Using the user's recent browsing history for personalized query suggestions

1st Ioan Bădărînză
Dept. of Computer Science
Babes-Bolyai University of Cluj-Napoca
Cluj-Napoca, Romania
ionutb@cs.ubbcluj.ro

2nd Adrian Sterca
Dept. of Computer Science
Babes-Bolyai University of Cluj-Napoca
Cluj-Napoca, Romania
forest@cs.ubbcluj.ro

3rd Florian Mircea Boian
Dept. of Computer Science
Babes-Bolyai University of Cluj-Napoca
Cluj-Napoca, Romania
florin@cs.ubbcluj.ro

***Abstract*—In this paper we use the user's recent web browsing history in order to provide better query suggestions in an information retrieval system. We have built a Chrome browser plugin that collects each web page visited by a user and submits it to our query suggestion server for indexing, thus building a personal history profile for each user. The contribution of this paper is a method of using this personal history profile for reordering the query suggestions offered by Google when the user searches information on Google, moving query suggestions more relevant to the user's information need to the front positions in the Google provided query suggestions list. We have collected browsing history log data for over 4 months from several users who installed our Chrome plugin on their local computers and then we performed an offline evaluation test that shows that our personalized query suggestion system improves the MRR (i.e. Mean Reciprocal Rank) score of Google query suggestions by approximately 0.04 (i.e. improves Google's MRR score by 12 percents).**

***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], [15] we can see that most of the search queries are very short, one or two words on average and in [9], [16] 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 eight 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, which is detailed in the next section of this paper.

The main focus of this paper is to use the user's personal browsing history in order to reorder the query suggestion list provided by 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. In order to determine which suggestions from the list of query suggestions returned by Google are more relevant to the user's information need, we use the user's recent web browsing history and give higher ranks to Google query suggestions which contain terms that occur frequently in web pages recently visited by that user. The remaining of this paper is structured as follows. Section II outlines work related to ours. The main contribution of the paper, our method for query personalization, is described in section III, followed by the evaluations performed in Section IV. Finally, the paper ends with conclusions in Section V.

II. Related work

Query auto-completion. Auto-completion is used almost in all information retrieval engines. We have all seen how, in the search boxes of search engines, after we start typing the first character of our query, we immediately receive a possible auto-completion which will save us keystrokes when trying to fulfill an information need. What stands at the base of all these auto-completions is mostly the query logs of those particular search engines individually. We can see this kind of research in [2], [7], [5], [11], [10]. These approaches, do return good suggestion lists, but they lack a very particular thing, which is 'context'. This context is composed by the immediately preceding queries that a user submitted. In [3], Bar-Yossef and Kraus demonstrated how recently submitted user queries, i.e. the query context, can significantly improve query auto-completion. They compare their results with the *Most Popular Completion (MPC)* algorithm which is one of the popular techniques for query suggestion. In [3], the authors mention that the basic principle of MPC is users wisdom. 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. Bar-Yossef and Kraus named their approach *NearCompletion* and they demonstrated using the MRR (i.e. Mean Reciprocal Rank)

metric, that the context of a query is very important when trying to generate suggestions. However, the aforementioned papers only consider the query history of the user and not the personal browsing history of the user which is what we analyze in this paper.

Query Suggestion. Query suggestion and query auto-completion are very similar. The main scope of both of them is to save user keystrokes when performing a search. Query suggestion is an enhanced, proposed query that the user might be looking for, whereas an auto-completion is the possible query term that the user might want to type immediately after he started typing the first letter. Usually, auto-completion happens in the same search input where the user is writing his query, whereas query suggestion, usually appears as a drop-down list from where the user can choose the desired suggestion. Basically, we can say that auto-completion is the first item from the query suggestions list. In [4], authors proposed a context aware query suggestion approach by mining click-through data and session data. First, they grouped similar queries into concepts and represented them on a bipartite graph. After this offline step, in an online step they take the user query and find the concept for it in the graph and return the queries from that concept as suggestions. Another paper where click-through data was analyzed and used for generating query suggestions is [14], where authors demonstrate that the higher a suggestion is present in a suggestions list, the more it tends to attract more clicks. In [6] Jiang et al. are reformulating the query by analyzing how users previously reformulate their queries then adding words in the query and defining a set of features which were applied using the LambdaMart [12] learning algorithm. Other researchers have tried to apply probabilistic models like Markov Processes to predict what the user's query will be immediately after he starts typing [13]. 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.

Personalized search. All the above papers do not consider the recent browsing history of the user when offering query suggestions to the user. Our focus is to analyze the usefulness of the user's recent browsing history for query suggestions which will allow us to create a personal profile for each user and use that profile when ranking query suggestions. Personalized search, in general attracted attention of a lot of researchers, [18]–[22]. Each and every study showed that the user's personal query history is very important in search systems. Let's take for example the very well known query example "ajax". This query has three meanings that we are aware of: one would be the Dutch football team "Ajax", another one would be the cleaning product "Ajax" and the last one would be "Asynchronous JavaScript and XML" used in web development. In [1], [24], [25] we can see that these kinds of queries are used by users pretty often. If we do not know anything about the user's previous searches and interests, we can not know which result represents best the user's information need. In general, the way personalized search applies in auto-completion and query suggestion is by saving each query that a user used at a particular point in time, then use all this history in ranking query suggestions. In [17], we can see how Bennett et al. demonstrated that the long term query history is very useful when the user starts his search session and the short term query history is more relevant when the search session evolves. Matthijs and Radlinski [18] used a browser plugin to collect a browsing history and used that history to re-rank search results and demonstrated that the returned results are more relevant to the user. This is similar to our paper, but we use the personal browsing history to reorder query suggestions already provided by Google, not reorder search results. Others, like Shokouhi in [23], went even further with penalization and divided users into categories based on their age, gender and region and demonstrated that all these features have an impact on the suggestion that a user is waiting for when trying to search. For example, after typing letter 'i' in a search input, the most selected suggestion by male users is 'imdb' whilst female users were choosing 'indeed.com'.

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 in order to provide better query suggestions.

III. Proposed method for query personalization

Our method 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 [26] 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.

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). 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. 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.

In our system, we start by associating a relevancy score to a query suggestion Q_s with respect to each web page visited by the user in the past. This score is made of a temporal weight of the page (i.e. if the web page is visited more recently, it will have a higher weight) and a $tf - idf$ factor measuring the relevance of the web page for the query suggestion Q_s. We then compute a cumulative history score for a query suggestion Q_s by summing all these relevancy scores of Q_s with respect to each web page visited by the user in the past. This cumulative history score is $PTQS$ (*Personal Temporal Query Suggestion*) and is defined as:

$$PTQS(Q_s) = \sum_{p \in HPage} weight(p) \cdot qt_{freq}(Q_s, p) \quad (1)$$

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:

$$weight(p) = exp\left(\frac{minutes(t(p) - t_{ref})}{minutes(t_{current} - t_{ref})}\right) \quad (2)$$

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 $exp(.)$ from equation (3), 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.

$$exp(x) = \frac{10^x - 1}{10} \quad (3)$$

The second member of (1), $qt_{freq}(Q_s, p)$, is a metric that expresses how relevant page p is for the query suggestion Q_s:

$$qt_{freq}(Q_s, p) = \frac{1}{|\{q_t | q_t \in Q_s\}|} \cdot \sum_{q_t \in Q_s} (freq(q_t, p) \cdot idf(q_t)) \quad (4)$$

where:

- $|\{q_t | q_t \in Q_s\}|$ is the number of query suggestion terms;
- $freq(q_t, p)$ 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:

$$idf(q_t) = \frac{|EntirePageHistory|}{1 + |p \in EntirePageHistory;\ q_t \in p|} \quad (5)$$

where the denominator is the total number of web pages from the entire user page history, $EntirePageHistory$, that query term q_t appears in.

Because our query personalization method must rerank a list of 10 query suggestions provided by the Google search engine, if we would reorder them solely based on our PTQS score, it is quite probable that we will decrease the rank of the final query Q in the reordered list of suggestions (this is especially true if the search engine is good and places the final query

Q on the first position/rank in the list of suggestions returned by the search engine - a situation that can not be improved by our algorithm; we can not move the final query Q from the original, 1st rank, to a better rank in the reordered list of suggestions because there is no better rank than the 1st rank). 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 ranking based on the PTQS score:

$$HybridPageScore(Q_s) = OrigScore(Q_s) \cdot \beta + (1 - \beta) \cdot PTQS(Q_s) \quad (6)$$

where:

- $OrigScore(Q_s)$ represents a weight assigned to the suggestion Q_s based on it's position in the original order of the suggestions as provided by Google.
- $\beta \in [0, 1]$ defines the relative weight of the $OrigScore$ and $PTQS$ score.

An easy example of how to set the values for the $OrigScore$ would be to take the suggestion position index in reverse order, meaning that, if the list of suggestions contains 10 suggestions, the suggestion from the first position will have $OrigScore = 10$, the suggestion from the second position will have $OrigScore = 9$ and so on. 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 $OrigScore$. We performed several tests detailed in the following section in order to find the values for β and the length of the page history $HPage$ that achieve the best results.

IV. Evaluations

We performed an offline evaluation of our method, by collecting browsing history and Google query history data from 14 users who installed our Chrome plugin on their computers. In Fig. 1 we can see the amount of data that was collected by our Chrome plugin in the evaluation time frame of 4 months. 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 $Q_s(1)..Q_s(10)$.

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

Fig. 1. Collected data

In Fig. 2 we can see the database diagrams for the tables where we store page history data and the tables where we store query, subquery and suggestions data.

The server stores a timestamp associated to each page, representing the time when that page was visited, a timestamp for each subquery for when it was typed in the Google search input and also a timestamp for each query that the user actually performed a Google search.

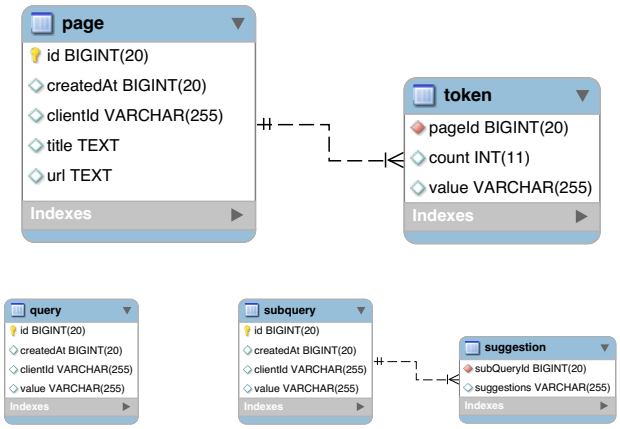

Fig. 2. Page history and Query related tables diagram

In the table from Fig. 3 we present the experiments that we have done in order to find the best values for t_{ref} and β parameters from equation (6). In the first column of the table are listed the values for t_{ref} parameter and on the second column we have the values for β parameter. We randomly chose 423 subqueries that had suggestion lists which contained the final query that the user ended up selecting. In order to easily determine whether the newly reordered suggestions list, resulted after applying the $HybridPageScore$, is better than the original Google order, we computed the difference between $originalPosition$, which is the original position of the final query, Q, in the list of suggestions provided by Google, and the $newPosition$, which is the new position of the final query, Q, after applying the $HybridPageScore$. The more positive values we have for this difference, the more improved results we get after applying the $HybridPageScore$, but at the same time we also have to be careful at the number of negative values, which means that these results were broken after applying the $HybridPageScore$ (broken results are considered those query suggestion lists for which the initial (original) position of the final query that the user selected at the end, Q, had a higher rank than the position obtained after applying the hybrid score). In order to be able to properly analyze this, in the third column of the table we listed the number of values for this difference which were greater than or equal to 0, meaning that these results were either improved or remained the same as in the Google suggestions list. In the forth column we listed the number of values greater than 0, meaning that these results were improved and in the fifth column we listed the number of values that are less than 0, which represents the number of broken results. Based on this simple analysis, we determined that the best values for $t_{ref} = 30\ minutes$ and for $\beta = 0.9$.

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

t_{ref} offset (in hours)	β value	# of suggestions where (originalPosition - newPosition) >= 0	# of suggestions where (originalPosition - newPosition) > 0	# of suggestions where (originalPosition - newPosition) < 0
4	0	197	76	226
2	0	204	72	219
1	0	233	65	190
24	0.95	327	38	96
24	0.90	291	49	132
24	0.85	278	58	145
24	0.80	271	63	152
2	0.95	390	25	33
2	0.90	350	27	73
2	0.85	338	40	85
2	0.80	332	43	91
1	0.95	407	21	16
1	0.90	377	27	46
1	0.85	361	33	62
1	0.80	352	40	71
0.5	0.95	410	18	13
0.5	0.90	406	24	17
0.5	0.85	381	28	42
0.5	0.80	368	31	55
0.25	0.995	423	1	0

Fig. 3. Tuning $HybridPageScore$ parameters

t_{ref} offset (in hours)	β value	Google *MRR*	HybridPageScore *MRR*
2	0.9	0,683835916	0,653943863
1	0.9	0,683835916	0,681600623
0.5	0.85	0,683835916	0,675903411
0.5	0.9	0,683835916	0,690289945

Fig. 4. MRR for different $HybridPageScore$ parameters

t_{ref} offset (in hours)	β value	Google MRR	HybridPageScore MRR
2	0.9	0,338041933	0,388896786
1	0.9	0,338041933	0,392788044
0.5	0.85	0,338041933	0,390630182
0.5	0.9	0,338041933	0,377775804
0.5	0.95	0,338041933	0,37603451

Fig. 5. MRR for Google and $HybridPageScore$ when considering only improvable Google suggestions

by our own score, the $HybridPageScore$. Please remember that the formula for computing MRR is: $MRR = \frac{1}{|SC|}\sum_{sc \in SC}\frac{1}{rank_{sc}}$, where SC is the set of all search contexts $sc = (SQ, Q_s(1), ..., Q_s(10))$ considered, sc is a search context and $rank_{sc}$ is the rank/position of the final submitted query of that search session, Q, in the query suggestion list of the search context $sc = (SQ, Q_s(1), ..., Q_s(10))$. Remember that $|SC| = 423$. 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. 4 we show MRR values computed for different parameters that are used in the $HybridPageScore$, and we can see that for $\beta = 0.9$ and $t_{ref} = 0.5$ we obtained a better MRR than the one obtained by Google and for $\beta = 0.9$ and $t_{ref} = 1$ we obtained a similar MRR value as Google. For the test case with $t_{ref} = 0.5$ and $\beta = 0.9$, 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. 5. We see here for the case with $t_{ref} = 0.5$ 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).

V. Conclusion

In this paper, we detailed a method for personalizing query suggestions at the client-side, using a browser plugin, 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 than Google.

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.

[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.

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

[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.

[5] Shengyue Ji, Guoliang 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.

[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.

[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

[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

[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.

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

[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] Liangda Li, Hongbo Deng, Anlei Dong, Yi Chang, Hongyuan Zha, and Ricardo Baeza-Yates. Analyzing user's sequential behavior in query auto-completion via markov processes. In Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '15, pages 123-132, New York, NY, USA, 2015. ACM.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] Ioan Bădărînză. Analyzing the usefulness of the users browser history for generating query suggestions. Studia Universitatis Babe-Bolyai Series Informatica, 62(2):5768, 2017.