedia article on the possible meanings of the word jaguar, `http://en.wikipedia.org/wiki/Jaguar_(disambiguation)`

automatically captured metadata together with human created data.

2. We enhance the interest models with data from similar users extracted from a social network where users are linked based on the similarities of their interests.

3. We designed a specialized proxy server [4] which allows us to monitor user's activity on the whole "wild" Web. Using the proxy, we can hook into every page and track not only its access, but also the implicit feedback indicators [5, 6] such as scrolling, copying and the real time a user spent on the page and personalize the page content.

We describe results of a month-long, real-users evaluation. Using the proxy server, we integrated our method of query expansion into Google search engine and injected the results based on our expanded queries next to the original results. We noticed not only that 70% of the expanded results were clicked but also observed a significant increase in the implicit relevance metrics concerning the expanded results. We evaluated the results page usefulness by looking at the time the user spent on it. A clicked result was declared as useful, if its dwell time was larger than 4 seconds. In our experiments, only 27% of clicked standard results were useful, while the clicked expanded results were useful in 54% of cases.

2 Related Work

Apart ambiguity, the biggest problem with search is that the expectations of different users entering the same query are probably very different [7]. If the search engine wants to provide a user with the relevant results, it needs to understand user's informational intent – a set of user-specific expectations and goals [8]. User's intent may be identified via information about who the user is, what are her needs, her interests and goals, both long-term and intermediate.

The value of personalized search has been recognized as early as in 2000; Lawrence [9] emphasized the role of personalization and predicted that personalized search will gradually become more and more important. In [3], the authors have identified two major strategies of search personalization.

- Query refinement, which is also commonly referred to as query expansion. The basic idea is that the original query is altered to better express user's intent. The individual terms in a query may be altered, new terms may be added or existing terms may be removed. From one point of view, this is a strategy that ensures that when two users enter the same query, it is changed so that each user runs a different query.

- Result processing is a less transparent strategy. If two users issue the same query, the same query is run, but the personalization occurs at the ranking stage. The user's needs may be taken into account and some documents with content that is more closely matching user's intent may be ranked higher. Similarly, the documents that are predicted to contain no useful information may be ranked lower.

From the two approaches, search results processing seems to be more widely used. The key work was presented in [10], where the authors describe a version of PageRank that is biased towards user's interests – a topic sensitive PageRank. Instead of a single rank value, 16 values are calculated, each representing the rank of the page in context of one of the 16 top-level ODP topics. Their work however does not deal with the problem of detecting the actual interests

of the users. This gap was later filled e.g., by [11]. They propose a method for building an interest-based user model based on the ODP taxonomy and the Topic-Driven Random Surfer Model. The user model is a weighted vector representing user's interests in 16 top-level ODP categories.

Unlike document re-ranking, query refinement is more visible and users see the modified query and are free to further refine it. The process itself does not have a consistent naming and different works refer to it as query reformulation, query expansion or query modifications. In its generic form, the query reformulation deals with query modification and may be employed for different purposes. From the point of view of a personalized search system, the goal of query expansion is different: it is to alter the query (remove, add or modify terms) in a way that the final query describes more closely the specific goal that the user has in mind. Query expansion is used in various areas concerning search, including:

- solving the index term synonymy problem by expanding each query term with a list of its synonyms [12];
- correcting the misspelled queries [13];
- suggesting similar queries, queries which have the same informational intent, but use different terms to describe it. The approaches range from statistics-based [14], which do not deal with the underlying meaning of the query terms, to semantic [15], which can distinguish between various meanings.

Query expansion is most often used for improving the precision of search; by adding new terms to the query, the scope of the query is narrowed, which might cause decrease of recall, but provides more relevant results. The main drawback of most query expansion techniques that prevents these methods from being called personalized is the way how the new terms are selected. The term selection methods usually do not consider user's specific interests and are often based on static information, like lexical affinities [16] (word co-occurrences) or thesauri [17], resulting in the same terms being presented to different users. Only few approaches try to personalize the expansion terms, like [18], where the expansion is based on analyzing user's desktop documents. Relevance feedback [19] is using query refinement to improve search results, but it can be seen more like an approach to capture extremely short user's context. It requires the user to formulate a good-enough query first and select few relevant documents and as such is not practical in situations when the user does not have knowledge about the problem domain.

In our approach, we use query refinement to expand the original query with additional keywords. User does not need to have deep knowledge of the problem domain, as the additional keywords are selected automatically, by matching her preferences with preferences of other users, whose interests are similar. This also means that the generated expansions are different for different users.

Any approach to search personalization must first build a user model which captures user's interests which can later be used to personalize the search. The interests are often modeled using tags [20] and other metadata extracted from visited pages. Sometimes, tags are even used to infer other characteristics, e.g., [21] use tags to not only model user's interests, but also to infer the magnitude of the interest in the tagged document (the more tags user applied

to a particular document, the more interested she is). The majority of research focused on metadata-based user modeling is however based on the assumption that it is the user herself who annotates documents with metadata and contributes to her own model. In our work we capture the metadata automatically, similarly to [22].

This approach brings the challenge of how to capture enough of the user's activity to build a rich user model. Two major approaches are widely used to tackle this problem. First is to use an ad-hoc, specialized browser (e.g., [23]), created just for the purpose of capturing user's behavior. A similar approach is to extend existing browsers by means of plugins (for instance in [24]). Both approaches are not very suitable for a real-world usage. The specialized browser is often limited in functionality and forces user to abandon her old habits; usage of browser plugins is slightly less intrusive for the user, but there is still a need for installing and updating the extension, not to mention the effort that needs to be put in maintaining the extension for multiple browsers. A very rarely used approach is using a proxy server; there were several attempts like COHSE [25], Scone[b], or IBM WBI3[c], but none of these projects is currently maintained.

Search personalization achieves best results when the created user model is rich and contains large amounts of data [26]. Many times, some kind of social structures are used to enrich the user model with data from similar users, sharing the same interests [27]. The groups of similar users may be obtained in two ways – by using a real-world social network, or by building an artificial network where users are connected not by their real-world relationships, but by their other attributes, such as shared interests.

There are approaches that try to leverage explicit connections, but they are scarce, due to the redundant work that they offload onto the users (redundant from the point of view of the user). For instance [24] let the users build their own communities by letting them select other users that they think are similar.

In [28], authors experiment with different types of social groups (work, social and task based) and their effect on the search. They let each group to manually rank the results for queries and studied the intergroup agreement. While previous research shows that there is a lot of variation in what different people consider relevant, their results suggest that there is a lot of variation even for people with shared goals. Their results also suggest that the most coherent expectations can be found for task-based groups, i.e., it makes more sense to group people according to their short-term goals, rather than according to their long-term relations. Also, unsurprisingly, the members of any of the three groups can agree more often than any random people.

A recent research has shown that there may be a correlation between natural social network connections and interests. In [29], authors compare real world network (Facebook) connections with artificially created connections based on the profile similarity. They selected a small part of Facebook and crawled the site for connections and profile information. The connections in the artificial network were created by comparing the profile keywords, but in order to increase a chance for a successful match, the profile keywords were enhanced with WordNet data (i.e., homonyms, hypernyms, etc.). If the similarity between two keyword-based profiles exceeded a threshold, the link was created. They discovered a surprising similarity between the artificial

[b] Scone proxy, `http://www.scone.de/`
[c] IBM WBI3 proxy, `ttp://www.almaden.ibm.com/cs/wbi/`

and natural network. Their results are further supported by a similar study on delicious and flickr data [30].

The connection between social links and interests is further emphasised by research area of social Web search, where searching is not looked upon as a solitary activity, but an activity where engaging social element can lead to improved results. The search collaboration can be explicit, e.g., in SearchTogether [31] system, where users collaboratively lookup information, or the collaboration can be more subtle, e.g., in HeyStaks [32] system, which learns preferences of groups of users and adapts search results appropriately.

Many of the existing methods focus only on re-ranking the standard results – they do not solve the problem of actually getting the results into the list. This is usually solved with query expansion, which is often based on static information which does not accurately capture user's interest. We believe that query refinement could achieve deeper level of personalization by also analyzing the documents and behavior of similar users. We propose a method which extends and combines social networks and query refinement methods. We link the users in a social network not only by analyzing URLs of the visited pages, but also by analyzing the content features of these pages. We later use these features to capture user's current interest when she is searching, and also to provide the basis for our query refinement methods.

3 Social Context Driven Query Expansion

Our approach to search personalization is based on query expansion. The additional keywords for the query are coming from artificially built social network. The basic assumption here is that we can infer user's interests by analyzing texts of the articles she had read. We then use these interests to link users together – the more similar interests do the users have, the more strongly are they connected in the artificial social network [33]. When the user issues a query, we infer additional keywords by analyzing query history of community members, or by analyzing texts of the documents those members have read. Limiting the expansion process only on the community data is the key property of our approach that actually makes the keyword selection personalized. The communities are created by considering similar interests that means that different users end in different communities and the expanded queries are different.

The interests are modeled using various metadata (keywords, tags, terms etc.) coming from the visited pages. These metadata are not provided explicitely, and need to be automatically extracted from the texts. Although the web pages usually provide some form of metadata [34], they are not appropriate for modeling purposes. To acquire the metadata, we use a personalized proxy server platform and techniques that are more closely described in Section 4. Using this platform, we are able to uniquely identify the user, to track and log her activity on the Web and even to collect detailed browsing information – implicit feedback indicators, such as the approximate time spent on page, number of mouse moves, scrolls, etc.

3.1 Interest-based User Model

The user model is based on user interests, automatically acquired from her activity on the Web. It is defined as a hypergraph $H :=< V, E >$, with a set of vertices $V = A \cup P \cup T$, where A represents a set of users accessing the pages: $A = (a_1, a_2, \cdots, a_k)$, P represents a set of pages $P = (p_1, p_2, \cdots p_l)$ and T represents a set of terms $T = (t_1, t_2, \cdots, t_m)$; and a set

of edges $E = (a, p, t)|a \in A, p \in P, t \in T$ and $P \cap T = \emptyset, A \cap P = \emptyset, A \cap T = \emptyset$. Using this representation is advantageous, as it allows for good denormalization and allows us to track each of the vertices independently. It may seem intuitive to merge accesses and pages, but this models allows us to abstract page from access, and if the document represented by the URL (page) changes, we can create new vertex in the graph and connect it respectively.

Each vertex $v \in A$ (user's access to a document) has these attributes:

- user identifier, a unique identification of a user $u \in U$
- timestamp
- referrer
- implicit feedback indicators

Each vertex $v \in P$ has these attributes:

- checksum (used to determine whether the content on the URL changed, which would lead to a creation of new vertex. We calculate the checksum from the cleartext (the actual textual content of the document), so the documents with the same content but different ads have the same checksum)
- content length (used along with implicit feedback indicators to calculate a page interest score)

Vertices $v \in T$ (terms) that are extracted from the documents represent the conceptual level of the user model and have following attributes:

- label – the name of the term itself
- relevance – denotes how much is the term relevant to the content of the page
- type – type of the term (e.g., person, city)

The user model is periodically updated. Each document view initiates the term extraction process and the acquired metadata is added to the user model. If needed, new edges, or vertices are added to the graph (e.g. the document does not exist, or its checksum is changed).

User's interests are thus modeled with document metadata. Although the automatically acquired metadata can be of variable quality, we rely on the fact that the most important document features representing the major user's interests will be repeated several times across multiple documents and prevail the noisy metadata by the count.

The described user model serves as a basis for query expansion. This is a process which consists of three steps:

1. enhancing the user model of each user with community data
2. detecting the user's context
3. expanding the query

The user model is enhanced periodically and offline, while the context detection and query expansion happen in real-time.

3.2 Building the Communities

For each user, we find similar users, based on the similarities between their interest-based user models. We use both the term-based similarity and document-based similarity. The main reason for this enhancement is that we gain more data for our query expansion algorithms.

We start by creating a social network, where each vertex represents a user and each edge connects two users with a weight that denotes their similarity. The network is constructed in the following steps:

1. Before we start the network creation, we first consider the implicit feedback of each access, calculated as

$$X = 1 - \frac{1}{1+w}; W = \frac{time\_on\_page + click\_count + copy\_count + select\_count}{content\_length}$$

   We do not weight the indicators, although some of them may have bigger impact than the others. As the Figure 1 shows, the distribution of X exhibits the long tail property, with the majority of Web pages with the value of $0 < X > 0.1$. We therefore prune all accesses with the value of $X < 0.1$

2. New network is created, each user is connected with each other by an edge of weight 0

3. For each unique visited domain (as in the domain system of the internet)

   (a) If any two users visited the exactly same URL (we also consider URL parameters, as a change in a parameter could result in a different page), the weight w of the edge e connecting them is increased by a value of parameter d.

   (b) Otherwise, the weight w of the edge e connecting them is increased by a value of parameter s.

   Generally, $s < d$ as viewing the exactly same document implies more similarity than just viewing a document from the same domain (e.g., if both users read the exact same newspaper article vs. if both read some article on the newspaper portal). We use en empirically derived value of $s = 1$ and $d = 0.5$.

4. For each pair of users, an overlap of their term-based models is calculated, and the weight w of the edge e connecting them is increased by the cardinality of this overlap; i.e., we aggregate all terms from the user model of the first user into a set T_{u_1}, and similarly, for the second user into a set T_{u_2}, then the weight is increased by $card(T_{u_1} \cap T_{u_2})$.

In order to avoid repeated calculation of the whole network, we do not build a new network, but rather evolve the existing one by incremental updates. When updating, we do not consider all accesses, only those, which were added after the last network update. We create a network using the described algorithm and merge it with the existing network. Let u_i and u_j be the users, $w(u_i, u_j)$ be the weight of the edge connecting u_i and u_j in the new network, and $w'(u_i, u_j)$ the weight of the edge connecting u_i and u_j in the old network, then the weight of the edge in the merged network is $w_n(u_i, u_j) = p.w'(u_i, u_j) + w(u_i, u_j)$ and p is the decay factor, representing the loss of preference. We use an empirically derived value of $p = 0.9$.

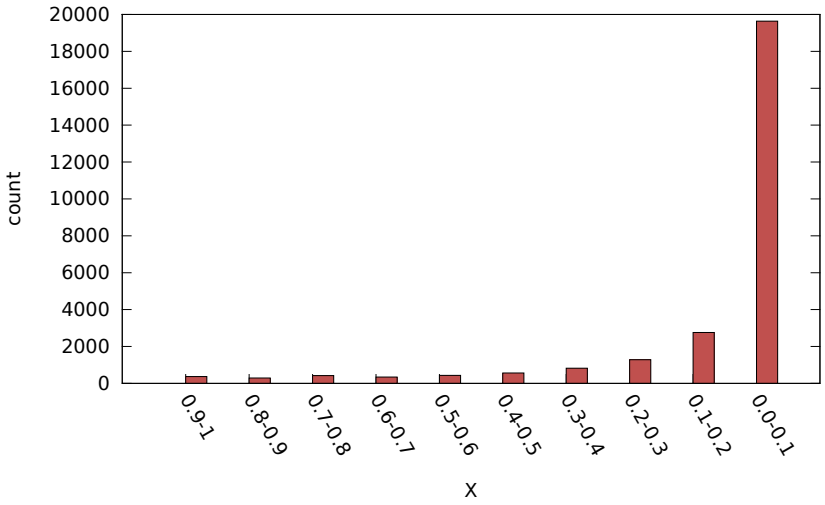

Fig. 1. Distribution of the implicit feedback values.

In the next step, we detect communities of similar users by applying a spreading activation algorithm over the weighted social network. We randomly select a vertex (user) not yet assigned to any community which is activated and the activation spreads over the weighted edges to other vertices. All activated edges are considered part of one community. Similarly to [35], we do not detect communities for all vertices, since most of them are part of the same communities, but use a following algorithm:

1. select a random vertex
2. spread the activation, create a community from all activated vertices
3. select a vertex, not yet assigned to any community and continue with step 2
4. exit when all vertices are linked to a community

Each community is stored and represented as a set of its member users.

3.3 Context Selection

Given the communities of similar users, the method now proceeds with the context selection. Some basic assumptions have to be established in order to disambiguate the query: who is the user, what is her intent, her informational goal.

It is possible to use two approaches:

1. Long-term context, which captures all of user's interests. This is generally a simple process, as we just use all the collected data and the communities build from all the data to serve as the basis for our query expansion methods.

2. Short-term context, which captures user's current interests. In order to use a short-term context, we need to split the user's browsing into episodes of activities with a single underlying informational intent, referred to as browsing sessions [36]. Given these browsing session, we may proceed to:

   - build communities just from the data from currently active browsing session
   - select existing long-term communities, by matching community metadata (i.e., the aggregation of metadata from the user models of users who belong to the particular community) with the session metadata (i.e., the aggregation of metadata from each document from the session) and selecting the communities with best overlap. This approach is more viable than the first one, as it does not require the time-consuming social network creation.

Using the long-term context is generally simpler, while the short-tem context is more focused, but may not contain enough data to generate accurate query expansion. It has been shown that the best approach is to combine both long and short-term context [37] but the exact effects are still subjects to research. In our work, we utilize only the long-term context.

3.4 Query Expansion

Outputs of the previous stage are the social networks the user belongs to and they serve as the basis for our query expansion strategies. We use two types of query expansion:

- Query reformulation analysis
- Metadata co-occurrence analysis

3.4.1 Query Reformulation Analysis

Query reformulation analysis is based on a simple observation of how we do our searches: if a particular query does not satisfy user's needs, it is reformulated. We analyze the logged accesses and find query streams: time-sorted sets of queries which were issued by a single user and share at least one keyword, e.g., *jaguar; jaguar os; jaguar mac*.

When the user starts searching, we find all query streams issued by the users from her community which contain the searched keyword(s). We analyze each query stream, and assign it a rank that is calculated as:

$$X_{qs} = \sum_{d=1}^{D} \frac{X_d}{|D|}$$

where D is a set of documents which were accessed from the search engine result page for that query and X_d is an implicit feedback for that document.

We select the query stream with a maximum rank and use its last query as a refinement. Involving the implicit feedback is important, as the fact that user entered the query does not mean that its results were useful. Considering the implicit feedback allows us to prune queries for which no result was clicked, or for which the results were not deemed useful. Table 1 contains some of the results of this strategy when applied on the AOL dataset.[d]

[d] AOL search engine logs, `http://www.gregsadetsky.com/aol-data/`

Table 1. Examples of queries and their redefinitions.

Original query	Expanded query
Java history	history of Java Indonesia
jaguar	jaguar animal
sphinx	sphinx cats

The main strength of this algorithm lies in the underlying communities. They ensure that the expanded queries are relevant to user's interests. For example when a user is searching with a query "jaguar", there are probably many query streams; some of them lead to a query "jaguar animal", some of them lead to a query "jaguar mac" or "jaguar battle tank" but if we limit the search only to user's community, the uninteresting redefinitions are pruned away, as they are issued by users with different interests and thus in different communities.

3.4.2 Metadata Co-occurrence Analysis

This method is based on the idea that we may refine the meaning of a keyword by looking at all documents where it occurs and select the words that it co-occurs with most frequently. We also expect that the co-occurrences, or keyword pairs, will differ in different interest-based communities, e.g., in a community composed of car enthusiasts, the most frequent co-occurrence of keyword "jaguar" may be "jaguar car", while in the community of animal enthusiasts the more frequent co-occurrence may by "jaguar animal". Table 2 shows examples of documents about the jaguar – mammal and a sample of their metadata. We see that the pair "jaguar" and "animal" is present in all of them, and that "animal" is the most frequent co-occurrence of "jaguar".

Table 2. Documents about jaguar and their respective metadata.

URL	Metadata
`http://en.wikipedia.org/wiki/Jaguar`	university, **animal**, culture, dictionary, online, name, symbol, retrieved, warrior, **jaguar**
`http://www.essortment.com/all/jaguaranimalca_rhvk.htm`	**animal**, **jaguar**, prey, threat, search, jungle, area, cat, body, water
`http://www.tropical-rainforestanimals.com/Jaguar-Animal.html`	**animal**, prey, name, web, page, territory, picture, forest, story, **jaguar**
`http://www.buzzle.com/articles/jaguar-the-rainforest-animal.html`	coat, **jaguar**, rainforest, mating, rosette, world, panthera, prey, **animal**, body

When the user enters a query, we search all documents accessed by the users in her community and search for the most frequent co-occurrences. Similarly to social network creation process, we prune all documents with implicit feedback score lower than 0.1.

Some samples of the redefinitions provided by this strategy are shown in Table 3.

Table 3. Examples of query redefinitions.

Original query	Expanded query
passenger	passenger apache
branch	branch git
apache	apache server

Similarly to query stream analysis, this strategy leverages the interest-based communities. The co-occurrences of the same keyword may be different, but grouping the users into clusters of similar users helps to distribute the differences into separate groups.

4 Capturing User's Interests on the Open Web

Given the disadvantages of using browser extensions or using custom browser, we designed and implemented a specialized, personalized proxy server that we use to capture user's activity on the Web. The proxy server has access to the data (unless it is encrypted) that is exchanged between client and the server and can modify it. Using this feature, we are able to modify existing pages and provide a personalization layer. Unlike browser extensions or custom browsers, using a proxy server is advantageous both for the user and developer. There is no need for the user to install anything, or to change her browsing habits by using a new user interface. The developer does not have to maintain and improve several versions of the same extension for different browsers. Moreover, proxy servers are widely used in corporate environments, where making the interaction with Web more efficient can bring many benefits, for instance personalized access to a company knowledge base, or search results that are adapted towards user's current work task can decrease the time that is needed to find an information and thus make the user more efficient and less prone to distractions.

Generally speaking, the primary motivation behind our personalized proxy server is to create a user modeling platform and move the personalization methods from server-side towards the user. Nowadays, many personalized systems exist, but they are scattered around the Web and build their own proprietary user models. The particular personalization methods do not usually have access to other data about the user than that which was collected via her interaction with the narrowly focused service (e.g., the level of knowledge in an educational system, or user's reading preferences in a book shop). Once a user model is created, other systems could benefit from this knowledge. There are relatively few attempts to reuse the user models in multiple adaptive applications in form of user modeling servers (e.g., a generic modeling server GUMSAWS [38], or MEDEA [39] in the domain of adaptive learning), but they are not widely spread. The main problem that is still not satisfactorily solved is how to acquire the user modeling data from various sources and applications, even those that do not support adaptation or explicitly collect user data. A proxy server is able to tackle both sides of the problem – by being a gateway to the Web it can collect data from any web application, site, or document and on the other hand, it can modify content of any page before it is received by the client and thus bring personalization to any part of the Web.

4.1 Personalized Proxy Server

The standard proxy server serves as a forwarder of messages sent between user's Web browser and the remote server. Although many free or commercially used proxy servers are able to do simple modifications to the request/response chain (e.g., remove ads, deny access to resources etc.), their abilities are usually limited. Our personalized proxy server was designed from the ground with the idea that any part of the proxy (extension, or a plugin) should be able to modify any part of request or response data. We will furthermore refer to either request or response as message. This enables the implemented methods to modify the structure of the messages, modify the transported HTML code and personalize the whole browsing

experience. It is possible to provide the user with new services, extend existing systems, exploit her knowledge of existing interfaces and unobtrusively use her browsing customs.

Our personalized proxy server is based on the existing proxy server Rabbit.[e] We improved Rabbit with an extension system, which allows capturing and modifying user's requests and server's response. It is possible to create extensions of two types:

- *Services.* Services provide a generic method for manipulating the message data. Each service represents a single well-defined and reusable module that performs a specific action over the message. Examples of services range from simple ones, such as creating a textual representation of the message from the underlying binary data, to more advanced, which provide access to a document object model (DOM).
- *Processing plugins.* Processing plugins combine and compose the available services to create a process which results in the modification of the underlying message. For example, a processing plugin may use a DOM Manipulator service to insert new results to a search engine page, and the DOM Manipulator will internally use a String service to convert the bytes to its textual representation.

The incoming message is then passed through a series of processing plugins, each taking a specific action based on the message, or over the message. Of course, not all registered processing plugins have to be run all the time, e.g., if the Search Results Modifier plugin detects that the underlying message is not a search engine results page, it will yield the execution to the next plugin in chain. The whole process is graphically described in Figure 2.

4.2 Capturing the Interests

In order to gain a good understanding of user's interests, we track user's activity on the whole Web and model the interests as various metadata automatically extracted from the viewed document. The process of capturing user's interests is described in Figure 3.

User's browser needs to be set up to communicate with Web servers via the proxy server. This means that each request is routed via the proxy (in practice, this can be, and is, controlled by the user so some level of privacy can be achieved). Here, it can be modified, or other actions may be taken (e.g., log) and the request is sent to the target server. The server responds and the response is again routed via proxy, where it can be modified, exchanged or other action may be taken. For the purposes of tracking user's activity, we extract and store the document metadata and enhance each response with a tracking JavaScript that asynchronously logs the visit (this approach is used to override browser caches) and reports user's activity on the particular web page.

The processing of each document can be basically broken down into these few steps:

1. Identify the user
2. Log the document access
3. Extract and log document metadata
4. Collect and log implicit feedback signals

[e] Rabbit proxy server, `http://khelekore.org/rabbit`

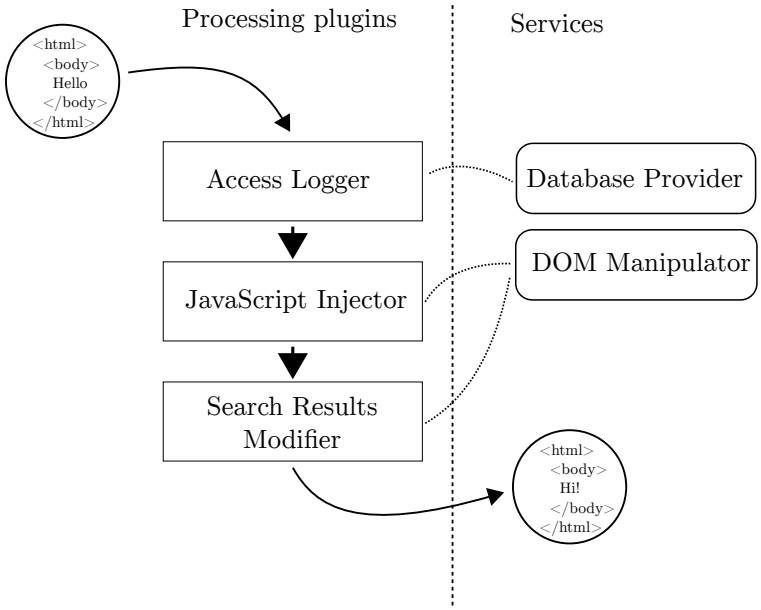

Fig. 2. Example of message processing. The message comes in and passes through a series of processing plugins. First, Access Logger logs the message, then JavaScript Injector modifies the HTML and inserts a JavaScript code and at last, Search Results Modifier injects or removes search results. Each processing plugin uses a set of services to accomplish its job.

Usage of the proxy server to monitor user's activity on the Web raises many privacy concerns and we have tried to make our proxy platform as open and transparent as possible, while actively educating our users about privacy control. First, we provide instructions about how to set up the proxy server in the browser in a way that the user is in control of which Web sites are accessed via the proxy and which are accessed directly via their internet connection provider. Browsers provide convenient tools to make this kind of setup easy, i.e. tools that allow users to specify whitelist or blacklist URL patterns. We have also built a Web site, where users could see everything that was logged about them and selectively delete records. We have provided an option to opt-out completely, by immediately deleting all user data.

User Identification

The first challenge in the data acquisition process is to unambiguously identify a user [40]. This is usually solved using IP addresses, but it is often difficult and inaccurate, as multiple users may share the same IP address if they are behind a NAT router, or using another proxy server. Sometimes advanced heuristics are used, such as User-Agent HTTP header field analysis, but these are still prone to errors. Given the flexibility that our personalized proxy offers, we use a completely different approach.

- We set an HTTP cookie with the uniquely generated user ID on the domain that we own and insert a reference to a bogus JavaScript on that domain to every HTML response.
- Browser manages a set of domain-constrained HTTP cookies and when it makes requests

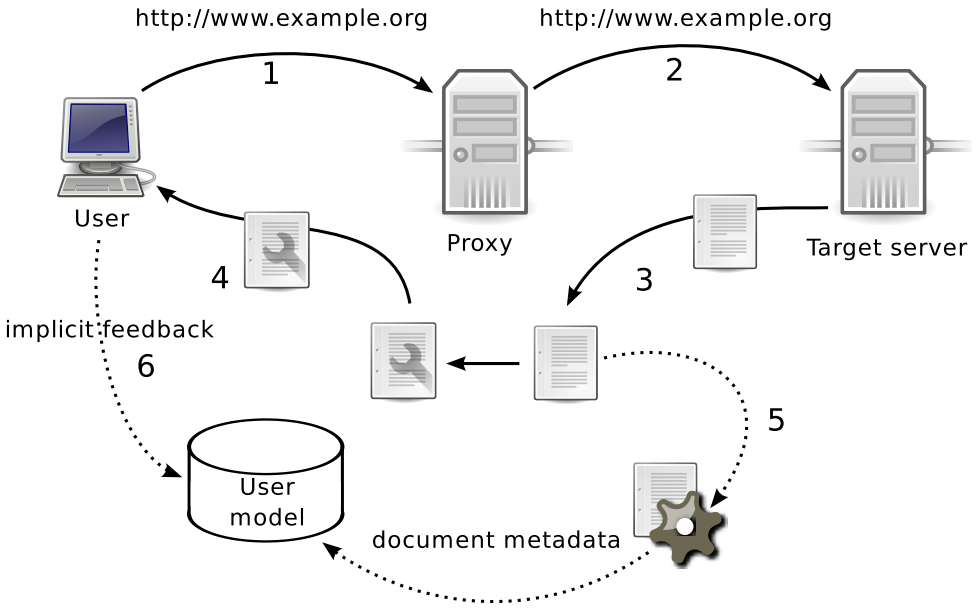

Fig. 3. Data acquisition process. User's request passes through the personalized proxy server (1) and it is forwarded to the target server (2). The server's response (3) arrives at the proxy, where it is modified and a tracking JavaScript is injected. The modified response is sent back to the client (4). Meanwhile, the document is processed (5) and various metadata are extracted and stored in the user model. On the client side, the tracking JavaScript periodically updates the user model with implicit feedback data (6).

to a domain, it sends its corresponding cookies along.

- When the response is processed in a user's browser, it sends request to the bogus script, along with user ID cookie.
- This request also passes through the proxy, where it is intercepted, the user ID is parsed from the cookie and a bogus response is generated. The request is never sent to the target server.
- The generated response contains JavaScript which is automatically evaluated in the user's browser and sets a global variable containing the user ID.

Cookies are a vastly used technology [41], but of course, there are some risks associated with its usage. If the user deletes the cookie, or maybe uses a different browser, her identification is lost. We solve this problem by using EverCookie[f]– a JavaScript code that can store the cookie in various storages (even storages outside the browser) in user's computer. If the cookies are accidentally deleted, EverCookie searches its storages and if it finds the cookie, it is automatically restored to each of the available storages. This also means that if the user runs another browser, EverCookie will still be able to find the cookie from the original

[f]EverCookie, `http://samy.pl/evercookie/`

browser in a non-browser storage. Of course we do provide users with a mean how to really decouple their computer with their anonymous proxy identity.

This solution works for reliably separating different users and maintaining a consistent log of user action for the whole duration of using the proxy server, however, this does not address the problem of disambiguating users sharing the same computer. This problem is outside the scope of this work, but there are solutions that involve biometric data or behavioral patterns that could be possibly used to address this issue.

Logging Document Access

Second, we need to actually log the fact that the user accessed a document. The commonly used server-side logging fails if the user hits browser cache, most commonly when hitting a back button or accessing an aggressively cached resource. To avoid this type of problem and to capture a complete trace of user's activity, we inject a logging JavaScript into every accessed page. It is evaluated on every page display (even on cached pages) and waits until the global variable containing the user ID is set (described in previous paragraph). Afterwards, it issues an asynchronous request, which is again intercepted by the proxy and the access is logged.

We log:

- user identification
- URL of the page
- referrer – this denotes a previous page that contains a clicked link to this document
- content length – number of characters in the text
- timestamp of the access
- document metadata
- document checksum

User ID, URL and referrer are sent in the asynchronous request, other attributes are filled in on the proxy side.

Metadata Extraction

In order to extract the document metadata, we run the message through a series of following steps. First, the message is stripped of the auxiliary content: ads, menus, header, footer etc., until only the main content remains (e.g., the text of the newspaper article). We further refer to the cleansed content as cleartext. We do this by reimplementing the publicly available Readability script[g] which is in essence based on counting commas in paragraphs. The basic presumption is that the main content contains higher amount of commas than ads, or menus. This idea works best on pages which contain continuous text, while usually fails on homepages or various listing (e.g., newspapers title page). In that case, we use the original HTML.

In the next step, the extracted cleartext is translated to English. This is due to the fact that the metadata extraction methods in general work best on English content. We use Google

[g]Readability, `http://lab.arc90.com/experiments/readability/`

Translate,[h] which is able to automatically detect the source language. The translated content does not have the same quality as the original content, but the main characteristics and the semantic information are preserved and the extracted metadata are of sufficient quality [42].

We use the following types of metadata:

- *keywords*; a one-word long metadata, which characterizes the main ideas/information of the document it was extracted from, e.g., *jaguar*, *animal* or *forest*
- *terms*; a multiple-word long metadata; it characterizes the main ideas/information of the document it was extracted from, but its meaning is more narrow than that of a keyword. A term combines multiple related keywords, e.g. *city hall* or *fire department*
- *named entities*; are a special category of terms. They name any of the real-world entities, like persons, cities, countries, etc., e.g. *Abraham Lincoln* or *Mississippi*
- *tags*; characterize the main ideas/information of the document, but unlike keywords or terms, they were created manually by humans and although sparse, they are very accurate

We run the translation through various publicly available metadata extraction Web services and standalone libraries:

- **JATR** (`http://www.dcs.shef.ac.uk/~ziqizhang`) – Java Automatic Term Recognition Toolkit implements some of the traditional algorithms for keyword extraction: TF.IDF, CValue, TermEx, GlossEx and Weirdness
- **Alchemy Orchestr8** (`http://alchemyapi.com`) – a Web service which provides named entity and term extraction
- **OpenCalais** (`http://www.opencalais.com`) – similarly to Alchemy, provides named entity and term extraction
- **tagthe.net** (`http://tagthe.net`) – a simple Web service for keyword and term extraction
- **Delicious** (`http://delicious.com`) – provides tags, created by its human users. The tags are therefore very accurate, but the disadvantage is that they are available only for a small subset of all documents
- **DMOZ** (`http://dmoz.org`) – similarly to delicious, provides human created metadata – Web page categories

Collecting Implicit Feedback

The extracted metadata should not be considered equal. There might be cases when the user opens the document, but quickly realizes that the document is not interesting. In order to detect how much did the document interest user, we need to capture interest indicators. It is possible to ask user explicitly, but that might be considered disturbing. A much better

[h] Google Translate, `http://translate.google.com`

alternative is an implicit feedback, derived from automatically collected signals. Often, a time spent on page is derived from the server logs and used as a primary feedback indicator.

However, the time mined from logs may not be accurate. The commonly encountered problem is a Web page opened in a background tab, for which the real time and derived time differ greatly. To cope with this problem we use a more accurate approach. We inject a tracking JavaScript into every document that the user requests. This script monitors user's activity on the page and tracks the real time user spent on the page. Currently, there is no direct way to detect that a page is active, so we use a slightly complicated approach. The script monitors short time windows (we used 4 seconds) for activity. If there is some activity (mouse movement or scrolling) in the time window, the length of the window is added to the estimated time. While even this approach is prone to inaccuracies, the time spent on page calculated this way is certainly more accurate than the time estimated from server logs.

To further evaluate the usefulness of the page, we collect additional interest signals:

- estimated time on page
- number of clicks
- number of scrolls
- number of copying into clipboard
- number of text selections

Using the proxy platform is an improvement over existing approaches as it allows for both data collection and adaptation of any web page that the user accesses. There are however few problems and disadvantages of this solution. First and foremost, the set up process is still not completely automatic and requires intervention of the user, who has to set up the browser. This is not a problem in corporate environments, but represents an initial barrier for individual users. There is also problem with the proxy server being a central place to access the Web – it is prone to outages. Even though it is relatively simple to automatically instruct the browser to stop using the proxy if it is not responding, it is still a good idea to run multiple proxy server instances. Multiple instances are required to serve more users and they should be placed geographically close to its users to reduce latencies. Many existing services also depend on IP geolocation to control some aspects of their interface (for instance localization) so using a geographically close proxy server is preferred.

Nevertheless, using proxy server to collect data about users and provide personalized versions of existing systems is advantageous, especially in the research in the domain of search, where the search engine itself does not have many opportunities to get to know user's preferences. We also believe that this architecture is sustainable in long-term. Proxy servers have become standard architectural piece of the internet and they are widely used especially at companies. Deploying this kind of personalizing proxy server in a company could bring many benefits to its users.

5 Evaluation

We evaluated our approach to query expansion on real users, using the described proxy server. We have advertised the proxy server throughout our faculty and invited students to try it.

Obviously, each of the methods will work best if the queries are short and ambiguous and as most of the user studies report, this is generally true. Since the users of our proxy server are all students of Computer Science and as such are well aware of how search engines work, we conducted a user study to see if there is a room for improvement in their queries.

5.1 User Study

The proxy server was deployed on 23rd March 2010. Following a short, private testing phase, it was publicly announced on 7th April. On the day of evaluation (4th May), it had been used by a total of 40 users. The Figure 4 shows the number of users from the first proxy server deployment to 29th April. After this day, the number of users remained static.

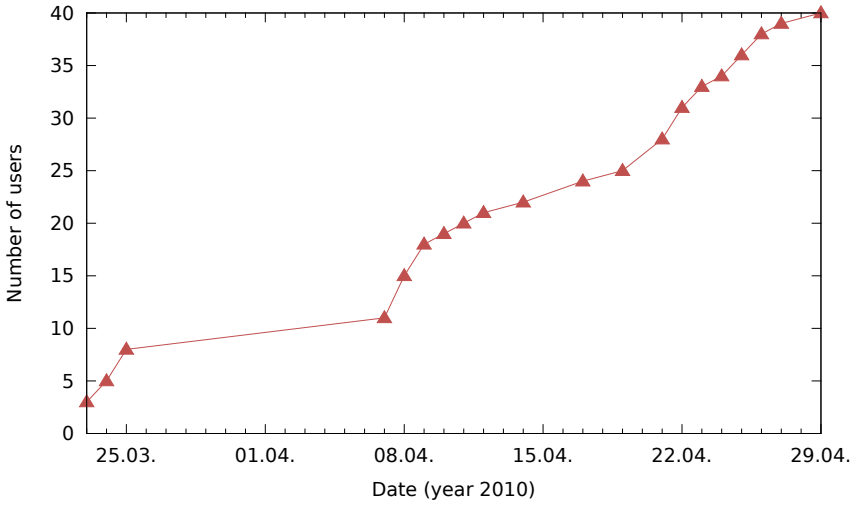

Fig. 4. Usage of the proxy server.

During this time, users have accessed 643 939 document (111 215 of them unique). But the proxy has never been used by all users at the same time. Figure 5 show the usage in the weeks following its launch.

In total, 3 637 queries were issued in various browsers. Table 4 summarizes the shares of each search engine found in the logs. Practically all of the queries were entered into Google search engine. Numbers of searches in Yahoo or Bing are insignificant and other popular search engines (e.g., AOL, Wolfram Alpha or Cuil) were not present in the logs.

Table 4. Number of queries per search engine.

Search engine	**Number of queries**
Google	3628
Yahoo	4
Bing	5

For the purposes of query disambiguation, the most interesting information is how long a typical query is. Figure 6 shows the query length distribution. Our findings are consistent

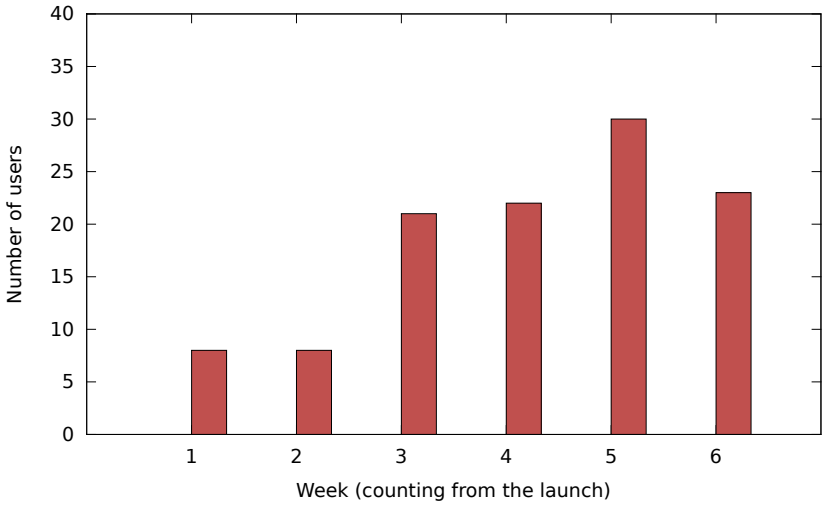

Fig. 5. Usage of the proxy in the weeks after its launch.

with that of other studies; we confirmed that even the technical users issue short queries.

We will now focus on Google as it is the search engine where majority of the searches takes place for our users. From all issued queries, users clicked 6 152 results. From this number, only 27.4% (1686 results) were viewed for more than 4 seconds. In other cases, the page was abandoned after less than 4 seconds after opening. We assume that these were the cases, where the clicked result was not relevant to user's information needs, what means that only about every fourth clicked search engine result is considered useful.

We were also interested if longer queries mean that users get more satisfactory results. Figure 7 shows the fraction of useful results (i.e., results that were displayed for more than 4 seconds) with respect to query length. The results do not indicate that longer query means better search results; while short queries are too ambiguous, long queries are too specific, making it hard to find the relevant documents.

5.2 Real Users Experiment

Our method for social context driven query expansion was deployed on the personalized proxy platform. We integrated the query expansion into Google search engine. During the experiment, if the method generated any query recommendations, they were shown among the regular results. We limited the maximum numbers of query recommendations to 4 to not overwhelm the results page and confuse the user. The expanded queries could be generated by any strategy, but we strived for balance, so that if possible, both strategies (query stream analysis and metadata co-occurrence) generated the same amount of expansions. For each expansion, we queried Google using its standalone API[i] and retrieved a set of top results. These results were shown on the results page together with standard results. Again, we strived for balance, so each expanded query provided the equal amount of search results.

[i] Google Web search API, `http://code.google.com/apis/customsearch/v1/overview.html`

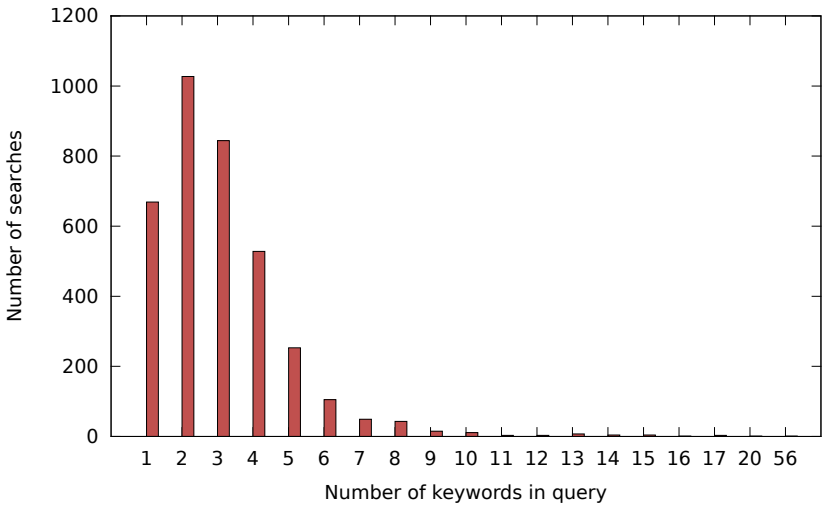

Fig. 6. Distribution of query length.

Using this technique, we assembled a set of 4 results, randomly ordered, and injected it into the standard search results page.

Even through the course of the experiment, our main goal was to bring value to our users in the first place. We were concerned that if our approach did not work and the (bad) results were mixed with organic results, our users would stop using the proxy server. Therefore we have decided to separate our results from the organic results and present them as recommendations. The expanded queries and results were clearly separated from the standard results. Figure 8 shows the search engine results page with the expanded queries.

Users could click either the search result, which would take them directly to that result, or they could click the expanded query, which would start a new search with the expanded query. We also incorporated an explicit negative feedback, by showing an icon of a trash can next to query expansions as well as next to each search result. Users were aware that clicking this icon sends a signal that either the result or expansion are incorrect. Clicking the trash icon next to a search result would remove that result from the list, while clicking the trash icon next to an expanded query would remove all results retrieved from that query.

The experiment was run in two sequential stages:

1. the *partial query expansion*, which could be generated by considering only a part of the query, e.g., if the user entered a query “jaguar amiga”, the query could be expanded by only considering the word “jaguar” or “amiga”.

2. the *whole query expansion*, which had to be generated from complete query. If a user issued query “jaguar amiga” then both of the keywords had to be present in the query stream for the query stream analysis strategy, or in the document metadata for the metadata co-occurrence strategy respectively.

Table 5 summarizes the results of both stages. The main metric we used was number of

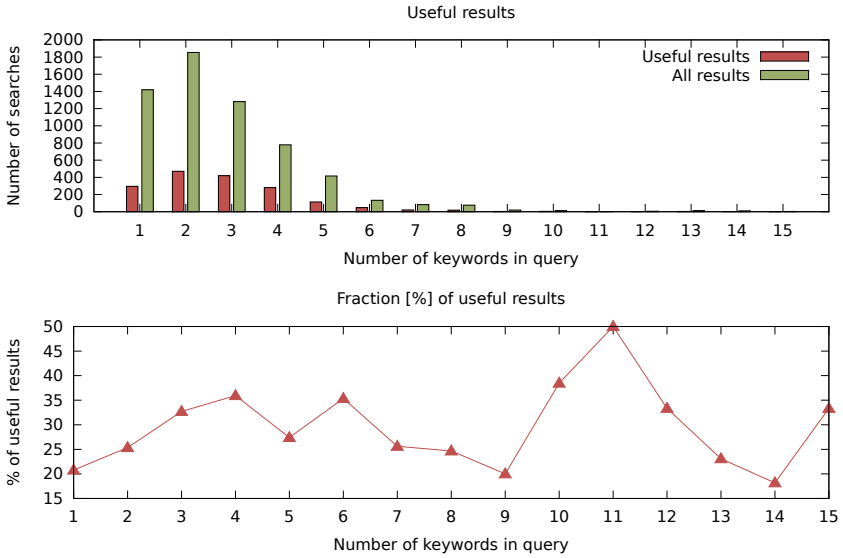

Fig. 7. Useful results by query length.

successful expansions. The expansion was considered successful if either the expanded query, or some of the expanded results were clicked. Note that multiple results might have been clicked for a single expansion, so (# of expanded queries clicked + # of results clicked) may differ from # of successful expansion.

Table 5. Summary of results from both stages of the experiment.

	Partial query	**Whole query**	**Whole query (considering duplicate results)**
# of expansions	311	95	95
# of results	1230	377	377
# of queries clicked	12	9	9
# of results clicked	35	14	63
# of successful expansions (i.e. query, or a result from this query was clicked	43	21	67
% of successful expansions	13.8%	22.0%	70%

Expansions based on the whole query are more successful. After empirical evaluation of the collected logs, we believe that this is caused by long queries. If the query is short, then both strategies generate logically correct expansions. However, as the queries get longer, strategy based on whole query is usually unable to expand the query, while the partial strategy picks up and expands a word and generates even longer and meaningless queries. As the *whole query expansion* strategy proved to be better, we will now focus on analyzing it further.

Table 6 shows the comparison of the used expansion strategies. There seem to be no difference in the number of successful expansions they generated, although the metadata co-occurrence strategy generated more expansions. This is expected, since the query stream

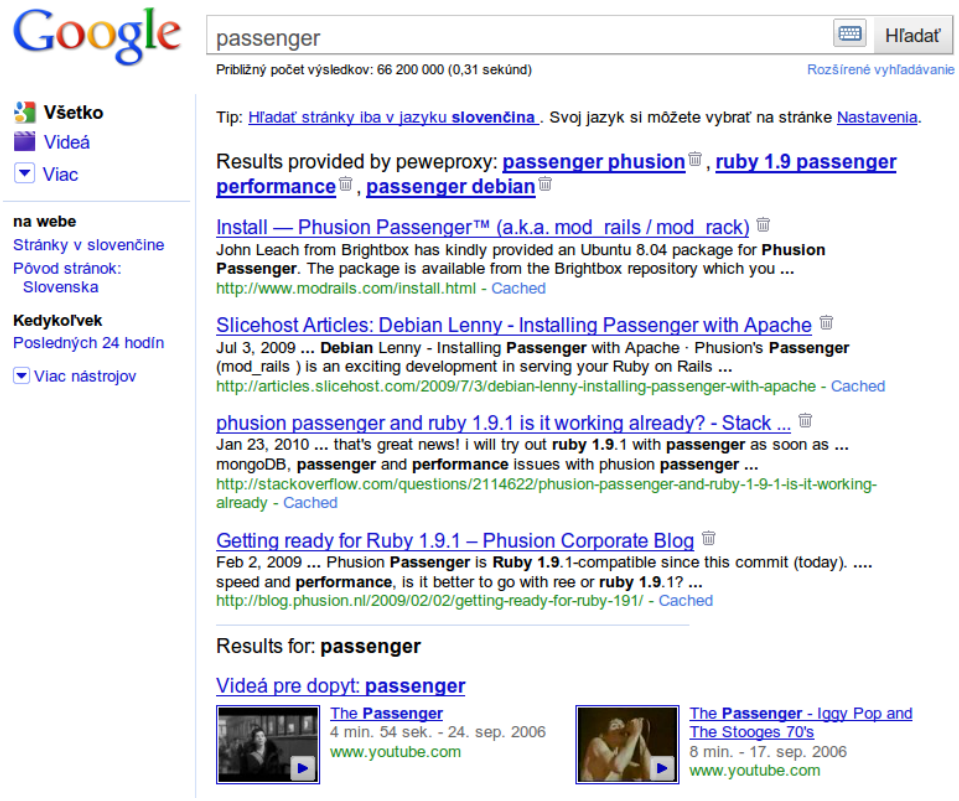

Fig. 8. Screenshot of the Google search engine with the experiment in progress. On top, the expanded queries are shown, followed by search results retrieved by querying Google with the expanded queries. The injected results are clearly separated from the standard results by a label and a separator line.

analysis requires that a similar query was issued before by a member of the community which is a condition that limits its applicability.

This kind of experimental setup raises a concern that by separating the recommended results from the organic results, we have created a bias, because according to the studies [43], people tend to click on the top results, disregarding the actual relevance of the results and deciding solely on the result position. We argue that this was not the case, because we have observed different kind of phenomenon.

According to the post-evaluation survey and personal interviews, users tended to ignore the recommendations and followed the standard stereotype of perceiving new interface elements and at first scanned the standard result set and proceeded to the expanded result only if the standard results were unsatisfying. This claim is further supported by the data collected during the live experiment. When generating the expanded results, we did not check the standard results, so, many times, the expanded result was already present among the standard results. We have looked at all the cases when the result was present in both the recommended results and organic results groups and looked which group was the clicked result from. We

Table 6. Comparison of the expansion strategies.

	Query stream analysis	**Metadata co-occurence**
# of expanded results	133	1474
# of clicks on expanded results	5	44
% of clicked results	4%	3%
% of useful results	66.6%	53.8%

have found that when the same result was present in both groups, in 81% of cases, the result was clicked from the organic results group, supporting the claim, the users scanned the organic results first.

If the same result is among both organic and expanded results and we consider clicks on the organic result as a success for the query expansion method which generated it, then the success rate increases from 22% to 70%.

Most importantly, if we look at the usefulness rate, i.e., how many pages were opened for more than 4 seconds, it's 54.7% for our expanded results, what is an improvement compared to only 27.4% for standard results.

5.3 Qualitative Analysis

To further asses the obtained results from live experiment we have manually analyzed each query and the set of clicked results. We built a list of all queries for which some recommendations were generated together with all the possible actions that the user could take, i.e., click an organic search result, proceed to next page in the search results list or manually refine the query. We have then asked a human judge to asses the relevance of the recommendations for each query. It is not fully possible to judge the relevance of the recommended query reformulations externally, since the judge does not have the same context as the user at the moment when the query was issued. However, the intent can be manually inferred from user's actions on search results page. We have asked the judge to carefully review user's activity, read the pages that the user clicked, follow through the query reformulations and carefully assign one of the relevance labels to each recommended query reformulation according to these rules:

- *perfect* - the recommended reformulation fits user's intent perfectly. The user either manually reformulated the query to one of the recommended queries or clicked an organic search result, which was also included in the list of recommended results (retrieved by issuing the recommended queries);
- *satisfying* - the recommended reformulation fits user's intent. The activity on the search results page and the content of the clicked organic results suggest that the recommended reformulations were correct and matched user's intent at the query time;
- *poor* - based on the activity on the search results page and the content of clicked organic results, the recommended query reformulations do not match user's intent;
- *cannot judge* - based on the data available, it is not possible to confidently assess the relevance of recommended reformulations.

The judge's assessments are shown in Table 7. The data shows that 63% of the reformulations passed the *satisfying* boundary, while 34% of the reformulations were marked as poor.

Upon further examination of the logs from live experiment and the manual assessments, we have found out that the reformulations marked as *poor* were generated solely by the metadata co-occurence approach, where the query was expanded by a word that was too generic.

Table 7. Qualitative analysis of the query log.

Assessment	% of queries
perfect	34%
satisfying	29%
poor	34%
cannot judge	3%

6 Discussion and Conclusions

In this work, we focused on search disambiguation by expanding the short and usually ambiguous queries with other keywords, based on the user's interests. We focused on leveraging social networks as a mean to capture similarities between users. Our primary hypothesis was that similar users visit similar pages and view similar documents and that by using the metadata of the documents we will be able to group similar users together.

Our method relies on the similarity-based social networks. We designed a method to build this social network from the stream of users' activity, a method to weight the strength of relationship and to mine the virtual communities. Tracking the users' activity is possible mainly because of our personalized proxy server platform, which we used to both track users' activity and to seamlessly integrate our method into an existing search engine to conduct a real user evaluation.

We build the user model by analyzing user's activity on the "wild" Web and model her interests using the metadata (keywords, terms, tags) automatically extracted from each visited page. We use a combination of automatically captured metadata with human created data (tags from delicious and categories from ODP).

We designed and evaluated two query expansion strategies, first based on the observation that after an unsuccessful search the query will be reformulated and the second, based on the observation that the keyword meaning can be refined by looking at the document metadata; keywords it frequently co-occurs with. Using the described proxy server platform, we integrated our query expansion method into Google search engine and injected our expanded results next to the original results. We noticed that in 70% of the searches where expansion were generated, some of the expanded results were clicked and furthermore, we observed a significant increase in the relevance metrics of the expanded results in comparison with the standard results. A clicked result was considered useful if its dwell time was larger than 4 seconds. In our experiments, only 27% of standard results were considered useful in the baseline environment, while the results expanded by our method were considered useful in 54% of all cases. A post-hoc analysis performed by a human judge revealed that about 63% of all recommended reformulations matched user's intent.

There are many factors that affect the quality of expanded queries:

- The method is sensitive to metadata quality. When the metadata vectors contain commonly repeated elements, it will impact the expanded queries for the metadata co-

occurrence strategy.

- Our method does not deal with semantic similarity of the metadata. Metadata are considered similar only if they match character by character. The performance of the method could be improved by including a more sophisticated matching algorithm, e.g., enhancing the metadata by related concepts (from ConceptNet[j] or Wordnet[k], or by using their hyponyms and hypernyms [44]. For example, the non-overlapping metadata vectors *{java}* and *{python}* may be enhanced by their hypernyms *{java, programming language}*, *{python, programming language}* and they would now overlap.

In the next work, we will focus on improving the metadata quality – both by automatically analyzing the extracted keywords to remove noise, and by enhancing them with hyponyms, hypernyms and related concepts. We also plan to explore the possibility of using short-terms contexts, instead of currently used long-term contexts.

Acknowledgements

This work was supported by grants No. VG1/0675/11, APVV 0208-10 and it is a partial result of the Research and Development Operational Program for the projects Support of Center of Excellence for Smart Technologies, Systems and Services I and II, ITMS 26240120005 and ITMS 26240120029, co-funded by ERDF.

The authors wish to thank colleagues from the Institute of Informatics and Software Engineering and all students (members of PeWe group, `pewe.fiit.stuba.sk`) for their invaluable help in experimental evaluation of the work presented in this paper.

References

1. G. Marchionini. Interfaces for end-user information seeking. *J. of the American Society for Information Science*, 43:156–163, 1992.
2. 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. Management*, 36(2):207–227, January 2000.
3. J. Pitkow, H. Schütze, T. Cass, R. Cooley, D. Turnbull, A. Edmonds, E. Adar, and T. Breuel. Personalized search. *Commun. ACM*, 45:50–55, September 2002.
4. T. Kramár, M. Barla, and M. Bieliková. Peweproxy: A platform for ubiquitous personalization of the "wild" web. In *UMAP '11: Adjunct Proc. of the 19th Int. Conf. on User Modeling, Adaptation, and Personalization*, pages 7–9, Girona, Spain, 2011.
5. M. Holub and M. Bieliková. An inquiry into the utilization of behavior of users in personalized web. *J. of Universal Computer Science*, 17:1830–1853, 2011.
6. Návrat P., T. Taraba, A. Bou Ezzedine, and D. Chudá. Context search enhanced by readability index. In *IFIP Series. Vol. 276: Artificial Intelligence in Theory and Practice II*, IFIP, pages 373–382. Springer-Verlag, 2008.
7. J. Teevan, S. T. Dumais, and E. Horvitz. Characterizing the value of personalizing search. In *Proc. of the 30th Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, SIGIR '07, pages 757–758, New York, NY, USA, 2007. ACM.
8. E. Agichtein and Guo Q. Towards inferring web searcher intent from behavior data. In *Proc. of the 28th Int. Conf. on Human Factors in Computing Systems*, CHI '10, Atlanta, GA, USA, 2010. ACM.

[j] ConceptNet, `http://csc.media.mit.edu/conceptnet`
[k] WordNet, `http://wordnet.princeton.edu/`

9. S. Lawrence. Context in web search. *IEEE Data Engineering Bulletin*, 23(3):25–32, 2000.
10. Taher H. Haveliwala. Topic-sensitive pagerank. In *WWW '02: Proc. of the 11th Int. Conf. on World Wide Web*, pages 517–526, New York, NY, USA, 2002. ACM.
11. F. Qiu and J. Cho. Automatic identification of user interest for personalized search. In *WWW '06: Proc. of the 15th Int. Conf. on World Wide Web*, pages 727–736, New York, NY, USA, 2006. ACM.
12. N. Kanhabua and K. Nørvåg. Quest: query expansion using synonyms over time. In *Proc. of the 2010 European conference on Machine learning and knowledge discovery in databases: Part III*, ECML PKDD'10, pages 595–598, Berlin, Heidelberg, 2010. Springer-Verlag.
13. J. Gao, X. Li, D. Micol, C. Quirk, and X. Sun. A large scale ranker-based system for search query spelling correction. In *Proc. of the 23rd Int. Conf. on Computational Linguistics*, COLING '10, pages 358–366, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.
14. A. Anagnostopoulos, L. Becchetti, C. Castillo, and A. Gionis. An optimization framework for query recommendation. In *Proc. of the third ACM Int. Conf. on Web Search and Data Mining*, WSDM '10, pages 161–170, New York, NY, USA, 2010. ACM.
15. C. Biancalana, A. Micarelli, and C. Squarcella. Nereau: a social approach to query expansion. In *Proc. of the 10th ACM workshop on Web information and data management*, WIDM '08, pages 95–102, New York, NY, USA, 2008. ACM.
16. D. Carmel, E. Farchi, Y. Petruschka, and A. Soffer. Automatic query refinement using lexical affinities with maximal information gain. In *Proc. of the 25th Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, SIGIR '02, pages 283–290, New York, NY, USA, 2002. ACM.
17. S. Liu, F. Liu, C. Yu, and W. Meng. An effective approach to document retrieval via utilizing wordnet and recognizing phrases. In *Proc. of the 27th Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, SIGIR '04, pages 266–272, New York, NY, USA, 2004. ACM.
18. P. A. Chirita, C. S. Firan, and W. Nejdl. Personalized query expansion for the web. In *Proc. of the 30th Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, SIGIR '07, pages 7–14, New York, NY, USA, 2007. ACM.
19. C. D. Manning, P. Raghavan, and H. Schtze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.
20. R. Wetzker, C. Zimmermann, C. Bauckhage, and S. Albayrak. I tag, you tag: translating tags for advanced user models. In *Proc. of the third ACM int. conf. on Web search and data mining*, WSDM '10, pages 71–80, New York, NY, USA, 2010. ACM.
21. F. Carmagnola, F. Cena, O. Cortassa, C. Gena, and I. Torre. Towards a tag-based user model: How can user model benefit from tags? In *Proc. of the 11th int. conf. on User Modeling*, UM '07, pages 445–449, Berlin, Heidelberg, 2007. Springer-Verlag.
22. M. Shepherd, C. Watters, J. Duffy, and R. Kaushik. Browsing and keyword-based profiles: A cautionary tale. In *Proc. of the 34th Annual Hawaii Int. Conf. on System Sciences*, HICSS '01, pages 4011–, Washington, DC, USA, 2001. IEEE Computer Society.
23. M. Claypool, P. Le, M. Wased, and D. Brown. Implicit interest indicators. In *Proc. of the 6th international conference on Intelligent user interfaces*, IUI '01, pages 33–40, New York, NY, USA, 2001. ACM.
24. B. R. Barricelli, M. Padula, and P. L. Scala. Personalized web browsing experience. In *Proc. of the 20th ACM conf. on Hypertext and hypermedia*, HT '09, pages 345–346, New York, NY, USA, 2009. ACM.
25. Y. Yesilada, S. Bechhofer, and B. Horan. COHSE: Dynamic linking of web resources. Technical report, Sun Microsystems, Inc., Mountain View, CA, USA, 2007.
26. J. Teevan, S. T. Dumais, and E. Horvitz. Personalizing search via automated analysis of interests and activities. In *Proc. of the 28th Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, SIGIR '05, pages 449–456, New York, NY, USA, 2005. ACM.
27. M. Barla. Towards social-based user modeling and personalization. *Information Sciences and*

Technologies Bulletin of the ACM Slovakia, 3:52–60, 2011.
28. J. Teevan, M. R. Morris, and S. Bush. Discovering and using groups to improve personalized search. In *Proc. of the Second ACM Int. Conf. on Web Search and Data Mining*, WSDM '09, pages 15–24, New York, NY, USA, 2009. ACM.
29. P. Bhattacharyya, A. Garg, and S. F. Wu. Social network model based on keyword categorization. In *Proc. of the 2009 Int. Conf. on Advances in Social Network Analysis and Mining*, ASONAM '09, pages 170–175, Washington, DC, USA, 2009. IEEE Computer Society.
30. R. Schifanella, A. Barrat, C. Cattuto, B. Markines, and F. Menczer. Folks in folksonomies: social link prediction from shared metadata. In *Proc. of the third ACM int. conf. on Web search and data mining*, WSDM '10, pages 271–280, New York, NY, USA, 2010. ACM.
31. M. R. Morris and E. Horvitz. Searchtogether: an interface for collaborative web search. In *Proc. of the 20th annual ACM symposium on User interface software and technology*, UIST '07, pages 3–12, New York, NY, USA, 2007. ACM.
32. K. McNally, M. P. O'Mahony, M. Coyle, P. Briggs, and B. Smyth. A case study of collaboration and reputation in social web search. *ACM Trans. Intell. Syst. Technol.*, 3(1):4:1–4:29, 2011.
33. T. Kramár, M. Barla, and M. Bieliková. Disambiguating search by leveraging the social network context based on the stream of user's activity. In *UMAP '10: Proc. of the 18th Int. Conf. on User Modeling, Adaptation, and Personalization*, pages 387–392, Hawaii, USA, 2010. Springer-Verlag.
34. E. Wilde and A. Roy. Web site metadata. *J. of Web Engineering*, 9(4):283–301, December 2010.
35. A. Lancichinetti, S. Fortunato, and J. Kertesz. Detecting the overlapping and hierarchical community structure of complex networks. *New J. of Physics*, March 2009.
36. D. Gayo-Avello. A survey on session detection methods in query logs and a proposal for future evaluation. *Inf. Sci.*, 179:1822–1843, May 2009.
37. R. W. White, P. N. Bennett, and S. T. Dumais. Predicting short-term interests using activity-based search context. In *Proc. of the 19th ACM Int. Conf. on Information and Knowledge Management*, CIKM '10, pages 1009–1018, New York, NY, USA, 2010. ACM.
38. J. Zhang and A. A. Ghorbani. Gumsaws: A generic user modeling server for adaptive web systems. In *Proc. of the 5th Annual Conf. on Communication Networks and Services Research*, pages 117–124, Washington, DC, USA, 2007. IEEE Computer Society.
39. M. Trella, C. Carmona, and R. Conejo. MEDEA: an Open Service-Based Learning Platform for Developing Intelligent Educational Systems for the Web. In P. Brusilovsky, R. Conejo, and E. Millan, editors, *Workshop on Adaptive Systems for Web-Based Education: Tools and Reusability*, pages 27–34, Amsterdam, The Netherlands, 2005.
40. M. Agosti and G. M. Di Nunzio. Gathering and mining information from web log files. In *Proc. of the 1st international conference on Digital libraries: research and development*, DELOS'07, pages 104–113, Berlin, Heidelberg, 2007. Springer-Verlag.
41. A. Tappenden and J. Miller. A survey of cookie technology adoption amongst nations. *J. of Web Engineering*, 8(3):211–244, September 2009.
42. M. Barla and M. Bieliková. Ordinary web pages as a source for metadata acquisition for open corpus user modeling. In *Proc. of IADIS WWW/Internet 2010.*, pages 227–233. IADIS, 2010.
43. M. T. Keane, M. O'Brien, and B. Smyth. Are people biased in their use of search engines? *Commun. ACM*, 51(2):49–52, February 2008.
44. M. Barla and M. Bieliková. On deriving tagsonomies: Keyword relations coming from crowd. In Ngoc Nguyen, Ryszard Kowalczyk, and Shyi-Ming Chen, editors, *Computational Collective Intelligence. Semantic Web, Social Networks and Multiagent Systems*, volume 5796 of *Lecture Notes in Computer Science*, pages 309–320. Springer-Verlag, 2009.