User Model-based Metrics for Offline Query Suggestion Evaluation

Eugene Kharitonov[†‡], Craig Macdonald[‡], Pavel Serdyukov[†], Iadh Ounis[‡]
[†]Yandex, Moscow, Russia
[‡] School of Computing Science, University of Glasgow, UK
[†]{kharitonov, pavser}@yandex-team.ru
[‡]{craig.macdonald, iadh.ounis}@glasgow.ac.uk

ABSTRACT

Query suggestion or auto-completion mechanisms are widely used by search engines and are increasingly attracting interest from the research community. However, the lack of commonly accepted evaluation methodology and metrics means that it is not possible to compare results and approaches from the literature. Moreover, often the metrics used to evaluate query suggestions tend to be an adaptation from other domains without a proper justification. Hence, it is not necessarily clear if the improvements reported in the literature would result in an actual improvement in the users' experience. Inspired by the cascade user models and state-of-the-art evaluation metrics in the web search domain, we address the query suggestion evaluation, by first studying the users behaviour from a search engine's query log and thereby deriving a new family of user models describing the users interaction with a query suggestion mechanism. Next, assuming a query log-based evaluation approach, we propose two new metrics to evaluate query suggestions, *pSaved* and *eSaved*. Both metrics are parameterised by a user model. *pSaved* is defined as the probability of using the query suggestions while submitting a query. *eSaved* equates to the expected relative amount of effort (keypresses) a user can avoid due to the deployed query suggestion mechanism. Finally, we experiment with both metrics using four user model instantiations as well as metrics previously used in the literature on a dataset of 6.1M sessions. Our results demonstrate that *pSaved* and *eSaved* show the best alignment with the users satisfaction amongst the considered metrics.

Categories and Subject Descriptors

H.3.3 [**Information search and retrieval**]: Information search and retrieval

Keywords

query suggestions, evaluation measures, user models

SIGIR'13, July 28–August 1, 2013, Dublin, Ireland.

1. INTRODUCTION

The query suggestion mechanism within a search engine is a tool aimed to help users type less while submitting a query. In its most basic form, the list of suggested queries is formed to contain queries starting with the user's input as a prefix. Similar to the evaluation of other information retrieval systems, the evaluation of query suggestions is crucial to ensure progress in this area of research. How to perform this evaluation is the general topic addressed in this paper.

The target of evaluation in information retrieval is to assess how well a retrieval system meets the needs of its users. For most web search evaluation experiments, this assessment is performed by means of measuring document ranking effectiveness, i.e. how highly relevant documents are ranked. A variety of the web search ranking evaluation approaches exist and they can be naturally divided into *user-based evaluation* and *system evaluation* classes [20]. The system evaluation paradigm originated from the Cranfield experiments [9] and is used in evaluation initiatives such as the Text REtrieval Conference (TREC). The evaluation is performed using a test collection that consists of three components: a collection of documents, a collection of queries (called "topics" in TREC terminology) and a set of document relevance assessments, usually obtained through human judgements. As a result of the evaluation, each system is associated with a value of an evaluation metric, such as Expected Reciprocal Rank (ERR) [6] and Discounted Cumulative Gain (DCG) [14]. Once the test collection is built, it can be repeatedly used to compare different retrieval systems or even adjust the system parameters in a learning-to-rank process. The results of evaluations are reproducible and easily interpretable. However, the system-based evaluation approach has several drawbacks, as discussed by Voorhees [20]. Firstly, due to the manual judgement process, it is hard to perform a large-scale evaluation. Taking that into account, special care must be taken while selecting a set of queries to be used for evaluation, since they should be representative of the users' needs. This becomes even more complicated as these needs and users' intentions behind queries tend to change and evolve over time. Apart from that, the actual users' needs expressed by a query can be significantly different from those a judge can think of and this results in additional noise in the evaluations.

On the other hand, a user-based evaluation can be performed in either online or offline manner. For instance, AB-testing and interleaving [16] are examples of the former approach. One possible way to perform the offline user-based evaluation is to use the implicit user feedback observed in

historical query logs as a substitute for the human judges. For instance, in the work of Bennett et al. [3], the document that received the last satisfied click of the user was labelled as personally relevant to this user. Once the labels are generated, the test collection obtained has little differences from the system-based evaluation test collections. One can estimate the system effectiveness by means of an appropriate effectiveness metrics, e.g. Mean Reciprocal Rank (MRR) or Mean Average Precision (MAP) used by Collins-Thompson et al. [10].

We believe that the offline user-based evaluation is particularly promising for evaluation in the query suggestion domain and we support this with the arguments in Section 5. However, in order to follow this methodology an appropriate metric must be selected. A good effectiveness metric should be aligned with the user preferences, i.e. favour a system with higher level of user satisfaction. Indeed, a possible way to ensure alignment is to design the metric on top of the realistic model of the user behaviour. For evaluation in the web search domain, Expected Reciprocal Rank (ERR) [6] and Expected Browsing Utility (EBU) [21] are examples of user model-inspired efficiency metrics.

Unfortunately, to the best of our knowledge in the query suggestions domain neither a realistic model of the user interaction with the system nor an appropriate effectiveness metric based on this model were proposed. As we will discuss in the next section, a variety of evaluation metrics are used in the query suggestion literature. Often, these metrics are selected without justifying their alignment with the user satisfaction, hence it is not necessarily clear if the improvements reported would result in an actual improvement in the users' experience.

In this paper, we address these gaps and consider our contribution to be four-fold:

- We study the ways users interact with the query suggestion mechanism, as it is observed in the session logs, and propose a simple cascade model of the user behaviour on top of these observations;
- We introduce an algorithm to learn the parameters of the model;
- We propose a family of effectiveness metrics, called *Saved* parametrised by a user model and study several possible instantiations of these metrics;
- We perform a thorough experimental study of the considered user models and evaluation metrics, as well as the evaluation metrics used in the literature.

The remainder of this paper is organised as follows. After reviewing some related work in Section 2, we study the ways that users interact with a query suggestion mechanism and propose a simple user model in Section 3. In Section 4, we introduce an algorithm to learn the parameters of the user model from the session log. Next, we briefly discuss the considered evaluation framework in Section 5. Further, we introduce a novel family of evaluation metrics, called Saved, and discuss its connection to other metrics used in information retrieval in Section 6. Section 7 describes a methodology we use to compare the proposed user models and metrics. The dataset used in our experiments is presented in Section 8. We report and discuss the experimental results in Section 9. Finally, we close the paper with some conclusions and future work discussion in Section 10.

2. RELATED WORK

We consider our work to be mostly related to three areas of research: the user model-based evaluation metrics, the methods used to compare the effectiveness metrics, and the metrics used to evaluate query suggestions. Models of user search behaviour and their connection to the evaluation metrics gained a lot of attention in the web search domain and inspired us to follow this direction in our work (Section 2.1). Given a variety of metrics, the question arises how to compare them and this problem received a considerable attention in the research community (Section 2.2). We finish the overview of the related work with the discussion of the methods and metrics used in the evaluation of query suggestion previously used in the literature (Section 2.3).

2.1 User model-inspired IR metrics

One of the state-of-the-art web search evaluation metrics, Expected Reciprocal Rank (ERR), has a strong relation with a cascade model of the user behaviour and is defined as part of the cascade-based family of metrics [6]. The cascade model assumes that a user examines a list of ranked documents one-by-one, from top to bottom. After examining the document on the ith position, either the user is satisfied and stops the examination process or continues to the document on the position $i + 1$. The probability of satisfying a user depends on the document's relevance. A cascade-based metric is the expectation of a utility function $\phi(r)$, where r is the rank where the user finds the document she was looking for [6]. In case of ERR, the utility function $\phi(r)$ is equal to $\frac{1}{r}$.

An extension of ERR based on the cascade model with abandonment [7] was also discussed in [6]. Apart from being based on a different user model, this extension leverages a different utility function which is equal to 1 if the user find a satisfactory result, and 0 otherwise. As a result, the value of this metric is equal to the probability of the user finding a relevant result as predicted by the underlying user model.

The Expected Browsing Utility (EBU) is another search effectiveness evaluation metric proposed by Yilmaz et al. [21], which is defined as the expected document utility a user "collects" while examining a result list. As a basis, EBU uses a more sophisticated cascade user model that accounts for snippet attractiveness.

The user model that we propose in this paper can be considered as related to the cascade family and the effectiveness metrics we introduce resemble the cascade family of the web search effectiveness metrics, but applied to the query suggestion domain with different user behaviour patterns.

2.2 Comparing IR metrics

Since the aim of the evaluation is to predict whether the retrieval system meets the user needs, it is natural to require the values of evaluation metrics to be aligned with the user preferences. A considerable effort was deployed to ensure that the metrics used in the web search evaluation setting meet this criterion. However, the papers in the literature differ in the way the user preferences are obtained.

Some authors conducted online user-based studies to address this question. For instance, Radlinski and Craswell [15] studied the agreement between Cranfield-based measures such as MAP and nDCG with results of online user-based evaluation experiments. Sanderson et al. [17] compared the outcomes of a large-scale side-by-side evaluation of retrieval

systems performed by MTurkers with a preference relation over these systems imposed by Cranfield-based measures such as nDCG, MRR and Precision@10 as well as the diversity measures including α-nDCG [8], cluster recall and intent-aware precision.

The methodology considering user preference evidence in the session logs was also proposed. Chapelle et al. [6] supported ERR by a series of experiments which demonstrate that ERR shows better alignment with the user behaviour data in comparison with other widely used metrics. These experiments fall into two different categories. Firstly, the authors demonstrate that across different queries and possible rankings, ERR is better correlated with user click metrics such as search success rate. Secondly, in a simulated experiment it was shown that the difference in ERR of two ranking functions is better correlated with the difference in actual user preferences in comparison with other metrics. A similar approach to compare various metrics used to evaluate diversified result set was leveraged in Chappelle et al.'s work [5].

The evaluation methodology we use in this paper is influenced by the methods of Chapelle et al. [6], in that we compare the considered metrics in terms of their correlation with the user preferences observed in historical query logs.

2.3 Query suggestion evaluation

Shokouhi et al. [18] evaluated the quality of suggestions for a given prefix by the mean reciprocal rank of the most popular results (MRR), and the Spearman correlation between the predicted and ground-truth ranks of the selected queries. These metrics were averaged over a set of test prefixes. Considering a query as relevant if it is top-ranked according to the ground-truth query frequencies, Strizhevskaya et al. [19] reported P@3, AP@3 and nDCG@3 averaged over the observed prefixes.

Bar-Yossef and Kraus [2] used a session log-based approach for evaluation, which aimed to emulate user experience. From a session in the log, they extracted the user's context and the submitted query. After that, the suggestions are filtered to have the first character of the query as a prefix and ranked according to the user's context. The quality of the ranking is assessed as a reciprocal rank of the user's query in this ranked list of suggestions, weighted by the number of completions available for the prefix. They reported the weighted mean reciprocal rank (wMRR) averaged over the query-context pairs.

Duan and Hsu [12] used the minimal number of key presses the user has to make in order to issue the target search query (Minimal Keystrokes, MKS) to evaluate query corrections. This metric evaluates the effectiveness of the suggestion mechanism with respect to the user who always selects the optimal way of submitting a query.

As we can see from the related work, the query suggestion effectiveness metrics used in the literature are not specifically designed to reflect user satisfaction nor has their suitability been empirically shown. Following research in the web search evaluation, we address this gap by firstly modelling the user behaviour in Section 3 and devising an effectiveness metric upon it later in Section 6.

3. USER MODEL

The interface usually used to present query suggestions leads to the following process of users' interaction with the query suggestion mechanism. Let us suppose that a user is going to submit a query q to a search engine. After typing each character, the list of suggestions is changed to match the updated input and the user has a chance to examine the list before typing the next character. The examination is performed in steps and at the jth step the user either examines the query suggested on the jth position or skips it. If the query q is suggested by the system and examined by the user, she selects it from the list and that ends the interaction with the query suggestion mechanism. In the worst case, the user types the entire query q manually.

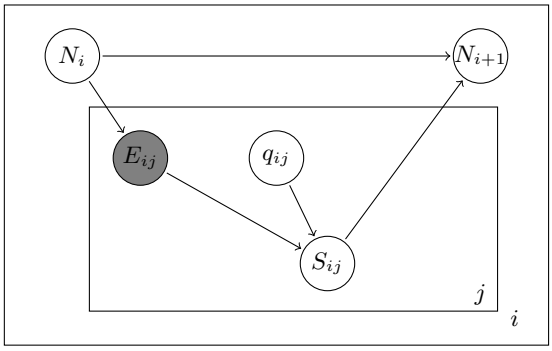

Figure 1: Graphical model of the user behaviour. Grey circles correspond to latent variables.

In order to build a user model upon this interaction schema, we assume that the user's behaviour satisfies the Markovian assumption, i.e. that given the user's current state, all of the future user's actions are independent from her earlier behaviour. The Markovian assumption is often used in the web search behaviour models, e.g. the cascade model [11] describes the user's click behaviour as a Markovian process.

The underlying graphical model is depicted in Figure 3. Denoting the prefix of the query q of length i as $q[1..i]$ and the query suggested on position j after submitting $q[1..i]$ as q_{ij}, we introduce the following random variables used in the graphical model:

- N_i: was the ith character of the query q submitted,
- E_{ij}: is the query suggested on position j for the prefix $q[1..i]$ (q_{ij}) examined,
- S_{ij}: is the user satisfied with q_{ij} after submitting $q[1..i]$,
- T_{ij}: is a variable denoting the user's state on the jth step after submitting $q[1..i]$

The model can be described using the following system of equations in terms of the random variables introduced above:

$$N_1 = 1 \tag{1a}$$
$$N_i = 0 \Rightarrow N_k = 0, k = (i+1)...|q| \tag{1b}$$
$$N_i = 0 \Rightarrow \forall j \; E_{ij} = 0 \tag{1c}$$
$$P(E_{ij}|T_{ij}) = f(T_{ij}) \tag{1d}$$

$$E_{ij} = 0 \Rightarrow S_{ij} = 0 \quad (1e)$$

$$S_{ij} = 1 \Leftrightarrow E_{ij} = 1,\ q_{ij} = q \quad (1f)$$

$$\exists j : S_{ij} = 1 \Rightarrow N_{i+1} = 0 \quad (1g)$$

$$\forall j\ S_{ij} = 0, i < |q| \Rightarrow N_{i+1} = 1 \quad (1h)$$

$$|q| = i,\ \forall j\ S_{ij} = 0 \Rightarrow N_{i+1} = 0 \quad (1i)$$

Indeed, the above equations describe the following constraints on the model: The first character is always submitted (1a); Characters are submitted sequentially (1b); Only the suggestions for the submitted prefixes can be examined (1c); The probability of examining the query suggested on the jth position is a function of the user's state (1d); A non-examined query suggestion cannot satisfy the user (1e); Examination of the query q is necessary and sufficient for the user to be satisfied, i.e. after examining the query q the user is always satisfied (1f)[1]; A satisfied user stops interaction with the query suggestion mechanism (1g); An unsatisfied user types the query until its end (1h) & (1i).

In the model described above, we do not specify the exact form of dependence of examination probabilities from the user's state and denoted it as a function $f(T_{ij})$. Varying the form of this dependence we can obtain different user models. For instance, the following functions can be considered:

1. The user always examines all the suggested queries: $f_1(T_{ij}) = 1$;
2. The examination probability depends only on position j and decays under reciprocal $f_{rr}(T_{ij}) = 1/(j+1)$ or logarithmic $f_{log}(T_{ij}) = 1/\log_2(j+2)$ laws[2];
3. The examination probability depends only on the position: $f_l^i(T_{ij}) = A_j$;
4. The examination probability depends not only on the position, but also on the prefix length: $f_l^d(T_{ij}) = B_{ij}$;

The functions $f_l^i(T_{ij})$ and $f_l^d(T_{ij})$ depend on parameters, A_j and B_{ij}. Instead of using heuristic functions such as $f_{rr}(T_{ij})$ or $f_{log}(T_{ij})$, these parameters can be learned to obtain a model that better reflects the user behaviour. In the following, we discuss an algorithm to adjust these parameters to the user behaviour observed in a session log.

The proposed model of the user behaviour is related to the cascade model [11] of the user's search click behaviour. Indeed, the user continues to type his query if and only if she is not satisfied with the current suggested queries. On the other hand, the process of examination of the list of suggestions resembles one considered in the position-based user models [11].

[1]Queries that have been seen by the user, but not selected from the suggestions list are also considered as non-examined. This do not influence generality of our proposed model.

[2]The logarithmic and reciprocal rank decays are used in DCG [14] and MRR web search metrics, respectively. We shift both functions so they start on the second rank position for two reasons: 1) the resulting probabilities are closer to those observed in our query log; 2) we avoid singularities in the user model evaluation in Section 7.1.

Input: Q: a set of sessions with the query suggestion mechanism used
Output: Examination probabilities A_j, B_{ij}
Initialise $\forall i, j\ A_j^c = 0,\ A_j^s = 0,\ B_{ij}^c = 0, B_{ij}^s = 0$
foreach *session* $\in Q$ **do**
  foreach $i \in 1..|q(session)|, j$ **do**
    if $q_{ij}(session) = q(session)$ *and* q_{ij} *not clicked* **then**
      $A_j^s \leftarrow A_j^s + 1$
      $B_{ij}^s \leftarrow B_{ij}^s + 1$
    end
    if $q_{ij}(session) = q(session)$ *and* q_{ij} *clicked* **then**
      $A_j^c \leftarrow A_j^c + 1$
      $B_{ij}^c \leftarrow B_{ij}^c + 1$
    end
  end
end
foreach j **do**
  $A_j \leftarrow \frac{A_j^c}{A_j^c + A_j^s}$
end
foreach $i,\ j$ **do**
  $B_{ij} \leftarrow \frac{B_{ij}^c}{B_{ij}^c + B_{ij}^s}$
end

Algorithm 1: Learning the prefix length-independent A_j and the prefix length-dependent B_{ij} probabilities of examination of a query suggestion presented on position j for a prefix of length i.

4. LEARNING THE MODEL PARAMETERS

Let us consider a session with a query q submitted by selecting it from the list of queries suggested to a prefix $q[1..l]$ on position k. Before the interaction stopped, the following random events took place. The user skipped the query q each time it was suggested for a prefix shorter than l. The probability of this is equal to the following expression:

$$P_{skip} = \prod_{i=1}^{l} \left[1 - \sum_j I(q = q_{ij}) P(E_{ij})\right]$$

where $I(q = q_{ij})$ equals 1 if the query q was suggested on position j for the prefix $q[1..i]$, and 0 otherwise. Further, the user examined kth suggested query for prefix $q[1..l]$. Thus, the likelihood of the entire session is as follows:

$$L(s) = \prod_{i=1}^{l} \left[1 - \sum_j I(q = q_{ij}) P(E_{ij})\right] P(E_{lk})$$

By $l(s)$ we denote the length of the prefix typed in the session s. Substituting $P(E_{ij})$ with $f(T_{ij})$ the log-likelihood of a set of sessions Q can be represented in the following form:

$$\hat{L} = \sum_{s \in Q} \sum_{i=1}^{l(s)} \sum_j \log \left[(1 - I(q_{ij} = q) f(T_{ij}))^{1 - S_{ij}} \cdot f(T_{ij})^{S_{ij}}\right] \quad (2)$$

The log-likelihood expressed in Equation (2) can be maximised with respect to the function f to find maximum likelihood estimates (MLE) of its parameters. Assuming that $P(E_{ij})$ are binomial random variables determined by the prefix length-independent f_l^i or the prefix length-dependent

f_l^d functions, discussed in the previous section, we can find MLE estimates of their parameters, A_j and B_{ij}. These estimates can be found by means of Algorithm 1, which resembles the learning process for obtaining the parameters of the cascade model [11]. Assuming that the user's examination process is the same for sessions with the query suggestion mechanism used and for sessions where it is not used, we restrict Algorithm 1 to process only sessions with the suggestion mechanism used in order to avoid noise caused by copy-pasted queries.

The functions f_l^i and f_l^d with parameters A_j and B_{ij}, respectively optimised on a training part of a dataset described in Section 8 are reported in Table 1. We additionally report values of f_{rr} and f_{log} for comparison.

Based on an analysis of Table 1, the following conclusions can be made. Firstly, the probability of examination E_{ij} indeed shows considerable dependence on the prefix length i: for shorter prefixes, the users tend to examine the suggested queries more carefully. For instance, the probability of examination of the first position changes from $f_l^d(T_{1,1}) = 0.55$ for prefixes of length 1 to $f_l^d(T_{6,1}) = 0.33$ in case of prefixes of length 6. Another observation is that the probabilities of examination for a fixed position become almost stationary for prefixes longer than four characters, e.g. $f_l^d(T_{4,1}) = f_l^d(T_{5,1}) = f_l^d(T_{6,1})$.

Even if one considers the approximation of the probabilities of examination to be independent from the prefix length, the resulting probabilities estimated from the query logs (the function f_l^i) are different from the one imposed by the position discount functions often considered in other domains of information retrieval: the first two positions have considerably lower chances to be examined ($f_l^i(T_{1,1}) = 0.36$) than it is predicted by f_{log} ($f_{log}(T_{1,1}) = 0.63$) or f_{rr} ($f_{rr}(T_{1,1}) = 0.5$) functions. Also, it is noticeable that the reciprocal rank function decays with the suggestion rank faster than the examination probabilities learned from the user behaviour in the session log.

We have introduced the user model, its possible instantiations and the algorithm to learn their parameters from the session log. Before defining the proposed user model-based effectiveness metrics in Section 6, we firstly describe the evaluation framework we are working within and which we define in Section 5.

5. OFFLINE EVALUATION OF QUERY SUGGESTIONS

Before defining new effectiveness metrics for the query suggestion evaluation we need to discuss our evaluation framework used to evaluate a query suggestion mechanism. We use the same query log-based approach as used in [2] which falls into the user-based offline category of the evaluation approaches discussed in Section 1. In this section, we discuss the evaluation scenario imposed by the framework, its restrictions and how it compares to the experimental methodologies used in the query suggestions literature.

In the case of query suggestions domain, the considered approach implies the following evaluation algorithm. Given a query suggestion mechanism with a ranking function r and a log of user sessions Q, the evaluation is performed in three steps. At the first step, the process of submitting a query $q \in Q$ is simulated as if a user typed it. For each prefix $q[1..i]$, all of the possible candidate suggestions are ranked according to the ranking function r, i.e. the simulated user is presented with a ranked list of suggestions $r(q[1..i])$. Considering q as the only query relevant to a user, this simulated output is used to estimate the effectiveness metric. Finally, the metric is averaged over all simulated sessions.

In order to perform such an evaluation, a query log is needed. However, we consider this requirement as not restrictive in the query suggestions effectiveness assessment, since query suggestion mechanisms are often built upon the query log mining [1].

We believe that this approach is sufficiently general to cover several interesting evaluation scenarios, e.g. the minimal dataset required to evaluate the query suggestion mechanism includes only queries (and possibly their frequencies). In the more advanced setting, the same methodology is suitable to evaluate personalisation algorithms by associating each session with the context (e.g. [2, 18, 19]) or the user's long-term profile. It is also noticeable that this evaluation scenario generalises other approaches to evaluate query suggestions discussed in the literature. Strizhevskaya et al. [19], as well as Shokouhi and Radinksy [18] both sampled the test sets of prefixes and evaluated the considered systems by assessing how good the most popular prefix completion was ranked having this prefix as the user input. It is possible to consider their methodology as a special case of the evaluation approach used in this paper. Indeed, considering the set of popular queries for all the test prefixes, one can generate simulated sessions with users submitting these prefixes and measuring the system effectiveness afterwards.

Guided by this evaluation scenario, in the next section we introduce novel evaluation metrics for query suggestions.

6. PROPOSED METRICS

As we mentioned in Section 1, the family of cascade-based metrics (e.g. ERR [6]) is defined as the expectation of a utility function at a position the user finds the result she is looking for. In order to generalise this family to the query suggestion evaluation, let us recall the notation previously used in Section 3. We denote the query to be submitted as q and its length as $|q|$. A prefix of q of length i is referred to as $q[1..i]$. E_{ij} is a binary variable equal to 1 if a query suggestion for the prefix $q[1..i]$ ranked on the jth position is examined by the user, S_{ij} is a binary variable representing if the user was satisfied with the jth suggestion shown for the prefix $q[1..i]$. q_{ij} denotes a suggested query ranked on position j after submitting $q[1..i]$. T_{ij} is a variable denoting the user's state, and $U(T_{ij})$ is a utility function.

Using this notation, we can adapt the notion of the cascade-based metric $V(q)$ to the query suggestion evaluation:

$$V(q) = \sum_{i=1}^{|q|} \sum_{j} U(T_{ij}) P(S_{ij} = 1) \tag{3}$$

where $\sum_j P(S_{ij} = 1)$ equals to the probability to stop after submitting $q[1..i]$. The question arises how to choose the utility function. In the simplest case, one can define the utility function to be a binary indicator of success: $I(S_{ij})$ equals 1 if the user used the query suggestions, and 0 otherwise. A similar utility function was used by Chapelle et al. [6] to build the modification of ERR based on the cascade model with abandonment. Given such an utility function, the metric equates to the probability of the user using the query suggestion mechanism. We refer to this metric as *pSaved*,

Table 1: Probability of examination. $f_{rr}(j)$, $f_{log}(j)$ and $f_l^i(j)$ correspond to the prefix length-independent logarithmic, reciprocal and learned examination probabilities, respectively. $f_l^d(i,j)$ denotes the prefix length-dependent probabilities of length i, as learned from the query log.

Query rank, j	1	2	3	4	5	6	7	8	9	10
$f_{rr}(j)$	0.50	0.33	0.25	0.20	0.17	0.14	0.13	0.11	0.10	0.09
$f_{log}(j)$	0.63	0.50	0.43	0.39	0.36	0.33	0.32	0.30	0.29	0.28
$f_l^i(j)$	0.36	0.24	0.20	0.19	0.17	0.16	0.16	0.16	0.16	0.15
$f_l^d(1,j)$	0.55	0.38	0.26	0.29	0.24	0.19	0.20	0.19	0.18	0.17
$f_l^d(2,j)$	0.56	0.34	0.31	0.26	0.22	0.20	0.18	0.18	0.17	0.14
$f_l^d(3,j)$	0.29	0.23	0.21	0.18	0.17	0.16	0.16	0.15	0.15	0.14
$f_l^d(4,j)$	0.33	0.27	0.23	0.21	0.19	0.18	0.18	0.18	0.18	0.16
$f_l^d(5,j)$	0.33	0.27	0.23	0.21	0.19	0.18	0.18	0.18	0.18	0.16
$f_l^d(6,j)$	0.33	0.27	0.23	0.21	0.19	0.18	0.18	0.18	0.18	0.16

and it is formally defined as follows:

$$pSaved(q) = \sum_{i=1}^{|q|} \sum_{j} I(S_{ij}) P(S_{ij} = 1) = \sum_{i=1}^{|q|} \sum_{j} P(S_{ij} = 1) \tag{4}$$

A more complicated utility function might decrease if it takes the user more effort[3] to find the satisfactory result and thus it can be considered as a formalisation of the amount of the effort the user can avoid due to the retrieval system under consideration. In the case of a query suggestion mechanism, the user's effort can naturally be represented as the number of characters (keypresses) the user has to type to submit her query to the system. Supported by this intuition, we propose the metric *eSaved*, which is defined as the expected ratio of characters a user can skip inputting until her query is submitted. The query can be submitted either by selecting it from the suggested list of queries or by fully entering the query. Formally, *eSaved* can be calculated using the following expression:

$$eSaved(q) = \sum_{i=1}^{|q|} \left(1 - \frac{i}{|q|}\right) \sum_{j} P(S_{ij} = 1) \tag{5}$$

where $\left(1 - \frac{i}{|q|}\right)$ is the utility function.

Both proposed metrics, *pSaved* and *eSaved* are parameterised by the user model, which defines the probability of the user satisfaction $P(S_{ij} = 1)$. In the user model proposed in Section 3, this probability is defined by Equations (1d), (1e) and (1f). The user is satisfied with a suggested query q_{ij}, only if it is the target query ($q_{ij} = q$) and if it is also examined ($E_{ij} = 1$):

$$S_{ij} = 1 \Leftrightarrow E_{ij} = 1, q_{ij} = q$$

In order to get an additional insight into the difference between the proposed metrics, we re-group Equation (5) in the following form:

$$eSaved(q) = \sum_{i=1}^{|q|} \sum_{j} P(S_{ij} = 1) - \sum_{i=1}^{|q|} \frac{i}{|q|} \sum_{j} P(S_{ij} = 1) \tag{6}$$

Comparing (4) and (6) we notice that *eSaved* equals to *pSaved* minus the expected part of the query the user needs to type to submit query q. As a result, *eSaved* additionally stratifies queries with equal chances to satisfy the user, according to the relative length of the query the user need to type.

This ability of *eSaved* to leverage this additional "dimension" to assess the query suggestion ranking functions can be particularly useful for longer queries where its utility function has a wide spectrum of values. We believe that improvements in the query suggestions for longer queries have a high influence on the user experience. Indeed, when submitting a long query, the suggestion mechanism can save greater effort for the user than in the case of a short query.

In this section, we proposed two novel metrics to evaluate the effectiveness of the query suggestions. However, it is unclear how one can compare effectiveness metrics and in the next section we discuss this issue.

7. EVALUATION METHODOLOGY

Our empirical study has the following goals. The first goal is to investigate the user model instantiations introduced in Section 3 compare to each other in terms of fitness to the observed user behaviour in the data. Each of these user models induces a corresponding metric from the *Saved* family and the question arises how well these metrics are aligned with the user behaviour data. It is also important to compare the proposed metrics with the ones previously used in the literature.

In order to answer these questions we perform a series of experiments with the methodology described in this section.

7.1 User model evaluation

The effectiveness of the user models is often studied by means of measuring the log-likelihood on the test data, e.g. [22]. The log-likelihood is defined as the average log probability of the events observed in the test dataset according to the probabilistic model under consideration.

Let s be a session from a dataset of sessions Q, q_s denotes the query submitted by the user in the session s and C_s^i is a binary indicator representing if the user's interaction with the query suggestions ended (i.e. no additional characters were typed) after submitting the prefix $q_s[1..i]$. By definition, $C_s^{|q_s|} = 1$ if the user typed the entire query. Again, $l(s)$ denotes the number of characters submitted in session s.

In the case of the query suggestions, the log-likelihood of the user model measures how well this model predicts

[3]The utility function used in ERR degrades as the user examines more retrieved results.

a prefix which the user typed before submitting the query. More formally, the log-likelihood of the model on the session with the submitted query q is defined in the following way:

$$L(q) = \sum_{i=1}^{l(s)} \left[P(C_s^i) \log_2 P(C_s^i) + (1 - C_s^i) \log_2 (1 - P(C_s^i)) \right]$$

The overall log-likelihood is calculated as the average of the session likelihoods:

$$LL(Q) = \frac{1}{|Q|} \sum_{q \in Q} L(q)$$

Another measure widely used to evaluate the performance of click models is the average *perplexity* for top ten positions [13]. However, since the queries differ in their length and the number of suggestions available, the perplexity becomes less intuitive in the considered case.

7.2 Metrics evaluation

In order to compare the considered metrics we adapt the evaluation methodology from Chapelle et al. [6] which is also used in Chapelle et al. [5]. This methodology is aimed to show how good the tested metrics are aligned with the user satisfaction indicators. Chapelle et al. considered different click measures as indicators of the user interest, such as the search abandonment rate, the position of the first click and others. We believe that the query suggestion mechanism can be considered as useful and successful in a particular session if the user used it to submit her query. Thus we use the query suggestion mechanism's usage frequency (how often the query suggestion mechanism is used by users to submit their queries) as a ground-truth indicator of the user satisfaction. In the following, we refer to this value as *success rate* (SS). In general, other indicators of user satisfaction can be considered, e.g. mean ratio of characters a user can skip inputting until her query is submitted. The indicator selected might influence the evaluation result and should be chosen in agreement with the query suggestion mechanism's performance indicators.

Due to various personalisation algorithms, geographical contextualisation features, drifts in query popularity and changes in suggestion mechanism, different users may be presented different suggested queries while submitting the same query. We denote by configuration c a unique sequence of queries suggested for query q.

The considered metric evaluation method [5] is performed in two steps. Firstly, given a dataset of user sessions one can estimate the values of the considered metrics for each configuration observed by a user. On the other hand, for each configuration shown to the users the average value of the success rate can be calculated. In the next step, the correlation between these two values across configurations is calculated.

As discussed by Chapelle et al. [6], this approach has one possible drawback. Indeed, the correlation measures the alignment of the metrics with the user satisfaction across configurations and queries, i.e. it also measures how useful the metric is to compare the effectiveness of different queries. However, in a real-life scenario, the primary goal of a metric is to compare different ranking functions of query suggestions given a fixed set of queries. Therefore, another metric's feature is essential: if, given a *fixed* set of queries one ranking algorithm outperforms its counterpart in terms of the considered metric, does this necessarily imply that the first algorithm has higher user satisfaction once deployed? There is a possibility that a particular metric can exhibit poor performance when comparing effectiveness across queries but exhibit a good alignment with the user satisfaction from the ranking function comparison perspective.

Input: Q: a dataset of user sessions
Output: Correlation between query suggestion success rate SS and a effectiveness metric M

```
foreach i ∈ 1..1000 do
    foreach query q ∈ Q do
        C_q ← set of configurations of q
        if |C_q| ≥ 2 then
            c_1, c_2 ← two random configurations from C_q
            Assign c_1 to a simulated system r_1, c_2 to a
            simulated system r_2
        end
    end
    Compute average value of M for r_1 and r_2
    Compute average value of SS for r_1 and r_2
end
Return correlation between differences in M and SS for
all simulated pairs of r_1 and r_2
```

Algorithm 2: Algorithm used to evaluate if the difference in effectiveness metric values correlated with the differences in user satisfaction proposed in [6].

In order to study the metric quality from the latter point of view, we perform an experiment proposed by Chapelle et al. [6], which simulates a comparison of the ranking functions scenario. The description can be found in Algorithm 2. Informally, the idea behind the algorithm is as follows. For each query with at least two configurations in the session log, configurations c_1 and c_2 are randomly sampled. These configurations and the corresponding sessions are associated with two simulated ranking functions, r_1 and r_2, as if all sessions with c_1 and c_2 shown to the users were served by ranking algorithms r_1 and r_2, respectively. After that, the average values of the considered metric M and the user satisfaction indicator SS are calculated for both systems. Then, the differences of the user satisfaction indicator $SS(r_1) - SS(r_2)$ and the effectiveness metric values r_1 and r_2, $M(r_1) - M(r_2)$ are found. Finally, after repeating this simulation, the correlation between the metric and the satisfaction indicator differences is returned.

8. DATASET

Before discussing the experimental results in the next section, we shortly describe the dataset used in this paper. The dataset was randomly sampled from the query log of a commercial search engine. The search sessions were performed by users from a European country with two non-English languages[4] widely spoken during two consecutive workdays in January 2013. In order to reduce noise in our evaluation, we applied the following filtering procedure to the dataset. Firstly, we do not consider sessions with misspelled queries, leaving the adaptation of the proposed models to misspelled

[4]We believe that results we report in this paper generalise across different languages. However, we leave the experimental verification of this assumption as future work.

queries as a direction of future work. We also removed sessions where users selected a query from the suggested list and edited it afterwards since it is unclear if the query suggestion mechanism was useful in these sessions. All queries were normalised to the lower case, since the character capitalisation is typically ignored by query suggestion mechanisms. Finally, only query sessions with the query suggestions shown were sampled.

As a result, the dataset contains 6.1M query sessions, 3.8M unique configurations and 3.3M unique queries. The mean query length is 26.4, the median is 24.0 characters. The mean number of whitespace delimited terms is 3.3, the median is 4.0. The length of the query suggestion lists was restricted by the deployed query suggestions mechanism to be no longer than 10.

The sessions from the first day were used to estimate the user model parameters discussed in Section 3, while the evaluation of the models and the effectiveness metrics comparison were performed on the subset of sessions performed on the second day.

9. RESULTS AND DISCUSSION

We split our evaluation experiments into two parts. Firstly, we discuss the experimental evaluation of the user models (Section 9.1). After that, we report the results of the effectiveness metrics evaluation (Section 9.2).

9.1 User model evaluation

The results of the user model evaluation on the test dataset are reported in Table 2. We report the log-likelihood of the models with the following functions determining the probability of examination of the query suggestions discussed in Section 3: f_{rr}, f_{log}, f_l^i and f_l^d. The first three functions correspond to the probability examination functions which are independent from the prefix length, while the last one is the prefix length-dependent. The parameters of f_l^i and f_d^l are learned from the train dataset by means of Algorithm 1. We repeat the experiment several times, splitting queries in groups according to their absolute frequency and length. All reported differences are statistically significant according to the paired t-test, $p \leq 10^{-3}$ except for pairs labelled by $\diamond$, which do not statistically significant differ.

As seen from Table 2, the models with the probabilities of examination learned from the query log (f_l^i and f_l^d) exhibit a better fit than the models parameterised by the heuristic functions on every subset of queries considered, except for the queries of length less than 10 characters, since they are not "typical" for the training dataset (median query length is 24.0 characters). In particular, f_l^d shows the best fitness to the whole dataset.

Overall, we conclude that adjusting the model parameters to the user behaviour in the session log leads to statistically significant improvements in the model's ability to "explain" the data in the dataset. The question now is whether this improvement leads the user model-based effectiveness metrics to have a higher alignment with the user preferences. We study this issue in the next section.

9.2 Metrics evaluation

Results of the two metric evaluation experiments are presented in Tables 3 and 4. The correlation coefficients are statistically significant with $p \leq 10^{-3}$, in each column $\triangle$ denotes a group of values which statistically significantly outperform other values and do not statistically significant differ from each other with $p \leq 0.05$. We applied Holm-Bonferroni adjustment for multiple comparisons when required. To report these results, we use the following notation. $pSaved(f)$ corresponds to a metric of the $pSaved$ family, parameterised by function f, e.g. $pSaved(f_l^i)$ is a metric obtained by assuming the model of user behaviour with probabilities of examination determined by function f_l^i. Similarly, $eSaved(f_{rr})$ is the $eSaved$ metric parameterised by f_{rr}.

Table 2: Log-likelihood of the user models parameterised by different examination probability functions. A higher log-likelihood corresponds to a better fit to the data. In each row all pairwise differences are statistically significant, except for the pairs labelled with $\diamond$.

	f_{rr}	f_{log}	f_l^i	f_l^d
All queries	-3.27	-3.84	-3.16	**-3.10**
Query frequency	f_{rr}	f_{log}	f_l^i	f_l^d
1 - 10	-3.66	-4.40	**-3.46**$\diamond$	**-3.47**$\diamond$
10-10^2	-3.54	-4.23	**-3.37**$\diamond$	**-3.36**$\diamond$
10^2-10^3	-3.25	-3.76	-3.14	**-3.12**
$> 10^3$	-2.01	-2.03	-2.19	**-1.92**
Query length	f_{rr}	f_{log}	f_l^i	f_l^d
1 - 10	-1.85	**-1.68**	-2.07	-1.86
11-20	-2.74	-2.94	**-2.74**$\diamond$	**-2.73**$\diamond$
21-30	-3.91	-4.84	-3.66	**-3.63**
> 31	-4.32	-5.57	-3.97	**-3.92**

MRR-n is a metric which is defined as the reciprocal rank of the submitted query q after submitting the first n characters of the query. For queries shorter than n characters we define *MRR-n* to be equal to *MRR-*$|q|$. Ranks higher than 10 are considered to be infinitely large (i.e., *MRR-1* and *MRR-3* are equal to 0 if the submitted query is ranked on positions below 10). The MRR metric is used in [18], though in a different evaluation scenario, as discussed in Section 5.

By *wMRR-n* we denote a modification of *MRR-n* weighted by the number of suggestions available for the corresponding query prefix, as used in [2].

MKS (Minimal Keystrokes) is a metric proposed by Duan et al. [12] to evaluate the query misspelling correction algorithms, which is defined as the minimal number of keystrokes a user has to perform in order to submit the query. The minimum is calculated among all possible interactions: the user can type the query's next character or can select the query from the list of suggestions using arrow keys. Despite the fact that it was proposed to evaluate misspelling correction algorithms, MKS can also be used to evaluate query suggestions. By definition, a better system should have lower *MKS*, hence the correlation between the user satisfaction indicator and *MKS* should be negative.

In Table 3, we report the correlation of the effectiveness metrics with the query suggestion success rate across different configurations. As we discussed in Section 7.2, this correlation implicitly includes comparison of different queries. Since for queries of different length the success rates differ (i.e. it is easier for the user to type a short query entirely) we also report the correlation for queries of different length. In

order to do this, the queries are split into four bins according to their length: less than 10 characters long; from 10 to 20 characters; from 20 to 30, and a set of queries longer than 30 characters. In addition, we report the correlation on the entire test dataset (length > 0).

On analysing Table 3, we observe that all eight combinations of the proposed metrics (*pSaved* and *eSaved*) and considered examination probability functions (f_{rr}, f_{log}, f_l^i, and f_l^d) perform better than the baseline metrics on each considered subset of queries ($p \leq 10^{-3}$). Comparing *pSaved* and *eSaved* we see that the *pSaved* metric is better correlated with the success rate. Moreover, this observation holds for both machine-learned functions f: $pSaved(f_l^i)$ outperforms $eSaved(f_l^i)$ and $pSaved(f_l^d)$ outperforms $eSaved(f_l^d)$ for each query length bin ($p \leq 10^{-3}$). Considering the *pSaved* metric, we notice that both machine-learned functions, f_l^i and f_l^d lead to a metric that is much better aligned with the user preferences than metrics induced by heuristic functions f_{rr} and f_{log} in each query bin ($p \leq 5 \times 10^{-3}$).

The experimental setting reported in Table 3 highlights that among the studied metrics, *pSaved is the most suitable metric to compare the effectiveness of the query suggestion mechanism across queries and configurations.*

In Table 4, we report the correlation of the difference in effectiveness metrics with the difference in query suggestion success rate. Similarly to Table 3, we split the queries into bins according to their length. Again, the proposed effectiveness metrics outperform metrics used in the literature when the entire dataset considered. However, in contrast to the previous experiment, *pSaved* parameterised by f_l^d is the second best performing metric, less effective than $eSaved(f_l^i)$.

As a result, we can conclude that *changes in ranking as measured by eSaved are better correlated with changes in user preferences* than that of the other considered metrics.

We conjecture that different outcomes of the experiments are caused by the fact that improvements in the ranking of long queries have a higher impact on the overall system performance than improvements for shorter queries and the last experiment reflects this fact. This seems natural from the user's point of view: since long queries are harder to submit, even small improvements in the ranking can be noticed. This observation is also supported by the fact that while $pSaved(f_l^i)$ dominates on short queries, its overall performance is worse than that of $eSaved(f_l^d)$.

On the other hand, as we discussed in Section 6, *eSaved* has higher ability to differentiate changes in the ranking of longer queries than *pSaved*, thus outperforming it in the last experiment (4).

To support this conjecture, we designed an experiment to answer the following question: which bin of queries has the highest impact on the correlation with the user preferences, as reported in Table 4? In order to address this, we artificially improved the alignment of *eSaved* with the user preferences by replacing its value by a linear combination of *eSaved* with the users' satisfaction rate, *SS*:

$$eSaved'(q, \epsilon) \leftarrow (1 - \epsilon)eSaved(q) + \epsilon SS(q)$$

We denote the correlation between differences in the $eSaved'_\epsilon$ "metric" and the success rate *SS* by $cor(\Delta SS, \Delta eSaved'_\epsilon)$. Further, we numerically calculate the partial derivative of the overall correlation value with respect to ϵ:

$$\left.\frac{\partial\left[cor(\Delta SS, \Delta eSaved'_\epsilon)\right]}{\partial \epsilon}\right|_{\epsilon=0} \quad (7)$$

Table 3: Correlation of the effectiveness metrics with the query suggestion success rate. The correlation was calculated on several subsets of the dataset containing queries with different length. Each experiment corresponds to a column. In each column △ denotes a group of values which statistically significantly outperform other values and do not statistically significant differ from each other.

	> 0	1 - 10	11 - 20	21 - 30	> 31
MRR-1	0.115	0.116	0.139	0.129	0.122
MRR-3	0.243	0.299	0.313	0.279	0.262
wMRR-1	0.094	0.052	0.103	0.104	0.099
wMRR-3	0.154	0.107	0.180	0.176	0.170
MKS	-0.470	-0.180	-0.601	-0.607	-0.875
$eSaved(f_{rr})$	0.816	0.522	0.731	0.871	0.902
$eSaved(f_{log})$	0.827	0.538	0.747	0.882	0.910
$eSaved(f_l^i)$	0.817	0.527	0.731	0.870	0.902
$eSaved(f_l^d)$	0.807	0.502	0.720	0.863	0.896
$pSaved(f_{rr})$	0.849	0.547	0.759	0.918	0.948
$pSaved(f_{log})$	0.838	0.524	0.747	0.916	0.947
$pSaved(f_l^i)$	0.857	**0.561**△	0.768	0.921	0.949
$pSaved(f_l^d)$	**0.860**△	**0.560**△	**0.773**△	**0.922**△	**0.950**△

Replacing *eSaved* by $eSaved'_\epsilon$ sequentially for queries from different bins, we estimate this derivative and report the results in Table 5. Informally, these results show how sensitive is the overall correlation to improvements in the metric on a particular bin of queries. From the table, we observe that the correlation is most sensitive to improvements in the metric on queries which are between 21 and 30 characters long. At the same time, a small improvement of the metric on queries longer than 10 characters leads to almost two times higher improvement in alignment with the user satisfaction than the same change on the queries shorter than 10 characters.

To summarise: on one hand, the *eSaved* metric is more discriminative than *pSaved* for longer queries by its design, and on the other hand the user behaviour (as well as alignment of the metric with it) is very sensitive to changes in longer queries. As a result, changes in *eSaved* demonstrate a better alignment with changes in the user preferences than that of *pSaved*, as can be observed in Table 4.

Overall, our experimental results suggest that the proposed metrics are better aligned with the user behaviour evidenced in the session log than other metrics considered, supporting the user model-based approach to build an effectiveness metric as being promising. Additionally, the experiments demonstrate that the most fruitful direction of improving the proposed metrics is to increase their alignment with user behaviour on long queries. Finally, *pSaved* is shown to be the most suitable metric to compare performances across queries, while changes in rankings measured by the *eSaved* metric correlate better with changes in the user preferences if the set of queries is fixed.

10. CONCLUSIONS AND FUTURE WORK

In this paper, we presented a novel model of the user interactions with the query suggestion mechanisms. We described a machine learning approach to adapt the model parameters to the user behaviour observed in a session log.

Using the described model, we introduced two user model-based evaluation metrics, *pSaved* and *eSaved*. The first metric, *pSaved* is defined as a probability of using a query

Table 4: Correlation of the *difference* in effectiveness metrics with the *difference* in query suggestion success rate. The correlation was calculated on several subsets of the dataset containing queries with different length. Each experiment corresponds to a column. In each column △ denotes a group of values which statistically significantly outperform other values and do not statistically significant differ from each other.

	> 0	1 - 10	11 - 20	21 - 30	> 31
MRR-1	0.557	0.868	0.983	0.924	0.427
MRR-3	0.576	0.931	0.989	0.930	0.486
wMRR-1	0.463	0.588	0.825	0.472	0.198
wMRR-3	0.459	0.549	0.613	0.166	0.229
MKS	-0.484	-0.713	-0.950	-0.805	-0.409
$eSaved(f_{rr})$	0.870	0.981	**0.991**△	**0.956**△	**0.580**△
$eSaved(f_{log})$	0.875	0.981	**0.991**△	**0.957**△	**0.586**△
$eSaved(f_l^i)$	**0.897** △	0.982	**0.990**△	**0.957**△	**0.583**△
$eSaved(f_l^d)$	0.839	0.972	**0.990**△	**0.957**△	**0.580**△
$pSaved(f_{rr})$	0.813	0.965	0.954	0.543	0.303
$pSaved(f_{log})$	0.739	0.933	0.506	0.072	0.202
$pSaved(f_l^i)$	0.839	0.978	0.981	0.819	0.398
$pSaved(f_l^d)$	0.878	**0.991** △	0.987	0.906	0.470

Table 5: Sensitivity of the correlation between the difference of the metric *eSaved* and the difference in the user preferences to small improvements in the metric.

Query length	1 - 10	> 10	11-20	21-30	> 30
$\frac{\partial cor(\Delta SS, \Delta eSaved'_\epsilon)}{\partial \epsilon}$	0.61	1.14	0.04	0.78	0.34

suggestion mechanism while submitting a query. *eSaved* equates to the normalised amount of keypresses a user can avoid due to the deployed query suggestion mechanism.

Our empirical study using a session log encapsulating 6.1M sessions demonstrated that the proposed metrics show the best alignment with the user preferences exhibited in the session log. Among the proposed metrics, our experiments showed that changes in *eSaved* are the most correlated with changes in the user preferences, hence it can be recommended as a target metric in the ranking optimisation. On the other hand, *pSaved* shows the highest correlation with the user preferences across queries. Finally, we experimentally demonstrated that the correlation of the changes in both metrics with the changes in the user behaviour is most sensitive to improvements on long queries, hence making future work in this direction promising.

Since this is the first work to model the user interaction with a query suggestion mechanisms, a variety of extensions can be considered. First of all, the users differ in typing speed, the device they use to select their query from the list of suggestions (e.g., they can use keyboard, mouse or touchpad with their desktop or use a mobile device). All these differences can lead to different patterns in the user behaviour. Consequently, a possible extension of the proposed user model can incorporate these features. This idea is closely related to the work of Carterette et al. [4] which discussed incorporating the user variability into the system-based evaluation paradigm.

Further study of the user behaviour can lead to improvements in the metric alignment with the user satisfaction indicator. However, there are other ways to improve the evaluation of query suggestions, e.g. one can extend a set of queries relevant to the user. As discussed by Zhu et al. [23] in the context of query recommendations, a possible approach is to study the relevance of all queries submitted in the session.

11. REFERENCES

[1] R. Baeza-Yates, C. Hurtado, and M. Mendoza. Query recommendation using query logs in search engines. In *Current Trends in Database Technology*, volume 3268 of *LNCS*, pages 588–596. 2005.

[2] Z. Bar-Yossef and N. Kraus. Context-sensitive query auto-completion. In *WWW '11*.

[3] P. N. Bennett, F. Radlinski, R. W. White, and E. Yilmaz. Inferring and using location metadata to personalize web search. In *SIGIR '11*.

[4] B. Carterette, E. Kanoulas, and E. Yilmaz. Incorporating variability in user behavior into systems based evaluation. In *CIKM '12*.

[5] O. Chapelle, S. Ji, C. Liao, E. Velipasaoglu, L. Lai, and S.-L. Wu. Intent-based diversification of web search results: metrics and algorithms. *Inf. Retr.*, 2011.

[6] O. Chapelle, D. Metlzer, Y. Zhang, and P. Grinspan. Expected reciprocal rank for graded relevance. In *CIKM '09*.

[7] O. Chapelle and Y. Zhang. A dynamic bayesian network click model for web search ranking. In *WWW '09*.

[8] C. L. Clarke, M. Kolla, G. V. Cormack, O. Vechtomova, A. Ashkan, S. Büttcher, and I. MacKinnon. Novelty and diversity in information retrieval evaluation. In *SIGIR '08*.

[9] C. Cleverdon. The Cranfield tests on index language devices. *Aslib Proceedings*, 19(6):173–194, 1967.

[10] K. Collins-Thompson, P. N. Bennett, R. W. White, S. de la Chica, and D. Sontag. Personalizing web search results by reading level. In *CIKM '11*.

[11] N. Craswell, O. Zoeter, M. Taylor, and B. Ramsey. An experimental comparison of click position-bias models. In *WSDM '08*.

[12] H. Duan and B.-J. P. Hsu. Online spelling correction for query completion. In *WWW '11*.

[13] G. E. Dupret and B. Piwowarski. A user browsing model to predict search engine click data from past observations. In *SIGIR '08*.

[14] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446, 2002.

[15] F. Radlinski and N. Craswell. Comparing the sensitivity of information retrieval metrics. In *SIGIR '10*.

[16] F. Radlinski, M. Kurup, and T. Joachims. How does clickthrough data reflect retrieval quality? In *CIKM '08*.

[17] M. Sanderson, M. L. Paramita, P. Clough, and E. Kanoulas. Do user preferences and evaluation measures line up? In *SIGIR '10*.

[18] M. Shokouhi and K. Radinsky. Time-sensitive query auto-completion. In *SIGIR '12*.

[19] A. Strizhevskaya, A. Baytin, I. Galinskaya, and P. Serdyukov. Actualization of query suggestions using query logs. In *WWW '12 Companion*.

[20] E. Voorhees. The philosophy of information retrieval evaluation. In *Evaluation of Cross-Language Information Retrieval Systems*, LNCS, pages 355–370. 2002.

[21] E. Yilmaz, M. Shokouhi, N. Craswell, and S. Robertson. Expected browsing utility for web search evaluation. In *CIKM '10*.

[22] Y. Zhang, W. Chen, D. Wang, and Q. Yang. User-click modeling for understanding and predicting search-behavior. In *KDD '11*.

[23] X. Zhu, J. Guo, X. Cheng, and Y. Lan. More than relevance: high utility query recommendation by mining users' search behaviors. In *CIKM '12*.