The Role of the User's Browsing and Query History for Improving MPC-generated Query Suggestions

Ioan Bădărînză, Adrian Sterca, and Florian Mircea Boian

Original scientific paper

***Abstract*—In this paper we present a way of using the very recent user's browsing history and query history, in order to improve the query suggestions mechanism used by an information retrieval system. In order to collect this kind of data, we have built a Chrome browser plugin that monitors user's web activity and stores that data so that we can create a better personal profile for each user. We then analyzed if future queries submitted by a user to the search engine can be predicted from web pages visited by that user in the past (i.e. his recent browsing history) or from queries submitted by that user in the past (i.e. his recent query history). The contribution of this paper is twofold: a) an evaluation of the relevancy of the user's recent browsing and query history to future queries submitted by this user and b) a method for personalizing the query suggestions offered by the Google search engine. More specifically, we are using the user's personal (browsing) history profile in order to reorder query suggestions offered by the Google search engine (i.e. we move query suggestions more relevant to the user's information need to the front positions in the Google provided query suggestions list).**

***Index Terms*—information retrieval, query suggestion, query auto-completion, personalized query history.**

I. Introduction

Searching for information on the web can be very difficult sometimes. There are a lot of users that do not know what terms to enter in a search input of a search system to better describe their information need. In [8], [14] we can see that most of the search queries are very short, one or two words on average and in [9], [15] we can see that these words are ambiguous. In order to help the user when performing a search, most search engines like Google, Yahoo!, Bing and others, provide query auto-completion and query suggestions. In order to better explain how search suggestions are generated, we will first describe how query auto-completion works. In almost all modern browsers, search engines and text editors we can see how, after we start typing words, it automatically tries to predict what we actually want to type. These are called 'predictive auto-completion systems' where the candidates are matched against the prefix using information retrieval techniques and also natural language processing techniques. This auto-completion is actually the highest ranked suggestion from a suggestions list. The query suggestions list is a list that contains from one to ten words (or group of words), which are usually prefixed with the subquery that the user is typing, items that are extracted from a huge log of queries submitted by all users. A very well known technique of extracting suggestions from a common query log is called *Most Popular Completion (MPC)*.

The basic principle of Most Popular Completion is users wisdom [3]. This means that, if a particular query was used by a lot of users in the past, it is more likely that, that particular query will be the first candidate as an auto-completion. We can take, as example, a very popular and well known at the moment this article was written, social media website, "facebook". If we start typing letter "f" on www.google.com, one may find the first query suggestion returned by Google to be "facebook" and that's because a lot of people are performing this particular query on the Google search engine website [1]. Simply said, MPC is actually ranking suggestions based on their popularity. Let's say that we have a search log with previous queries $QLog$, a subquery (or the prefix of the intended query) sq and a list of query-completions $QC(sq)$, where all the items are starting with the desired subquery. Using the MPC formula [22], we can calculate a rank for all items in $QC(sq)$ and order these items by their rank:

$$MPC(sq) = argmax_{q \in QC(sq)} w(q),$$

$$w(q) = \frac{freq(q)}{\sum\limits_{i \in QLog} freq(i)}$$

where $freq(q)$ is actually the number of occurrences of query q in $QLog$. In this way, the query $q \in QC(sq)$ that has the highest frequency in $QLog$ would be the first suggested by the search engine to the user.

The main focus of this paper is to use the user's personal browsing history in order to reorder the query suggestions list provided by the Google search engine so that query suggestions that are more relevant to the user's information need are moved to front positions in the final, reordered list. We do this by assigning higher ranks to query suggestions (from the Google-offered list of suggestions) that contain terms which occur frequently in recently visited web pages by the user. It is worth mentioning that at present time, Google offers query suggestions largely based on the MPC strategy, plus additional heuristics, some known, others kept secret. It also includes a minimum personalization in the offered query suggestions by

Manuscript received August 31, 2018; revised December 10, 2018. Date of publication January 28, 2019.

I. Bădărînză, A. Sterca and F. Boian are with the Department of Computer Science, Babes-Bolyai University of Cluj-Napoca; Str. M. Kogalniceanu, No. 1, Cluj-Napoca, Romania (emails: ionutb@cs.ubbcluj.ro, forest@cs.ubbcluj.ro, florin@cs.ubbcluj.ro).

Digital Object Identifier (DOI): 10.24138/jcomss.v15i1.608

[1]Google applies several heuristics on top of MPC, depending on the country the user is located in, the user is logged into his/her Google account etc., so "facebook" might not be the first query suggestion returned by Google to all possible users for a query prefix of "f" - this is shown here only as an example

considering the user's query history submitted to Google.com (either when the user is logged into Google services like Gmail or Youtube or when the browser is used anonymously). This is visible in the *History* item of the *Settings* menu that is located under the search input on the Google website. This paper is an extended version of our work [27] where we have outlined and tested the method used for personalizing query suggestion. In the current paper we review the results from [27] and also perform an analysis on the usefulness of the user's browsing history and query history for predicting future queries of the user. We will try to find to what degree the log data from the user's query history and the log data from the user's browsing history, respectively, can be used to improve the list of query suggestions offered by Google (by reordering these query suggestions so that query suggestions more relevant for the user's information need are moved to front positions). We will show that our method can further improve these MPC-based with minimum personalization query suggestions provided by the Google search engine.

The remaining of this paper is structured as follows. Section II outlines work related to ours. Following, in Section III we analyze the usefulness of the user's recent browsing history and the user's recent query history for generating relevant query suggestions. The main contribution of the paper, our method for query personalization based on the user's recent browsing history, is described in section IV, followed by the evaluations performed in Section V. Finally, the paper ends with conclusions in Section VI. Part of this work was presented in [27].

II. Related Work

A. Query Auto-Completion

Auto-completion is the way that a software system (predominantly information retrieval systems) uses, in order to predict what a user might want to type, right after the first key was pressed. In information retrieval systems, these predictions are usually based on the query logs, which are the actual queries that the users have used when they were trying to satisfy their information need [2], [7], [5], [11], [10]. The common problem that these methods have is that they are lacking a 'context', which represents the immediately preceding queries that a user submitted. A paper that addresses this problem is [3], where Bar-Youssef and Kraus have demonstrated how important these recent queries are in suggest systems. [30] further extends this work by predicting queries as search intents from short contexts consisting of previously submitted queries and click data. Authors in [31] do a similar job by considering the search context derived from both the query auto-completion log and the click log (e.g. dwell time, click number, time duration of the click session etc.). Then they use a probabilistic model in order to predict better auto-completion queries. In [28] authors build a neural-network language model for generating new query autocompletions from query prefixes that are not found among queries previously submitted by users in the past. The most important thing that differentiate our proposed method from other studies is the fact that we also consider the personal browsing history together with personal query history in trying to predict what the user might want to type next.

B. Query Suggestion

A query suggestion is an enhanced query that a search system presents to the user when he is trying to search for something. Query auto-completion and query suggestion are very tied together, because the query-autocompletion is the first suggestion from the query suggestions list. The main difference between these two is that the query auto-completion needs to be prefixed with the characters that the user has typed in the search input, whereas the query suggestion doesn't need to be prefixed with that. In [4] and [13] authors are using the click-through data as context and demonstrate that the higher a suggestion appears in a list, the more clicks it attracts. Machine learning techniques also approached the field of query suggestions in information retrieval. Jiang et al. are using the LambdaMart [12] learning algorithm in [6] in order to be able to define a set of features that would help them build suggestion lists by reformulating the original query. LambdaRank neural networks are neural networks that speed up the learning process of machine learning algorithms that use gradient descend methods. LambdaRank defines the gradient of the cost function by taking into account the rank order of the documents with respect to the query. The LambdaMart learning algorithm is just LambdaRank instantiated using gradient boosted decision trees. In [32] authors also use neural networks to offer query suggestions by either using (i.e. copying) words from the current session context in the proposed query suggestion or generating new words that should be incorporated into the suggestion. All these studies take into consideration data that is available at the search engine (i.e. in the server logs), while our paper uses data only available at the client side, namely web pages previously visited by the user and queries previously submitted by the user.

C. Personalized Search

Personalized search attracted the attention of a lot of researchers, [17], [18], [19], [20], [21]. All these studies are taking into consideration only the personal query history when offering new suggestions to users. There are lots of examples of queries [1], [23], [24] like "ajax", "jaguar" etc. which might have multiple meanings and a suggestion system might fail very easily without taking into consideration a personalized factor when computing the suggestion lists. In [22] the authors tried to extract the personalization factor by dividing the user by age, gender and region, in [16] Bennett et al. analyzed different lengths of the search sessions, and in [17] Matthijs and Radlinski started to collect browsing history and built a system through which they were re-ordering the results that Google search returned. Following the line of thinking in [3], the authors of [29] build a query suggestion system which personalizes the offered query suggestions based on the clicked documents and preceding queries in the same user session. All the above papers either consider the global or personal query history (measured at the search engine) or they use a form

of browsing history, but for re-ranking search results returned by the search engine. In contrast, we consider the personal browsing history of the user (observed at the client) in order to provide better query suggestions.

III. Analyzing the Usefulness of the User's Browsing History and Query History for Generating Query Suggestions

In this section we try to answer the question how much of the future queries submitted by a specific user to a search engine (in our case, the Google search engine) can be predicted from that user's recent *browsing history* (i.e. the web pages visited by this user in the past) or his/her *query history* (i.e. the queries submitted by this user in the past). In order to do this, we have developed a Chrome plugin that captures all web pages visited by the user and also all queries and subqueries submitted to Google together with the list of query suggestions returned by Google. The reason for collecting only Google searches is the fact that according to comScore in [25], in February 2016, out of the total explicit core searches performed on web, 64% were Google searches. The other part of 36% is divided between Bing, Yahoo, Ask Network etc. We choose to build a Chrome extension, and not an extension for other browsers, because according to latest Browser Statistics [26] from October 2017, made by www.w3schools.com, 76.1% of users are using a Chrome browser.

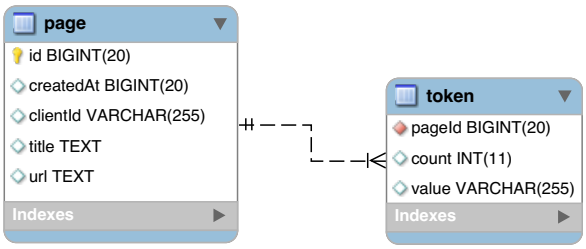
Our Chrome plugin is written in *javascript* which makes REST calls to a remote server that persists all user information in a MySQL database for later offline analysis. Whenever a new page is loaded, our plugin does the following (all webpages that are email pages, facebook pages and other pages that may contain personal information will not be analyzed):

- If the URL of the page does not start with *"www.google."*, it will interpret it as a new webpage that was viewed and will extract the actual text from the HTML document and, alongside with page URL and page title, it will split the text into terms, eliminate stop words and calculate the term frequency for each unique word. All this history data is then passed to the remote server using Ajax HTTP requests.
- If the URL of the page matches *"www.google."*, it means the user performs a Google search. In this case, for each key pressed in Google's search input, the plugin will extract the value of the search input and the list of suggestions provided by Google for the written subquery. This information is passed to the server. When the user finishes typing the desired query and submits it to Google, this final query is also submitted to the remote server.

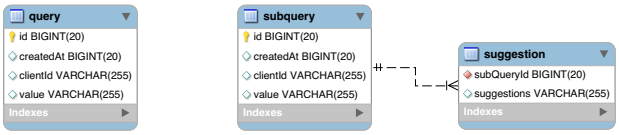
For all information that is passed by the plugin to the remote server, the plugin will associate a unique identifier for the user (which is generated once when the user installs the extension).

In Fig. 1a we can see the database diagrams for the tables where we store page history data and in Fig. 1b we can see the database diagrams for the tables where we store query, subquery and suggestions data at the remote server.

The server stores a timestamp associated to each page, representing the time when that page was visited, a timestamp



(a) Page history tables diagram



(b) Query related tables diagram

Fig. 1: Database tables diagrams

for each subquery for when it was typed in the Google search input and also a timestamp for each query that the user actually used to performed a Google search.

During a time interval of 4 months, we collected browsing history and Google query history data from 14 users who installed our Chrome plugin on their computers. In Fig. 2 we can see the amount of data that was collected by our Chrome plugin in the evaluation time frame of 4 months.

Time period	Total number of clients	Total number of visited pages	Total number of Google queries
4 Months	14	43121	4339

Fig. 2: The collected data

It is commonly accepted that search queries can be divided into two main types: navigational queries and informational queries. A navigational query is a search query entered by the user with the intent of finding a very particular webpage. For example, a user might type "facebook" into Google's search input in order to find and navigate to the Facebook website, instead of directly typing the full address in the address bar. We can say that whenever a user submits a query to Google, and the URL of the first page that he navigates to contains all the terms from the query, the query is a navigational query. An informational query is a search query that can cover a very large topic, for which, the search engine can return a very large number of relevant information (the website this information comes from is not important by itself). When a user submits such a query to Google, he is looking for some information and not a particular website. In Fig. 3, which is built from the data collected by our Chrome extension, we can see that 77% of the queries are informational queries and 23% are navigational queries. We considered a query to be a navigational query if the URL of the first page, that is visited by the user, contains all query terms; all other queries that do not follow this rule are considered as information queries.

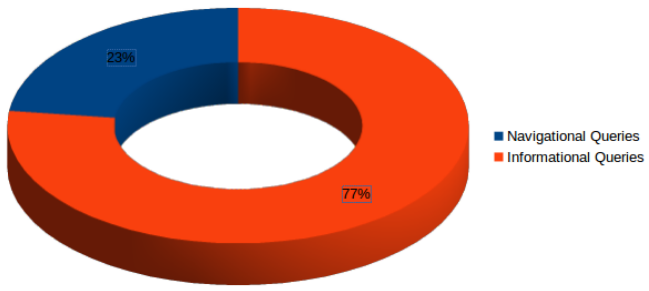


Fig. 3: Search query types

In Fig. 4 we analyzed how many queries from all the queries saved in our server's database, can be found in webpages that were previously visited by the 14 monitored users. If the query appears in the URL of the page or in the title of the page, we no longer look within the actual content of the page. We made several tests related to the length of the recent browsing history. First, we considered a recent browsing history of 30 minutes, then 60 minutes and finally 120 minutes. For each distinct query submitted by a user, we search if this query can be found in web pages visited by that specific user in the last 30, 60 and respectively 120 minutes (having as reference the time when the user submitted this query). The average percentage values across all 14 users are displayed in Fig. 4. For example, we can see that for the 30 minutes long recent browsing history, 28% of the queries submitted by a user appear in the user's recent browsing history, while 72% can not be found in this recent browsing history. We can observe how the number of queries that were found in the recent history, increases as the length of the history increases. We conclude from this figure that for the browsing history sizes considered, approximately 30% of the queries submitted by users can be predicted from the browsing history of those users.

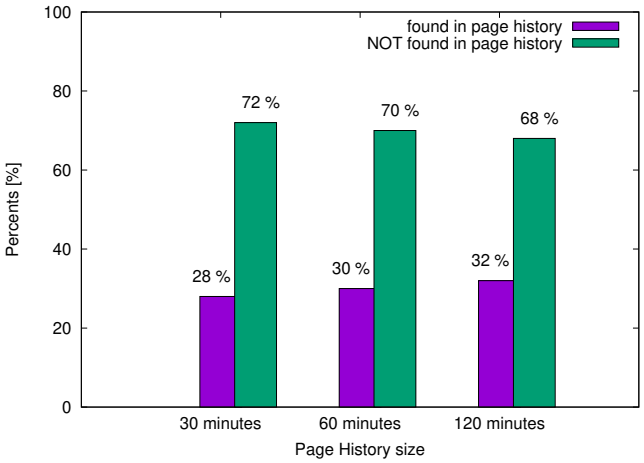


Fig. 4: The usefulness of the user's browsing history for generating query suggestions; queries that were found in pages from the browsing history

We also tried to assess if future queries can be predicted from past queries submitted by a user from the data we collected. In Fig. 5 we plotted for all queries stored in the server's databases the percentage of queries that were submitted more than once to the search engine (i.e. duplicated queries). For example, we found that out of our collected history, approximately 73% of the queries were submitted only once to the search engine (i.e. they were distinct queries), while 18.54% were submitted twice to the search engine and a small percentage of 0.03% were actually submitted 20 times to the search engine. This shows that maximum 27% of queries can be predicted from user query logs.

In the rest of the paper we only considered using the user's web browsing history for generating better query suggestions (i.e. personalizing query suggestions) and we left the user's query history influence for future work.

IV. Proposed Method for Query Personalization

Our method, which was also presented in [27], is based on the assumption that while a user is browsing web pages, at some point, he will develop an information need for which he will go to a search engine (mostly Google search) and will seek to satisfy this need. In the previous section we have shown that around 30% of the queries that a user is submitting to Google search can be predicted from a very short and recent browsing history. We assume that a query session takes place in the following way: as the user starts typing characters in the search input of the search engine, the search engine returns a list of query suggestions, $Q_s(i), i = 1..10$, ordered by their relevancy to the user's information need (relevancy is computed by the search engine using a Most Popular Completion technique, $Q_s(1)$ being the most relevant suggestion according to the search engine). The user might continue typing characters and ignore the suggestions offered or he may choose a suggestion to be the final query. This final query Q is submitted to the server. We call such a sequence of steps of the user a *search session*: starting from the first character typed by the user in the search input until he finally chooses a query suggested by the search engine. In a *search session* we can define several *search contexts*, i.e. $(SQ, Q_s(1), ..., Q_s(10))$ tuples that are made of the subquery SQ (i.e. the string typed by the user in the search input) together with 10 suggestions offered by the search engine for the subquery SQ.

In Fig. 6, which is an example extracted from the data that we've collected, we can see how the order of the suggestions change as the user types more characters in the search input. The final query that the user will choose at the end is *"pizza with pineapple"*, and we can see how this final query first appears on position 6 in the list, then, as the user typed more characters, it will move to position 3, and finally, after another couple of characters typed, it appears on position 1. The goal would be to predict and provide the final query on the highest position as soon as possible, without letting the user type many characters. Another thing that we have to mention here is that, all the subqueries and the final query, were made by a user at a specific point in time, when he had the Chorme plugin installed and that the order of these suggestions that are presented in

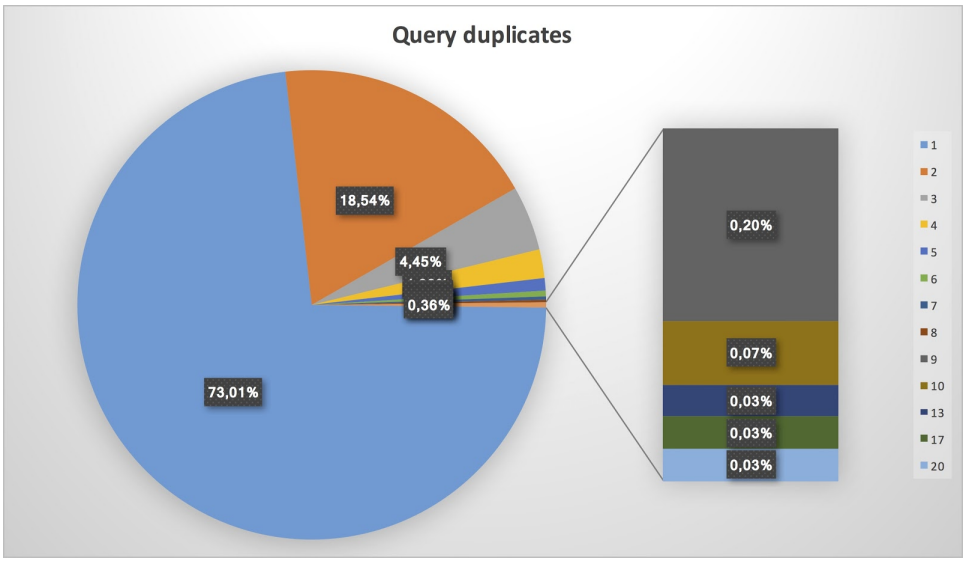


Fig. 5: The usefulness of the user's query history for generating query suggestions; queries that were submitted several times to the search engine

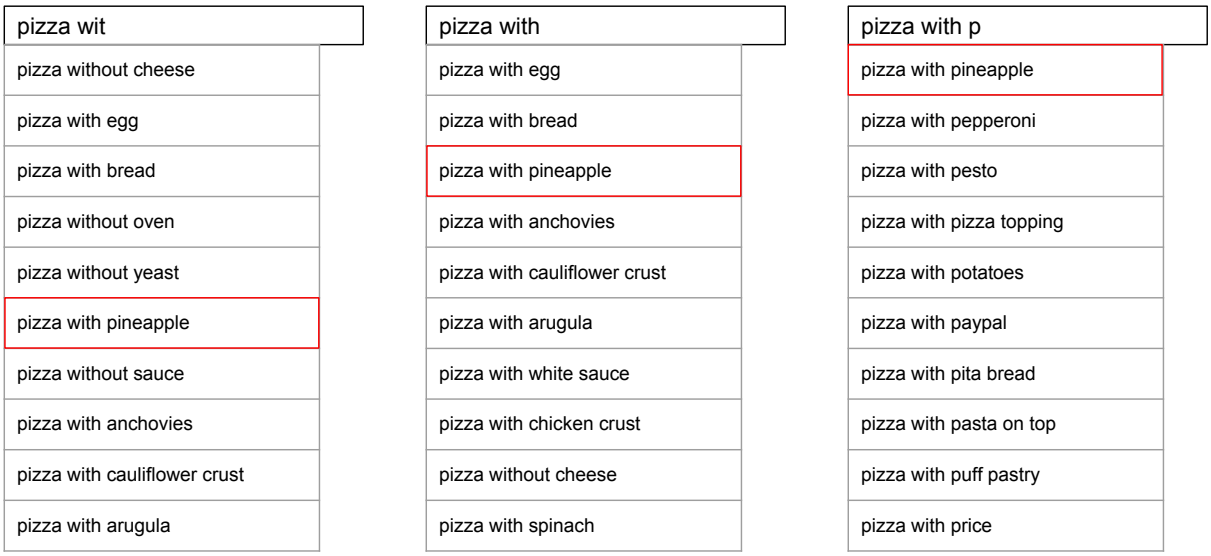


Fig. 6: Example of a (part of a) search session

this picture, is the exact same order that Goggle was offering at that time.

Our query personalization approach reorders query suggestions offered by the search engine (e.g. Google) by considering personal context metadata for each user, so that query suggestions more relevant to the user's information need are moved in front, to a higher rank, in the ordered list of query suggestions provided by the search engine. This personal metadata is extracted from the short browsing history of the user (i.e. the web pages visited by the user before the time he started typing subqueries in the Google search input).

Because our query personalization method [27] must rerank a list of 10 query suggestions provided by the Google search engine, we can not do this by relying solely on our personalization score and disregarding the Google, MPC-based, score of each query suggestion. For this reason, for reranking the query suggestions provided by Google, we use a hybrid score that takes in consideration the original ranking offered by Google and our personalized ranking score [27]:

$$HybridPageScore(Q_s) = OrigScore(Q_s) \cdot \beta + (1 - \beta) \cdot PTQS(Q_s) \quad (1)$$

where:

- $OrigScore(Q_s)$ represents a weight assigned to the suggestion Q_s based on its position in the original order of the suggestions as provided by Google. The (original)

order of suggestions is usually computed by the search engine using a *Most Popular Completion* mechanism.
- $PTQS$ is the *Personal Temporal Query Suggestion* score, our personalization score computed based on the user's recent browsing history which will be detailed below.
- $\beta \in [0,1]$ defines the relative weight of the $OrigScore$ and $PTQS$ score.

For a list of 10 query suggestions returned by Google, we have set the $OrigScore(Q_s) = 10 - GooglePositionIndex(Q_s)$ where the $GooglePositionIndex(Q_s)$ is the position index of suggestion Q_s in the list of suggestions offered by Google. The β parameter will determine how important is $OrigScore$ and $PTQS$, so for example if $\beta = 0$, then $HybridPageScore$ will be the same as $PTQS$, and if $\beta = 1$, then $HybridPageScore$ will be the same as $OrigScore$.

The *Personal Temporal Query Suggestion* score, $PTQS$, of a query suggestion Q_s is defined as [27]:

$$PTQS(Q_s) = \sum_{p \in HPage} weight(p) \cdot qt_{freq}(Q_s, p) \quad (2)$$

where:
- Q_s is the suggestion we want to compute the score for (which can contain multiple terms);
- $HPage$ represents the web pages that a user has visited (the page history of that particular user);
- $weight(p)$ is a temporal weight factor for the score of Q_s with respect to page p.

The $PTQS$ score is a personal metric (i.e. dependent on the specific user) that evaluates the dependency of the query suggestion Q_s to the recent browsing history of the user, $HPage$. In other words, it specifies numerically the correlation of Q_s to a part (or all) of the user's recent browsing history.

For each subquery SQ from a search session, we consider the (browsing) page history $HPage$ to be the web pages visited by the user in the time interval $[t_{ref}, t_{current}]$. $t_{current}$ is the end of the search session (i.e. the time when the final query, Q, of this search session is submitted) and t_{ref} is a reference time in the past, for example 30 minutes before $t_{current}$. The difference $t_{current} - t_{ref}$ describes the length of the browsing history that is considered by $PTQS$. The temporal $weight$ of a web page $p \in HPage$ used in the $PTQS(Q_s)$ score is thus:

$$weight(p) = exp\left(\frac{minutes(t(p) - t_{ref})}{minutes(t_{current} - t_{ref})}\right) \quad (3)$$

$$exp(x) = \frac{10^x - 1}{10} \quad (4)$$

where $t(p)$ is the time when page p was visited by the user ($t(p) \in [t_{ref}, t_{current}]$) and $minutes(t)$ is a function that returns the length in minutes of the time interval t. $\frac{minutes(t(p)-t_{ref})}{minutes(t_{current}-t_{ref})}$ is a linear mapping of $t(p)$ from the time interval $[t_{ref}, t_{current}]$ to the interval [0,1]. On top of this, we apply the exponential mapping $exp(.)$ which maps exponentially the values from the interval [0, 1] to the final interval [0, 0.9]. In this way, the weight difference of two pages, $weight(p_1) - weight(p_2)$, is an exponential of their timestamp difference, $t(p_1) - t(p_2)$. $weight(p)$ gives benefit to the most recent pages and lets them have a significantly higher weight than the other pages that are closer to t_{ref}, because we consider the more recent the page is, the more it might be relevant to what the user will try to type next.

The second member of (2), $qt_{freq}(Q_s, p)$, is a metric that expresses how relevant page p is for the query suggestion Q_s:

$$qt_{freq}(Q_s, p) = \frac{1}{|\{q_t | q_t \in Q_s\}|} \cdot \sum_{q_t \in Q_s} (freq(q_t, p) \cdot idf(q_t)) \quad (5)$$

where:
- $|\{q_t | q_t \in Q_s\}|$ is the number of terms from the query suggestion Q_s;
- $freq(q_t, p)$ is the frequency of query term q_t in page p;
- $idf(q_t)$ is the inverse document frequency of q_t in the entire user page history:

$$idf(q_t) = \frac{|EntirePageHistory|}{1 + |p \in EntirePageHistory;\ q_t \in p|} \quad (6)$$

where the denominator is the total number of web pages from the entire user page history, $EntirePageHistory$, that query term q_t appears in (1 is added to it in order to avoid divisions by zero).

V. Evaluations

We performed an offline evaluation of our method using the data already presented in Section III in Fig. 2. Out of all this query history data, we used in our experiments only the search contexts, $(SQ, Q_s(1), ..., Q_s(10))$, that contain the final submitted query of the search session, Q, among the list of query suggestions $Q_s(1)..Q_s(10)$.

We have performed a significant number of tests on the log data presented in Section III in order to select the best values for the length of the browsing history and β parameters from equation (1). The length of the browsing history in time units is given by $t_{current} - t_{ref}$, but since $t_{current}$ is the present time, this length is entirely determined by the t_{ref} value. We have used different values for these parameters, the length of the browsing history ranging from 0.25 hours to 24 hours and β ranging from 0 to 0.995. All these simulations are detailed in [27]. The conclusions of these simulation tests is that the best values for the length of the browsing history (i.e. $t_{current} - t_{ref}$) is 30 minutes and for β is 0.9.

In order to confirm this, we also computed the *Mean Reciprocal Rank (MRR)* [3] for the original Google ordering of query suggestions and also for the ordering given by our own score, the $HybridPageScore$.

Giving the fact that the query suggestion lists that we used for these tests, had the final query present among the suggestions, RR (reciprocal rank) will never have 0 (zero) values. In Fig. 7 we show MRR values computed for different parameters that are used in the $HybridPageScore$, and we can see that for $\beta = 0.9$ and the length of the browsing history $L = 0.5$ we obtained a better MRR than the one obtained by Google and for $\beta = 0.9$ and $L = 1$ we obtained a similar MRR value as Google. For the test case with $L = 0.5$ and

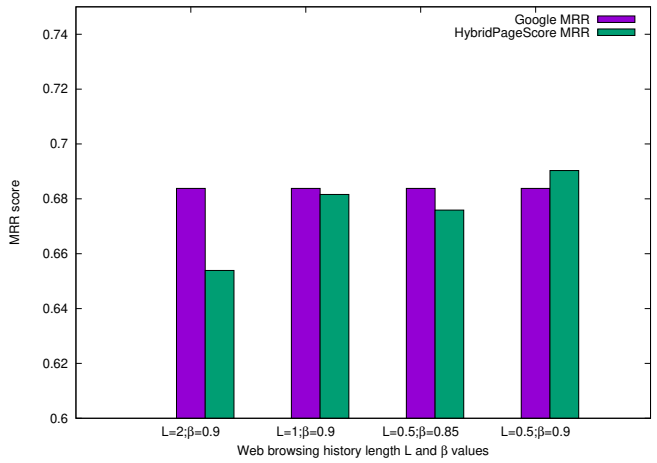


Fig. 7: MRR for different $HybridPageScore$ parameters : the length of the web browsing history, L, expressed in hours and the value of β from equation 1

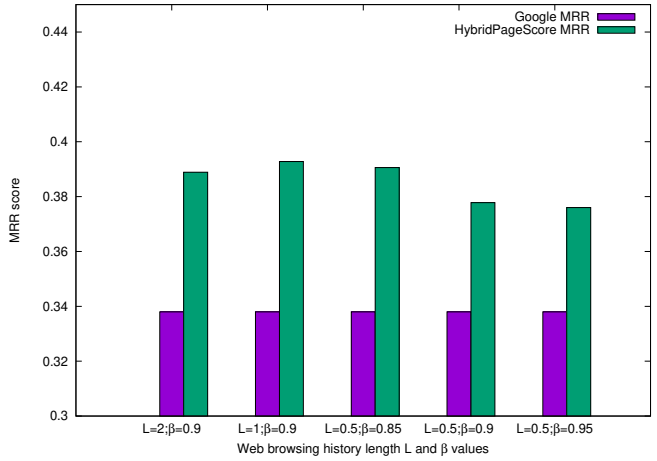


Fig. 8: MRR for Google and $HybridPageScore$ when considering only improvable Google suggestions; L is the length of the web browsing history expressed in hours and β is the parameter from equation 1

$\beta = 0.9$, we have 423 queries in total and out of these, 222 (i.e. more than half) were already placed on the 1st position/rank by the Google search engine. So these 222 lists of query suggestions can not be improved anymore. Out of the remaining 201 queries, we have 17 queries for which the original Google position is smaller than the new position computed by our Hybrid Page Score; please remember that a lower position actually means a higher rank and position 1 is actually the first suggestion in the suggestion list (having the highest rank/importance). There were 17 queries that had their rankings reduced by our $HybridPageScore$ algorithm (a total cumulative reduction of 20 ranks), but 21 queries had their rankings improved by our algorithm (a total cumulative improvement of 33 ranks). So, our algorithm improved more Google rankings than the number of rankings that were broken. And also, out of those 17 queries that had their original Google ranking reduced by our algorithm, 14 were reduced by only 1 position and 3 of them were reduced by 2 positions, so the overall reduction is rather small.

If we ignore the 222 queries that were already placed on the 1st position/rank by the Google search engine (i.e. the queries whose ranks can not be improved because they are already on the first position in the Google suggestions list) and compute the Google and $HybridPageScore$ MRR scores on the remaining 201 queries, we get the values from Fig. 8. We see here for the case with $L = 0.5$ and $\beta = 0.9$, that the $HybridPageScore$ MRR score has a 0.04 improvement over Google's MRR score (i.e. $HybridPageScore$ has an improvement of 12% over Google MRR).

VI. Conclusion

In this paper, we considered the user's recent browsing history and query history in order to provide the user better, personalized and more relevant query suggestions than the suggestions offered by a MPC-based search engine like Google. Using a Chrome browser plugin that we developed and installed on personal computers of several users, we performed an experiment that spanned over 4 months in which we collected browsing history data from all users and analyzed what percentage of the queries submitted by users to Google can be predicted from past web pages visited by this user or past queries submitted by this user to Google. In our experiments, this percentage was approximately 30%. After showing that 28% of the queries submitted by the user can be predicted from this user's recent browsing history and 27% of the queries can be predicted from his/her query history, we further developed a personalization method for query suggestions based only on the user's recent browsing history (and left the query history for future work). The query suggestions personalization method is built at the client-side, by reordering the query suggestions provided by Google, so that the ranking of the query suggestions offered by the Google search engine for a specific subquery is improved, meaning that more relevant query suggestions have higher ranks (e.g. lower positions) in the suggestion lists. We have shown using tests that span over a 4 months period that our algorithm obtains a better MRR score that improves by 12% the MRR score of Google suggestions. As future work, we plan to add this query history of the user to our algorithm and evaluate the algorithm on larger user data sets.

References

[1] Ryen W. White and Steven M. Drucker. Investigating behavioral variability in web search. In Proceedings of the 16th International Conference on World Wide Web, WWW '07, pages 21-30, New York, NY, USA, 2007. ACM. [doi: 10.1145/1242572.1242576]

[2] Holger Bast and Ingmar Weber. Type less, find more: Fast autocompletion search with a succinct index. In Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '06, pages 364-371, New York, NY, USA, 2006. ACM. [doi: 10.1145/1148170.1148234]

[3] Ziv Bar-Yossef and Naama Kraus. Context- sensitive query autocompletion. In Proceedings of the 20th International Conference on World Wide Web, WWW '11, pages 107-116, New York, NY, USA, 2011. ACM. [doi: 10.1145/1963405.1963424]

[4] Huanhuan Cao, Daxin Jiang, Jian Pei, Qi He, Zhen Liao, Enhong Chen, and Hang Li. Context-aware query suggestion by mining click-through and session data. In Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '08, pages 875-883, New York, NY, USA, 2008. ACM. [doi: 10.1145/1401890.1401995]

[5] Shengyue Ji, Cooling Li, Chen Li, and Jianhua Feng. Efficient interactive fuzzy keyword search. In Proceedings of the 18th International Conference on World Wide Web, WWW '09, pages 371-380, New York, NY, USA, 2009. ACM. [doi: 10.1145/1526709.1526760]

[6] Jyun-Yu Jiang, Yen-Yu Ke, Pao-Yu Chien, and Pu-Jen Cheng. Learning user reformulation behavior for query auto-completion. In Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '14, pages 445-454, New York, NY, USA, 2014. ACM. [doi: 10.1145/2600428.2609614]

[7] Mario Arias, Jose Manuel Cantera, Jesus Vegas, Pablo de la Fuente, Jorge Cabrero Alonso, Guido Garcia Bernardo, Cesar Llamas, and Alvaro Zubizarreta. Context-based personalization for mobile web search. In PersDB, pages 33-39, Auckland, New Zealand, 2008.

[8] Ji-Rong Wen, Jian-Yun Nie, and Hong-Jiang Zhang. Clustering user queries of a search engine. In Proceedings of the 10th International Conference on World Wide Web, WWW '01, pages 162-168, New York, NY, USA, 2001. ACM. [doi: 10.1145/371920.371974]

[9] Hang Cui, Ji-Rong Wen, Jian-Yun Nie, and Wei-Ying Ma. Probabilistic query expansion using query logs. In Proceedings of the 11th International Conference on World Wide Web, WWW '02, pages 325-332, New York, NY, USA, 2002. ACM [doi: 10.1145/511446.511489]

[10] Holger Bast, Debapriyo Majumdar, and Ingmar Weber. Efficient interactive query expansion with complete search. In Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management, CIKM '07, pages 857-860, New York, NY, USA, 2007. ACM. [doi: 10.1145/1321440.1321560]

[11] Ryen W White and Gary Marchionini. Examining the effectiveness of real-time query expansion. Information Processing and Management, 43(3):685-704, 2007. [doi: 10.1016/j.ipm.2006.06.005]

[12] Christopher J. C. Burges, Krysta M. Svore, Paul N. Bennett, Andrzej Pastusiak, and Qiang Wu. Learning to rank using an ensemble of lambda-gradient models. In Proceedings of the 2010 International Conference on Yahoo! Learning to Rank Challenge - Volume 14, YLRC'10, pages 25-35. JMLR.org, 2010.

[13] Yanen Li, Anlei Dong, Hongning Wang, Hongbo Deng, Yi Chang, and ChengXiang Zhai. A two-dimensional click model for query auto-completion. In Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '14, pages 455-464, New York, NY, USA, 2014. ACM. [doi: 10.1145/2600428.2609571]

[14] Bernard J Jansen, Amanda Spink, and Tefko Saracevic. Real life, real users, and real needs: a study and analysis of user queries on the web. Information processing and management, 36(2):207-227, 2000. [doi: 10.1016/S0306-4573(99)00056-4]

[15] Mark Sanderson. Ambiguous queries: Test collections need more sense. In Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '08, pages 499-506, New York, NY, USA, 2008. ACM. [doi: 10.1145/1390334.1390420]

[16] Paul N. Bennett, Ryen W. White, Wei Chu, Susan T. Dumais, Peter Bailey, Fedor Borisyuk, and Xiaoyuan Cui. Modeling the impact of short and long-term behavior on search personalization. In Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '12, pages 185-194, New York, NY, USA, 2012. ACM. [doi: 10.1145/2348283.2348312]

[17] Nicolaas Matthijs and Filip Radlinski. Personalizing web search using long term browsing history. In Proceedings of the Fourth ACM International Conference on Web Search and Data Mining, WSDM '11, pages 25-34, New York, NY, USA, 2011. ACM. [doi: 10.1145/1935826.1935840]

[18] Mariam Daoud, Lynda Tamine-Lechani, Mohand Boughanem, and Bilal Chebaro. A session based personalized search using an ontological user profile. In Proceedings of the 2009 ACM Symposium on Applied Computing, SAC '09, pages 1732-1736, New York, NY, USA, 2009. ACM. [doi: 10.1145/1529282.1529670]

[19] Zhicheng Dou, Ruihua Song, and Ji-Rong Wen. A large-scale evaluation and analysis of personalized search strategies. In Proceedings of the 16th International Conference on World Wide Web, WWW '07, pages 581-590, New York, NY, USA, 2007. ACM. [doi: 10.1145/1242572.1242651]

[20] Ahu Sieg, Bamshad Mobasher, and Robin Burke. Web search personalization with ontological user profiles. In Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management, CIKM '07, pages 525-534, New York, NY, USA, 2007. ACM. [doi: 10.1145/1321440.1321515]

[21] Jaime Teevan, Susan T. Dumais, and Eric Horvitz. Personalizing search via automated analysis of interests and activities. In Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '05, pages 449-456, New York, NY, USA, 2005. ACM. [doi: 10.1145/1076034.1076111]

[22] Milad Shokouhi. Learning to personalize query auto-completion. In Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '13, pages 103-112, New York, NY, USA, 2013. ACM. [doi: 10.1145/2484028.2484076]

[23] Jaime Teevan, Susan T. Dumais, and Eric Horvitz. Potential for personalization. ACM Trans. Comput.-Hum. Interact., 17(1):4:1-4:31, New York, NY, USA, 2010. [doi: 10.1145/1721831.1721835]

[24] Xuehua Shen, Bin Tan, and ChengXiang Zhai. Implicit user modeling for personalized search. In Proceedings of the 14th ACM International Conference on Information and Knowledge Management, CIKM '05, pages 824-831, New York, NY, USA, 2005. ACM. [doi: 10.1145/1099554.1099747]

[25] Comscore, https://www.comscore.com/Insights/Rankings/comScore-Releases-February-2016-US- Desktop-Search-Engine-Rankings, February 2016.

[26] W3schools, https://www.w3schools.com/browsers/, October 2017.

[27] Ioan Bădărînză, Adrian Sterca, Florian Boian, Using the user's recent browsing history for personalized query suggestions, SoftCom 2018, Split, Croatia, 2018. [doi: 10.23919/SOFTCOM.2018.8555774]

[28] Dae Hoon Park , Rikio Chiba, A Neural Language Model for Query Auto-Completion, Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, August 07-11, 2017. [doi: 10.1145/3077136.3080758]

[29] Fei Cai , Maarten de Rijke, Selectively Personalizing Query Auto-Completion, Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, July 17-21, 2016. [doi: 10.1145/2911451.2914686]

[30] Jyun-Yu Jiang , Pu-Jen Cheng, Classifying User Search Intents for Query Auto-Completion, Proceedings of the 2016 ACM International Conference on the Theory of Information Retrieval, September 12-16, 2016. [doi: 10.1145/2970398.2970400]

[31] Aston Zhang , Amit Goyal , Ricardo Baeza-Yates , Yi Chang , Jiawei Han , Carl A. Gunter , Hongbo Deng, Towards Mobile Query Auto-Completion: An Efficient Mobile Application-Aware Approach, Proceedings of the 25th International Conference on World Wide Web, April 11-15, 2016. [doi: 10.1145/2872427.2882977]

[32] Mostafa Dehghani , Sascha Rothe , Enrique Alfonseca , Pascal Fleury, Learning to Attend, Copy, and Generate for Session-Based Query Suggestion, Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, November 06-10, 2017. [doi: 10.1145/3132847.3133010]

Ioan Bădărînză is a teaching assistant at the Faculty of Mathematics and Computer Science, Babes-Bolyai University, Romania. He received the Ph.D. degree in Computer Science from Babes-Bolyai University in 2018. His current research interests are information retrieval, web services, and distributed systems. He has close collaborations with the IT industry from Cluj-Napoca.

Adrian Sterca is a lecturer at the Faculty of Mathematics and Computer Science, Babes-Bolyai University, Romania. He received the Ph.D. degree from Babes-Bolyai University in 2009. He received 2 research grants from the Romanian funding agency and was a visiting researcher at the Institute of Information Technology (ITEC), Klagenfurt University, Austria in 2003, 2004 and 2006. He has authored several research papers on networking, multimedia systems and information retrieval. He is a member of the ACM. His current research interests are networking, multimedia streaming, image processing and information retrieval.

Florian Boian is an Emeritus Professor of Computer Science at the Faculty of Mathematics and Computer Science, Babes-Bolyai University, Romania. He has authored more than 100 scientific papers and published several books on Java-based frameworks, operating systems and distributed systems. He won several research grants from the Romanian funding agency. His current research interests are web services and distributed systems.