Term-by-Term Query Auto-Completion for Mobile Search

Saúl Vargas*
University of Glasgow
Glasgow, UK
saul.vargas@glasgow.ac.uk

Roi Blanco
Yahoo Labs
London, UK
roi@yahoo-inc.com

Peter Mika
Yahoo Labs
London, UK
pmika@yahoo-inc.com

ABSTRACT

With the ever increasing usage of mobile search, where text input is typically slow and error-prone, assisting users to formulate their queries contributes to a more satisfactory search experience. Query auto-completion (QAC) techniques, which predict possible completions for user queries, are the archetypal example of query assistance and are present in most search engines. We argue, however, that classic QAC, which operates by suggesting whole-query completions, may be sub-optimal for the case of mobile search as the available screen real estate to show suggestions is limited and editing is typically slower than in desktop search. In this paper we propose the idea of term-by-term QAC, which is a new technique inspired by predictive keyboards that suggests to the user one term at a time, instead of whole-query completions. We describe an efficient mechanism to implement this technique and an adaptation of a prior user model to evaluate the effectiveness of both standard and term-by-term QAC approaches using query log data. Our experiments with a mobile query log from a commercial search engine show the validity of our approach according to this user model with respect to saved characters, saved terms and examination effort. Finally, a user study provides further insights about our term-by-term technique compared with standard QAC with respect to the variables analyzed in the query log-based evaluation and additional variables related to the successfulness, the speed of the interactions and the properties of the submitted queries.

Categories and Subject Descriptors: H.3.3 [Information Storage & Retrieval]: Query Formulation

Keywords: query auto completion, word prediction, user models, query logs

1. INTRODUCTION

Query Auto-Completion (QAC) is a widely known and deployed mechanism that facilitates the task of formulating queries in search engines. In its basic version, a QAC mechanism presents to the user possible completions for the query the user is entering. In more elaborated cases, QAC techniques can also rephrase and correct some parts of the query that the user has already typed. Completions for queries are typically based on previous queries that have been submitted to the search engine, although some QAC systems are capable of constructing new queries that are adapted to the search context of the user [4, 22]. QAC has attracted an increasing level of interest from the research community with proposals that take into account personalization, context, time-awareness and user behavior in QAC systems.

Assisting users to formulate their queries becomes fundamental in the context of mobile search. The use of search engines in mobile devices such as smartphones and tablets has experienced a rapid growth in the last years [13]. Text input in mobile devices is generally much slower and clumsier than in traditional desktop environments. Therefore, QAC can be considered, in principle, even more useful in the context of mobile search. We argue however that classical QAC mechanisms give sub-optimal results in such context, as their direct adaptation to mobile search disregards problems related to the smaller screen real estate and the slower and clumsier text input and editing in mobile devices.

In this paper we propose a novel QAC mechanism in which the system suggests to the user one term at a time, rather than whole-query completions. We believe that this method is more suited to mobile search as it allows a faster exploration of suggestions and it adapts better to the particularities of text editing in mobile devices. This idea is inspired by the word prediction mechanisms present in predictive keyboards available in major mobile operating systems. Based on this, we describe a term-by-term QAC system implemented using a query-term graph, a data structure that allows the efficient storage and exploitation of query log data. We also adapt a prior user model for QAC to design effectiveness metrics for both standard and term-by-term QAC in offline evaluations.

Our proposal is tested under two different experimental settings. In the first setting, metrics based on a user model for QAC are used to evaluate data from the mobile query log of a commercial search engine following standard evaluation methodologies. In the second, we conduct a user study which compares our approach with standard QAC for the formulation of queries for a series of search tasks and provides insights into the validity of our proposal in terms of different user-engagement measures.

The rest of the paper is structured as follows. Section 2 reviews the related work on mobile search, query auto-

*Currently in Mendeley Ltd., London, UK.

WSDM 2016, February 22 - 25, 2016, San Francisco, CA, USA

ACM 978-1-4503-3716-8/16/02 ...$15.00.
DOI: http://dx.doi.org/10.1145/2835776.2835813

completion and word prediction. In Section 3 we provide a discussion of the inconveniences of standard QAC in mobile search and the advantages that a term-by-term QAC mechanism might bring. A system for term-by-term QAC based on a structure called query-term graph is presented in Section 4. Metrics to measure the effectiveness with respect to saved terms and examination effort of the standard and our new QAC methods are provided in Section 5. In Section 6 we present and discuss two different experimental settings that have been carried out to validate our idea. Finally, Section 7 offers the conclusions and future work.

2. RELATED WORK

This Section presents prior research related to our proposal. First, we comment on studies that have analyzed the characteristics of queries submitted from mobile devices. We review then the literature on QAC, for which we present a new technique. Finally, we briefly discuss the literature on word prediction, which inspired part of our work.

2.1 Mobile Search

In the last decade, a number of studies have been dedicated to understanding the trends and search behavior in mobile search, focusing particularly on the differences between desktop and mobile search regarding topics and characteristics of search sessions and queries.

For example, the work of Kamvar and Baluja [16] presents the analysis of Google search logs from 2005 in the US. Their study shows, among other interesting results, that better text input capabilities in mobile devices result in longer queries and that search sessions are sensibly shorter (1.6 queries per session) than in desktop search (>2 queries per session). In a more recent study, Kamvar and Baluja [17] show the evolution of search behavior over time by analyzing search logs from 2007. This study indicates an increase in the length of queries (from 2.3 mean terms per query in 2005 to 2.7 in 2007) and in the percentage of queries with one accepted result (from 10% to 50%). In both studies, adult content is the most popular content category for mobile searchers.

Baeza-Yates et al. [2] compared the characteristics of mobile and desktop queries submitted to the Yahoo Japan search engine in 2006. Among other observations, the authors found a close correspondence between the number of terms of queries in both scenarios, but a larger discrepancy between the number of characters per query (7.9 in desktop and 9.6 in mobile). They also showed that online shopping, health and sports are the most popular search topics in that market.

Church et al. [8] analyzed the search habits of European users in 2005, revealing several specific details about their search patterns. For example, they report that 23% of the queries are modifications of previously submitted queries, of which half of them correspond to single term substitutions. Later on, the same authors extended in [7] their study by including a much larger sample of queries and an analysis of click-through data from Google mobile search.

Studying the particular nuances of mobile user behavior might yield improvements in mobile search. For instance, Song et al. [26] introduced a study performed on the Bing mobile search engine in 2012. Based on their results regarding the search behavior in terms of time distribution, search locality, query categories and other characteristics, the authors proposed a new framework to improve ranking on mobile search by incorporating mobile-specific features and transferring knowledge from desktop search relevance to mobile search.

2.2 Query Auto-Completion

Query auto-completion systems have attracted an increasing level of interest by both the research community and industry. Data for such systems comes from past user queries. In its simplest variant, QAC algorithms rank completions by their popularity in the past, an algorithm which we refer to as most-popular completion [3] throughout this work. However, recent research has focused on going beyond such basic mechanisms and taking into account the search context and the preferences of users. For example, Bar-Yossef and Kraus [3] proposed a context-sensitive QAC algorithm that considers the current browsing session in order to suggest completions related to the topics of the session. Shokouhi and Radinsky [25] presented a time-sensitive approach in which query suggestions are ranked by *forecasted* frequency adapted to the moment the query is formulated, instead of being ranked by their past frequency. In a similar direction, Whiting and Jose [27] have proposed a QAC method that takes into account query recency and emergence as improvements over plain time-independent query frequency. Additionally, user demographics have been considered by Shokouhi [24] to personalize the ranking of query suggestions. In our proposal, we take a different research direction in which, instead of improving QAC in terms of personalization, context and time-awareness, we propose a novel manner of suggesting query predictions that is better adapted to the characteristics of mobile devices.

Following a different research direction, other authors have studied how users interact with QAC systems as a way to provide better measurements of the effectiveness of such systems. For example, Kamvar and Baluja [18] studied how users interact with QAC systems in mobile search and, as a result, reported interesting usage patterns. Kharitonov et al. [19] proposed a user model for QAC based on cascade-based metrics [6]. By learning the model parameters from a query log of Yandex, the authors derive two different metrics for the evaluation of QAC effectiveness that show a higher correlation with success rate than prior metrics proposed for the task. Li et al. [20] proposed a two-dimensional click model that, instead of being used to evaluate the effectiveness of QAC systems, is used to improve the ranking of suggestions. The query log-based evaluation of our proposal is based on these two previous papers. Finally, Mitra et al. [21] and Hofmann et al. [14] studied user interactions with the QAC system of Bing. Our user study adapts some of the metrics for query formulation analyzed in [14].

2.3 Word Prediction

Word prediction technologies focus on the tasks of completing or predicting the next word in general text input. Darragh et al. [9] described an early proposal of this technology based on a new interface that partially automates typed input and increases the speed and ease of text input. Research on word prediction has typically been oriented towards assisting users with disabilities [12]. However, with the widespread adoption of mobile devices, this technology has experimented a massive adoption with the so-called predictive keyboards, present nowadays in modern mobile operating systems [23], where word prediction is one of the most relevant features. The success of predictive keyboards is the main motivation for our proposal for term-by-term QAC.

3. APPLYING NEXT-TERM PREDICTION TO QUERY AUTO-COMPLETION

Mobile search, despite its ever increasing usage, is limited by the characteristics of text input and display of mobile platforms. Specifically, text input in mobile devices such as smartphones and tablets is generally much slower and clumsier than in personal computers, as users have to rely on smaller, on-screen keyboards. Moreover, text display is also limited by the comparably smaller screen real estate of mobile devices, which further complicates text input and editing especially in single-line search boxes as long queries may not be displayed completely and text selection is, by our own experience, a difficult task in such devices. In order to facilitate the task of query formulation in such scenario, we identify two types of technologies that help users formulate queries in mobile search.

On the one hand, QAC techniques are the archetypal system for query formulation assistance, which, in principle, could be considered even more useful in mobile search than in the case of desktop search given the previously described limitations for entering text in smartphones and tablets. We find however that suggesting to the users completions for whole queries may be sub-optimal in this case, as we identify a number of problems related with the described characteristics of text input and display in mobile devices after having tested the mobile versions of several commercial web search engines, such as Bing, Yahoo and Google. These problems can be summarized as follows:

- Given the limited screen real estate (notably in the case of smartphones), QAC systems are restricted to show a small set of query completions and, moreover, when a completion does not fit in a single line, the explored interfaces of web search engines take one the following sub-optimal approaches: showing a fragment of the completions (thus making more difficult its reading) or display them in several lines (so that less different completions are shown at the same time).
- Text reading in mobile devices is cumbersome as well, so one can expect that users would have to spend some time to evaluate whole-query suggestions, which ultimately leads to ignoring the suggestions.
- The user may not be interested in the exact whole completion, but a fraction or a variation of it. However, as previously pointed out, text editing in mobile interfaces (especially in the general case of touchscreens) is generally a clumsy process and users may be reluctant to accept partially relevant suggestions that would require editing.

On the other hand, term prediction systems, which are a related but different technology that is applied to generic text input in mobile devices, have been found to be a useful feature implemented by predictive keyboards to cope with the aforementioned problems for general text input in mobile devices. Such systems provide two types of predictions: current term prediction, in which the keyboard shows completions for the term that the user is typing, and next term prediction, in which a small number of suggested terms (typically three) are presented to the user as possible next terms after the last term entered by the user. In this work, we are interested specifically in the second type of prediction. Next term predictions, as found in predictive keyboards, are typically based on common sequences of words that may take into account the writing patterns of the users and other data such as address books, thus not being specifically crafted for the purpose of query formulation. The success of next term prediction technologies in predictive keyboards is manifested by their inclusion as default input methods in all major mobile operative systems and also in the app markets of these, where the number of installations of alternative predictive keyboard apps are counted by millions.

We propose in this work a method for query formulation assistance that lies at the intersection of QAC and next term prediction. The idea is to re-define QAC by suggesting, rather than whole completions, one term at a time to formulate the intended query. Note that we do not propose to replace QAC by predictive keyboards, but to re-purpose algorithms for standard QAC algorithms to provide completions based on single terms. We believe that this approach, that we refer to as term-by-term QAC, solves the limitations found for standard QAC for the following reasons:

- As only single terms are suggested to the users, space limitations are drastically reduced.
- The time required to read suggested terms is significantly smaller than in the case of whole completions, which should increase the usage of such technologies and shorter interaction times.
- As only one term at a time is presented to the user, problems related with editing suggestions are also partly solved, as the need of editing the suggestion will be generally smaller and, even in that case, editing the suggestion will require less effort than editing whole completions.

As already pointed out, a term prediction mechanism deals with two closely related but different prediction tasks: completing the current term and predicting the next one. In order to simplify our analysis, in this study we focus exclusively on the second case. Focusing on the second case is reasonable as prior studies have found that QAC systems have a higher chance of being used after complete terms have been entered [20, 21]. Moreover, current term completion can also be seen as a special case of next term prediction, meaning that the conclusions of focusing on one type of prediction can be generalized to the other.

4. THE QUERY-TERM GRAPH

We now propose an efficient method to implement a term-by-term QAC system based on previous queries submitted to a search engine. Relying on previous queries is a standard way of implementing QAC by major search engines [3] and most proposals from the state of the art follow this approach [15, 24, 25, 27]. In particular, our solution makes use of queries gathered from a commercial search engine and stores them in a graph structure that we refer to as query-term graph, which allows an efficient way of storing and retrieving millions of queries. We describe next our query-term graph, a procedure to build it from previously submitted queries and two algorithms for suggesting the most popular completion in standard and term-by-term QAC.

4.1 Definition

The query-term graph is a directed, weighted graph that stores millions of queries in a compact way. It is composed of three types of nodes: a unique root node with no incoming edges that represents the start of every query, intermediate nodes that contain query terms and end-of-query nodes with no outgoing edges that mark the end of a query. All the paths of the query-term graph starting from the root node

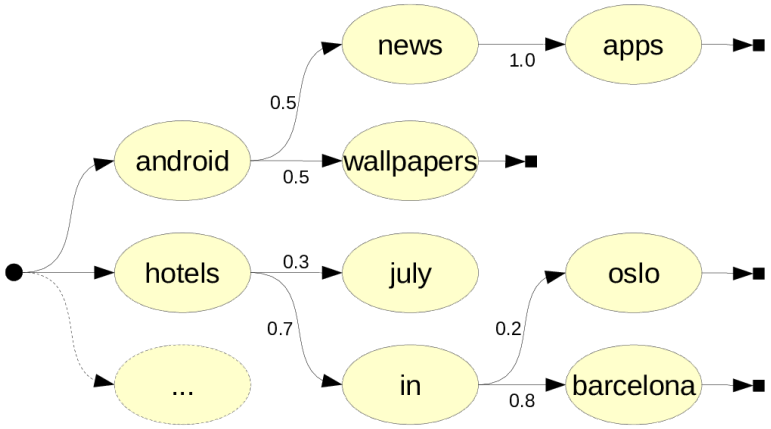


Figure 1: Example of a query-term graph. The root node (circle) is located on the left, the query terms occupy intermediate positions and the leaf nodes (squares) mark the end of queries.

and ending in a leaf node represent queries that the QAC system can provide as completions. Each edge has a weight that represents the probability of the paths that contain it to be the completions of the sub-path that starts from the root node and ends in the source node of the edge. See Figure 1 for an example of a query-term graph. In this example, we can see that queries such as `android news apps` and `hotels in oslo` are stored in the graph as paths from the root to leaf nodes.

Although the query-term graph has some resemblance with the well-known tries [11], it has significant differences nonetheless. First, the intermediate nodes of our graph store terms rather than characters. Second, ours is a weighted graph as we need weights to calculate the likelihood of completions. Third, our query-term graph is not necessarily restricted to a tree-like structure since we may allow intermediate nodes to share branches. That can be useful, for instance, when different starting terms share many suffixes, such as the case of synonyms or entities of the same category. See Figure 2 for an example of two nodes having common branches. In this case, both `android` and `iphone` share suffixes such as `apps` or `wallpapers`.

4.2 Construction

The construction of a query-term graph using a query log can be costly since millions of queries need to be processed and added to this structure. We briefly detail here the steps involved in an efficient procedure for building a query-term graph for millions of queries submitted to a commercial search engine. To illustrate the process, we consider the small example of a query log in Table 1 that generates the query-term graph of Figure 1.

In its most simple configuration, such as to implement algorithms to provide the most-popular completion that we detail in the next section, only the set of submitted queries in the query log is required. After an initial pre-processing to discard anomalous queries and queries composed of only one term – which our term-by-term mechanism has no use for –, a first step involves counting the number of times each unique query has been repeated. After that, we determine the structure of the query-term graph by identifying all the different paths starting from the root node together with the aggregated frequency of each of them. We do this by splitting every query by its terms and extracting all the different prefixes, that is, the possible paths of the graph starting from the root node. In this step we also assign an unique id to every unique path, including the zero-length path of the root node, with id 0. For instance, for the queries of the query-term graph of Figure 1, we would have generated the list of sub-paths with their respective aggregated frequencies and ids of Table 2.

Once the paths of the query log have been determined, the next step consists in determining the nodes and edges of the graph based on the extracted sub-paths. The nodes are determined by the id and last term of every sub-path extracted in the previous step. The edges are in turn extracted by finding for each path its immediate sub-path (the longest path which is a sub-sequence of the target one), a process that can be efficiently performed by having the list of paths in lexicographical order. In our example, the resulting edges together with their weights would be, following the format `(src, dest; count)`, the following: `(0, 1; 10)`, `(1, 2; 5)`, `(2, 3; 5)`, `(1, 4; 5)`, `(0, 5; 100)`, `(5, 6; 70)`, `(6, 7; 56)`, `(6, 8; 14)` and `(5, 9; 30)`. A normalization of the weights of the outgoing edges of each nodes so that they sum 1 may also be performed for convenience. Note that all the previous steps can be carried out by means of scalable, distributed programming models such as MapReduce [10]. Finally, the resulting sets of nodes and edges that form our query-term graph can be efficiently stored with help of compression graph techniques such as the Webgraph framework [5].

As mentioned, the previous steps describe the construction of a simple query-term graph – more precisely in this case, a directed tree – based on previous queries with weights derived from the popularity of each query. However, additional steps could be considered to enrich the construction of the graph. For instance, other indicators for the quality of the queries such as the number of clicked results for each query, could be easily incorporated to provide better completions. Also, identifying common suffix patterns between queries could be used not only to reduce the size of the graph (see Figure 2), but also to infer previously unseen query completions.

4.3 Usage

The standard and more direct approach for QAC consists in suggesting for a given query prefix its most popular completions [3]. Many of the state-of-the-art proposals build on this idea and refine it to make personalized, context-aware recommendations. We now describe how most popular com-

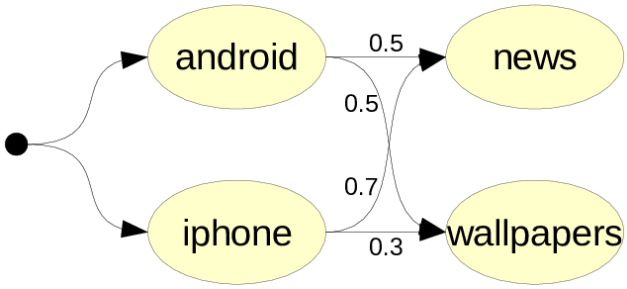


Figure 2: Example of the of common branches of nodes in the query-term graph.

pletions, either in standard or term-by-term QAC, can be implemented using our query-term graph.

On the one hand, the term-by-term most popular completion using the query-term graph basically consists in keeping track of the path followed by the user when submitting a query, as defined by the current terms of it, and presenting suggestions in the form of the N terms of the destination nodes of the outgoing edges with higher weights, possibly including the special end-of-query leaf nodes to indicate to the user that the current state of the query may be sufficient.

On the other hand, despite not being its intended purpose, the query-term graph can also be used to implement a standard most-popular completion mechanism suggesting entire queries. As in the term-by-term case, the algorithm needs to trace the path followed by the current state of the query. For a given query prefix, the suggestions in the form of whole-query completions are extracted by searching in the graph the continuing paths to leaf nodes whose probabilities – as product of the probabilities of their edges – are among the N highest. Note that this procedure can be efficiently implemented by means of a pruned depth-first search in which candidate branches whose maximum probability is lower than the current candidate completions are discarded.

By comparing the two previous algorithms, it is clear that the algorithm for standard most-popular completion – even with optimizations – is costlier than its term-by-term counterpart, in which the search is reduced to select the immediate edges with highest probabilities. Although standard QAC is generally implemented by means of alternative data structures and algorithms, possibly more optimal for that task, we claim that the computational cost of our term-by-term QAC is generally smaller than the standard whole-query approach given its simplicity. Therefore, even though computational efficiency of QAC mechanisms is not a goal of this paper, we believe our approach is appealing also in this respect.

5. A USER MODEL FOR TERM-BY-TERM QUERY AUTO-COMPLETION

The differences between the standard and term-by-term QAC methods can be further explained by means of a user model that explains how users examine and select the suggestions provided by each method. In this section we provide such user model for both alternatives and present three derived metrics, namely saved characters, saved terms and effort, to evaluate QAC using query log data. As stated in Section 3, in this paper we take the simplification of only dealing with the problem of providing suggestions after the user has finished entering a whole term of the query. This decision is reflected in the formulation of the user model we describe,

count	path
5	android news apps
5	android wallpapers
56	hotels in barcelona
14	hotels in oslo
30	hotels july

Table 1: Example of a really small query log.

id	count	path
1	10	android
2	5	android news
3	5	android news apps
4	5	android wallpapers
5	100	hotels
6	70	hotels in
7	56	hotels in barcelona
8	14	hotels in oslo
9	30	hotels july

Table 2: Possible sub-paths of the queries in Table 1.

although a generalized version that takes into account the current term completion case can be easily included.

Recently, Kharitonov et al. [19] proposed a user model for the evaluation of QAC in which the probability of a user examining a suggestion depends on two dimensions, namely the position of the suggestion in a given list of suggestions and the length of the current formulated query. In a similar manner, the work of Li at al. [20] explores these two-dimensions in a click model for QAC. These models build on two basic assumptions. First, they assume that a user, while typing a query, examines every completion suggested by the QAC mechanism with a probability that depends on the current state the entered query (e.g. the user might examine more suggestions at the beginning of the formulation process) and the position of the completion in the list of suggestions (typically with decreasing probability with the position) but does not depend on the suggestions for previous prefixes (i.e. even if the previous suggestions were not satisfactory, they do not influence the examination probability of the next suggestions). Second, it is assumed that only completions that match the query that the user has in mind may be accepted.

Following these assumptions, we consider the random variable E_i^j that models if the j-th suggestion provided after the i-th term of the query is examined. By denoting as $j = m(i)$ the position of the suggestion that matches the intended query after the i-th term (in the case that none of the suggestions matches the intended query, then $m(i) = \infty$), we consider then the probability $p(E_i^{m(i)})$ of providing a satisfying suggestion after i terms of the intended query have been entered. This probability plays however different roles depending on the method for QAC. In the case of standard QAC, where suggestions are whole-query completions, the probability of examination of a given suggestion after i terms have been entered has to take into account the fact that the matching completion may have been shown in previous suggestions. In case the matching completion was shown and selected, the suggestions at query position i are not examined. This way, in standard QAC the probability $p(M_{STD}^i)$ of matching the intended query after i terms have been entered is determined by the following equation:

$$p(M_{STD}^i) = p(E_i^{m(i)}) \prod_{i'<i} \left(1 - p(E_{i'}^{m(i')})\right) \qquad (1)$$

In the case of term-by-term QAC, suggestions only consist

of predictions for the next term of the query. Therefore, the fact that previous suggestions have been accepted does not affect, as assumed by our model, the examination probability at position i, so the matching probability $p(M^i_{TBT})$ at query position i is simply given by the probability of examination:

$$p(M^i_{TBT}) = p(E_i^{m(i)}) \quad (2)$$

In prior work [19, 20] the probability distribution of examination $p(E^j_i)$ for standard QAC was estimated with query log data. As data for term-by-term QAC is not available to date, we rely in this work on a fixed probability $P(E^j_i) = 1/(j+1)$ which is reciprocal to the position of each completion of the suggestion list for both standard and term-by-term QAC. While not taking into account the horizontal dimension, i.e. the state of the entered prefix of the query, the experiments of [19] show that this reciprocal estimation is only moderately worse than models learned from user interactions and, moreover, is in line with the commonly considered MRR metric used to evaluate QAC algorithms found in many offline experiments of prior work [3, 15, 20, 24, 25, 27]. Note that we assume the same probability of examination for both methods, although we could consider higher probabilities for the term-by-term completions as they are shorter than their whole-query counterparts.

Similarly to [19], we measure with our model the effectiveness in terms of the number of characters, as well as terms, that the QAC methods save the user to enter. Given our previous assumptions, the proportion of characters or terms that a standard QAC saves entering are given, respectively, by the following equations:

$$CS_{STD} = \frac{1}{c(q) - c(q_1)} \sum_{i=1}^{t(q)-1} (c(q) - c(q_i))\, p(M^i_{STD}) \quad (3)$$

$$TS_{STD} = \frac{1}{t(q) - 1} \sum_{i=1}^{t(q)-1} (t(q) - i)\, p(M^i_{STD}) \quad (4)$$

where $t(q)$ is the number of terms of the intended query q, $c(q)$ is the number of characters and q_i is the sub-query until the i-th term. The same magnitudes for term-by-term QAC are in turn given by the following equations:

$$CS_{TBT} = \frac{1}{c(q) - c(q_1)} \sum_{i=1}^{t(q)-1} (c(q_{i+1}) - c(q_i))\, p(M^i_{TBT}) \quad (5)$$

$$TS_{TBT} = \frac{1}{t(q) - 1} \sum_{i=1}^{t(q)-1} p(M^i_{TBT}) \quad (6)$$

Another variable of interest, related but different to the number of saved terms would be the effort involved in examining the suggestions provided. We define it as the expected number of suggestions examined by the user normalized by the query length. In the case of standard QAC this quantity is given by:

$$EF_{STD} = \frac{1}{t(q) - 1} \sum_{i=1}^{t(q)-1} \sum_{j \le m(i)} p(E^j_i) \prod_{i' < i} \Big(1 - p(E_{i'}^{m(i')})\Big) \quad (7)$$

while for term-by-term QAC is determined by:

$$EF_{TBT} = \frac{1}{t(q) - 1} \sum_{i=1}^{t(q)-1} \sum_{j \le m(i)} p(E^j_i) \quad (8)$$

These three metrics and their formulation for both QAC mechanisms are used in the query log-based experiments in the next section. Additionally, we will provide equivalent measurements with real user data.

6. EXPERIMENTS

The observations made in Section 3 suggest that a term-by-term approach to QAC might benefit the formulation of queries in mobile search. To test the validity of this proposal, we have conducted two different experiments. The first one is a query log-based experiment in which one month data from a commercial search engine is evaluated following standard practices in the related work [3, 19, 20, 27] in which both standard (STD) and term-by-term (TBT) most-popular completion algorithms are evaluated with the metrics defined in Section 5. The second experiment consists in a user study in which a set of users was asked to perform a series of query formulation tasks with the assistance of both methods in their mobile devices.

6.1 Mobile Query Log-Based Evaluation

6.1.1 Setup

We have performed an evaluation of the compared QAC methods using a query log from a commercial search engine. In particular, our query log for this evaluation consists of queries submitted from mobile devices to Yahoo, a major US search engine. We collected a set of queries submitted from mobile devices in one month period. For efficiency reasons, we have discarded queries having more than 8 terms. The data from the first 20 days was taken as training to build the query-term graph, while a random sample of 41 million queries from the remaining period has been used for test.

We have created two implementations of the systems detailed in Section 4, namely the most-popular completion for both standard and term-by-term QAC, using the data from the training subset. Although more complex QAC mechanisms have been proposed in the state of the art (see Section 2), many experimental evaluations acknowledge that the simpler most-popular completion approach offers competitive performance at the smallest implementation cost [3, 15, 20, 24]. Indeed, we believe that, for the purpose of our work, the improvements of more elaborate QAC techniques do not alter the comparison between the standard and our term-by-term QAC variants.

For testing purposes, an alternative implementation of the term-by-term QAC in which terms are picked from whole-query completions in the order determined by the rank of the latter was also considered. Even when this approach outperforms the standard QAC with respect to the considered metrics, it performs worse than our proposed, more efficient term-by-term QAC from Section 4. Thus, we do not include the results for this alternative system.

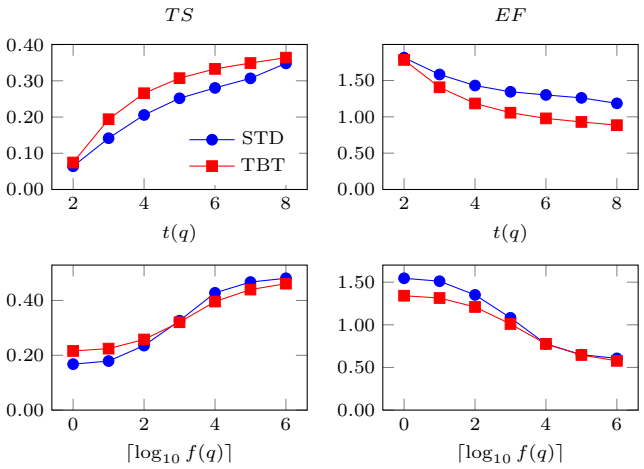
The evaluation of the remaining two systems is based on common assumptions found in prior work [3, 19, 20, 27]. In particular, each query in the test both serves as simulation of user behavior – our simulated user enters sequentially the terms of the query to activate the QAC mechanism – and as gold standard – only suggestions matching the rest of the query are considered as satisfactory. In this setting, the suggestion lists are composed of $N = 10$ completions each. As previously stated in Section 5, we assume that the probability of examining a suggested completion is reciprocal to its position (more precisely, $p(E^j_i) = 1/(j+1)$) for both standard and term-by-term QAC.

6.1.2 Results

The outcomes of our evaluation are found in Tables 3 and 4. Since our two compared most-popular completion

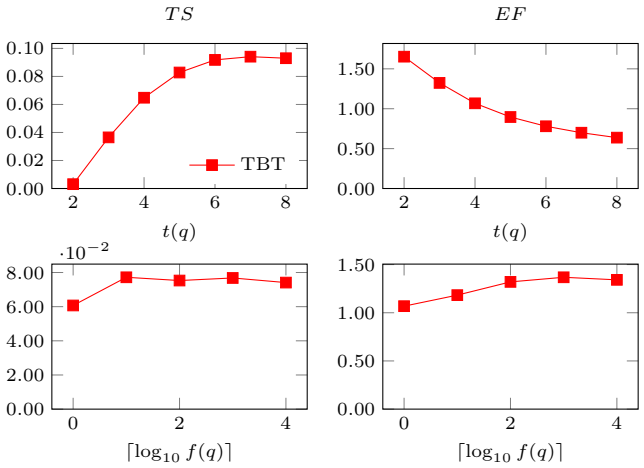
			CS		TS		EF	
		N	STD	TBT	STD	TBT	STD	TBT
		17,853,330	0.1759	0.2111	0.1761	0.2216	1.5206	1.3225
$\mathbf{t(q)}$	**2**	3,573,019	0.0641	0.0745	0.0641	0.0745	1.8143	1.7833
	3	5,081,503	0.1414	0.1859	0.1416	0.1939	1.5825	1.4066
	4	4,542,236	0.2021	0.2504	0.2057	0.2657	1.4311	1.1837
	5	2,657,039	0.2510	0.2913	0.2518	0.3073	1.3444	1.0566
	6	1,227,555	0.2858	0.3147	0.2804	0.3329	1.3014	0.9787
	7	517,222	0.3189	0.3312	0.3068	0.3490	1.2609	0.9298
	8	254,756	0.3633	0.3475	0.3481	0.3639	1.1857	0.8850
$\lceil\log_{10}\mathbf{f(q)}\rceil$	**0**	9,838,824	0.1672	0.2047	0.1677	0.2158	1.5457	1.3409
	1	7,100,953	0.1787	0.2138	0.1789	0.2243	1.5110	1.3141
	2	833,071	0.2393	0.2523	0.2353	0.2577	1.3507	1.2092
	3	75,054	0.3292	0.3168	0.3262	0.3205	1.0822	1.0096
	4	5,149	0.4286	0.3926	0.4278	0.3960	0.7754	0.7760
	5	265	0.4671	0.4371	0.4672	0.4397	0.6497	0.6463
	6	14	0.4838	0.4595	0.4810	0.4613	0.6065	0.5774

Table 3: Results for the set of previously seen queries in the mobile query log-based evaluation. Corresponding plots (for TS and EF) on the right.



		N	$\mathbf{CS_{TBT}}$	$\mathbf{TS_{TBT}}$	$\mathbf{EF_{TBT}}$
		24,311,303	0.0517	0.0618	1.0757
$\mathbf{t(q)}$	**2**	2,852,182	0.0031	0.0031	1.6524
	3	5,172,474	0.0306	0.0365	1.3235
	4	5,781,961	0.0549	0.0648	1.0679
	5	4,583,313	0.0703	0.0828	0.8961
	6	2,962,515	0.0766	0.0918	0.7802
	7	1,819,564	0.0766	0.0941	0.6993
	8	1,139,294	0.0743	0.0930	0.6377
$\lceil\log_{10}\mathbf{f(q)}\rceil$	**0**	22,761,496	0.0507	0.0607	1.0683
	1	1,542,835	0.0672	0.0772	1.1828
	2	6,718	0.0666	0.0753	1.3203
	3	245	0.0671	0.0768	1.3680
	4	9	0.0784	0.0741	1.3415

Table 4: Results for the set of previously unseen queries in the mobile query log-based evaluation. Corresponding plots (for TS and EF) on the right.



systems cannot suggest queries that are not present in the training set [22], we first divide the random sample of the test set into queries that appeared in the training set (Table 3) and new queries (Table 4). In particular, as the standard most-popular completion algorithm is not able to provide successful suggestions according to our evaluation methodology, Table 4 does not include its outcomes, as they are, as expected, completely unsatisfactory. For the remaining outcomes, we show the results of the characters saved (CS), terms saved (TS) and effort (EF) metrics according to the global average for unique queries and, also, detailed according to two different criteria: the number of terms of the final queries $t(q)$ and the frequency of them $f(q)$.

The results for previously seen queries in Table 3 show some of the advantages of our term-by-term approach, namely an average higher number of saved characters and terms with a smaller effort in terms of the number of suggestions explored. This can be already observed in the average over the almost 18 million queries in our test subset. A more detailed analysis can be done with respect to the number of terms and frequency of the queries.

Regarding the number of terms of the queries, we can first observe that the normalized number of saved characters and terms increases with the length of the final query for both compared approaches, while the relative effort decreases. This is expected as the longer the query is, the more specific it is so that popular existing completions match it. When comparing between approaches, the global trend shows that the term-by-term approach provides in general more saved characters and terms with less effort. Regarding CS and TS, we see that both methods have a similar performance for queries of size 2, although the gap between them is larger for longer queries with the exception of the queries with 8 terms, in which the performance is again similar. This is not the case of the effort metric, in which the relative improvements of our method over the standard QAC increase along with the length of the queries.

When we analyze our results in terms of the frequency of the queries, these illustrate the performance of both compared approaches for different levels of query popularity, showing in particular that our method works better than the standard one for long-tail queries and similarly for more popular ones. A common trend for both methods is that, the more popular the query is, the better both systems work in terms of the considered metrics. This outcome is expected as both systems are based on providing the most popular completions in the past. When comparing between systems by query frequency by saved characters and terms, the results indicate that our term-by-term method is better that the standard most-popular completion for not so popular queries (frequency

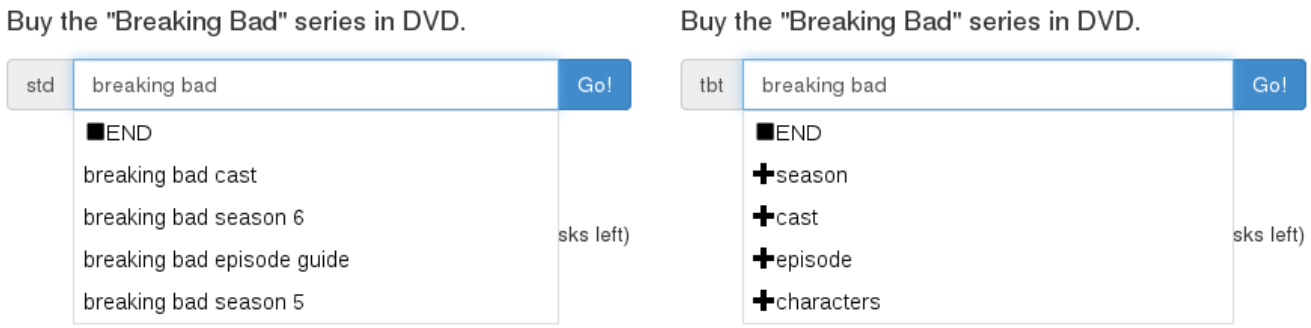


Figure 3: Examples of the search interface for the user study for the same search task with standard (left) and term-by-term (right) QAC.

less than or equal to 100 times), while for highly popular (and easier to predict) queries the standard most-popular completion method is slightly better. In any case, the effort involved in the term-by-term method is always smaller than in the standard approach, although the differences are smaller the more frequent the query is.

Finally, in Table 4 we show the results of our term-by-term most-popular completion approach for queries in the test subset that did not appear in the previous training subset. As previously explained, the results for the standard most-popular completion are not included here as this method, according to this offline evaluation methodology, is unable to provide satisfactory suggestions, given that their suggestions never match completely the intended query. However, our term-by-term method is able to assist the user even in such unfavorable scenario as it is still able to suggest terms that matches prefixes of unseen queries. As the results show, the number of saved characters and terms in this setting is much smaller than the values obtained from the evaluation previously seen queries. For instance, for unseen queries of two terms our system is practically incapable of providing any valid suggestion, although the success ratio improves for longer queries. In fact, for longer queries, our term-by-term system seems to be of some help as it successfully suggest terms in the beginning of longer queries. In terms of effort, we see smaller values than that of the case of seen queries. This is expected, since as soon as the system reaches a sub-query outside the query-term graph it stops providing suggestions. Contrary to the results in the previously seen queries, the frequency of the queries does not seem to affect significantly the number of saved characters and terms of them, although it has an influence on the effort, which increases for more popular queries.

6.2 User Study

6.2.1 Setup

We provide an additional perspective of the performance of our term-by-term approach in terms of a controlled user study. Based on data from the same query log for a longer period of three months, we have built another query-term graph and implemented a web-based QAC interface adapted to mobile browsers. It implements both standard and term-by-term most-popular completion algorithms showing $N = 5$ possible completions after a whole term has been entered. The interface logs every relevant interaction of the users with the system: when suggestions were shown, if they were rejected or accepted (and the position of the accepted completion) and when the user submitted the resulting query. In the case of standard most-popular completion, we allowed the users to keep formulating their queries even if they accepted a whole-query completion. See Figure 3 for examples of the query formulation interface. Users were asked to access the interface with their mobile devices and complete a series of query formulations for the 20 tasks in Table 5. The formulations were shuffled to counter-balance any order effect. Users were not aware of which completion mechanisms were being tested, and they had freedom to ignore query assistance if they wanted to. Each task was repeated for each QAC method. In total, we recruited a total of 25 evaluators who performed a total of 968 query formulations (some users did not complete all the formulations). From these formulations, we discarded one term queries and formulations that were not performed using both methods for the same search tasks, resulting in 350 pairs of formulations where both compared methods have been used by the same user to complete the same task. Despite its small scale, this study allows us to measure certain variables of interest of QAC systems that cannot be otherwise estimated by a query log-based evaluation.

#	description
1	Should I buy and iPad or a Samsung tablet?
2	Buy the "Breaking Bad" series in DVD.
3	Recipe for strawberry cupcakes.
4	What to see in Barcelona.
5	Upcoming Taylor Swift concerts in Europe.
6	When does the fifth season of Game of Thrones start?
7	Price of the tickets of The Lion King musical in Broadway.
8	Will Hillary Clinton run for president in 2016?
9	Cast of The Expendables 3.
10	What is the migration route of the monarch butterfly?
11	The weather forecast in Los Angeles for the next week.
12	Price of Tesla Motor cars.
13	Last news about the Ukraine crisis in the last month.
14	Which countries are affected by the ebola outbreak?
15	Find funny pictures of cute cats.
16	How to take a screenshot on Android phones.
17	Find a cheap laser print.
18	Resources for learning arabic language.
19	What is the highest mountain in California?
20	How many Starbucks are there in San Francisco?

Table 5: Search tasks of the user study.

From the recorded interactions of our test users we have measured a series of variables related to the effectiveness of both QAC completion approaches. First, we took some measurements related to the operation of the QAC methods themselves, namely the number of times they were activated (`suggestions`), the average ratio of accepted suggestions (`accepted`) and the average position of the accepted com-

	t	event	query
STD	0	suggest	ipad
	1,571	reject	ipad vs
	1,890	suggest	ipad vs
	10,353	reject	ipad vs samsung
	10,355	suggest	ipad vs samsung
	15,432	accept (3)	ipad vs samsung galaxy tablet
	15,444	suggest	ipad vs samsung galaxy tablet
	17,475	submit	ipad vs samsung galaxy tablet
TBT	0	suggest	ipad
	2,048	reject	ipad vs
	2,050	suggest	ipad vs
	6,969	accept (2)	ipad vs samsung
	6,977	suggest	ipad vs samsung
	8,853	accept (1)	ipad vs samsung galaxy
	8,860	suggest	ipad vs samsung galaxy
	10,213	accept (2)	ipad vs samsung galaxy tablet
	10,235	suggest	ipad vs samsung galaxy tablet
	11,366	submit	ipad vs samsung galaxy tablet

Table 6: Example of a query formulation task with both methods for a same user. For each formulation process, we indicate the sequence of events starting from the first activation of the QAC mechanism with the elapsed time and the state of the query. For acceptance of suggestions, the position of the accepted completion is indicated.

pletion (`accept_pos`). The second block of measurements is related to the benefits perceived by the user which were measured by our query log-based evaluation, namely the number of saved terms (`terms_saved`), the average number of completions examined for each suggestion assuming that users examined sequentially all listed completions until finding the right one (`effort`) and, also, the number of saved characters (`chars_saved`). The third set of measurements considers the time (in seconds) spent for formulating queries, including the total time of query formulation (`total_time`) and the average time taken to accept or reject suggestions (`accept_time` and `reject_time`, respectively). Finally, the last set of measurements evaluates the properties of the resulting queries in terms of the number of characters of the submitted queries (`query_chars`) and their number of terms (`query_terms`).

6.2.2 Example

Before providing a detailed discussion of the results of the user study, we show in Table 6 a selected example that illustrates the differences of the standard and term-by-term QAC methods for assisting the same user in formulating the search task number 1 in Table 5. As we can see, in this particular example the user ends up formulating the same query when using either mechanisms. However, in the process of formulating the final query we see significant differences between one method and the other.

On the one hand, in the formulation process assisted with the standard QAC method, we see that the user rejects the two first suggestions shown and only accepts the third one which completes the last two terms of the query that is submitted. We also observe how the user takes considerable time exploring the list of suggestions and is forced to go to the third position of the accepted suggestion to find the desired completion. In this case, the system is able to save the user entering two out for four terms of the query (not including the first term).

One the other hand, the formulation with help of the term-by-term QAC mechanism shows the advantages of our method for this particular user and search task. As we can see, the QAC system is activated a total of five times (one more than in the standard method), for which the completions shown are accepted in three cases. The position of the accepted completions is, moreover, very low (first and second positions only), which accounts for a faster decision time. In the end, the user is able to save typing three terms of the intended query.

property	STD	TBT	
`suggestions`	2.9514	3.1486	**6.68%**
`accepted`	0.3661	0.4029	**10.06%**
`accept_pos`	2.0908	1.9601	-6.25%
`chars_saved`	0.2386	0.2551	6.91%
`terms_saved`	0.2753	0.2838	3.08%
`effort`	3.9221	3.7924	**-3.31%**
`total_time`	12.6345	12.6133	-0.17%
`accept_time`	3.3058	2.8943	**-12.45%**
`reject_time`	4.8333	4.4531	**-7.87%**
`query_chars`	22.9686	22.6200	-1.52%
`query_terms`	3.4600	3.3914	-1.98%

Table 7: Results of the user study. Statistically significant relative improvements of the term-by-term method (Wilcoxon signed-rank test with $p < 0.05$) are marked in bold.

6.2.3 Results

The global results of the user study can be found in Table 7. As it can be seen, a number of statistically significant differences are observed between methods (Wilcoxon signed-rank test with $p < 0.05$). In the first block of results, which deal with the success of the QAC method under testing, we see that the term-by-term method is activated a 6.68% more times than the standard one. This is expected as the standard mechanism, by providing longer completions, is expected to be used less for an average task. Regarding the success of the suggestions, we see that completions provided by our term-by-term method have a higher ratio of acceptance. Also, the average position of the accepted completions is lower (i.e. appears higher in the list of suggestions) than in the standard method, although the difference is not found statistically significant.

The second block of results relates to the direct measurement of the metrics defined by the user model in Section 5, together with the number of characters saved. In general these results provide a weak confirmation of our claims and observations from the query log-based evaluation, showing that more terms are saved with a smaller effort, but with a much smaller relative difference than in the previous evaluation and with significant differences found only in the effort metric.

A third category of results measure the amount of time involved in exploring suggestions and the total time involved in query formulation. The outcomes indicate that users tend to spend a similar amount of time to formulate the queries in both methods, although the time required to accept or reject suggestions is smaller in the term-by-term approach. The latter observations can be explained by the fact that the suggestions are typically much shorter than in the standard case, which facilitates a faster exploration of the suggestions.

Finally, a last set of observations comprises the properties of the submitted queries, in terms of the number of characters and terms. As the results show, queries formulated with the assistance of the standard QAC method seem to be slightly longer (both in characters and terms) than the queries with help of the term-by-term method, although the difference

does not seem to be statistically significant. By taking a closer look at the submitted queries between methods, we see that in 37% of the cases the pairs of queries for a same user in a search tasks were identical for both methods and, for the pairs of different queries, the normalized Levenshtein distance was in average 0.38, which indicates that the queries were indeed different but shared similar terms.

7. CONCLUSIONS AND FUTURE WORK

In this paper we tackled the problem of using query auto-completion mechanisms in mobile search. We identified a series of inconveniences of standard whole-query completions caused by the limitations of text input and display in mobile devices such as smartphones and tables. In order to alleviate these issues, we proposed a term-by-term QAC mechanism that suggests, rather than whole-query completions, one term at a time. We described an implementation of this term-by-term QAC system that uses a structure called query-term graph, which allows an efficient storage and retrieval of query completions. We also proposed a user model for the interaction with QAC systems based on prior work from which we derive metrics to asses the efficiency of QAC in query log-based evaluations. An experiment with a query log from a commercial search engine demonstrated that our QAC mechanism significantly outperforms well-established state-of-the-art standard auto-completion mechanisms. Furthermore, a small user study provided insights about and confirmed the benefits of our system throughout a wide number of evaluation metrics.

We envision two main directions for future work. First, we want to enhance our term-by-term QAC method by incorporating additional sources of information. On one hand, we believe that most of the ideas of more recent proposals for standard QAC – such as those described in Section 2 – can be easily adapted to our approach. On the other hand, we will explore enhancements specific to our query-term graph. In particular, the detection and merging of sub-paths of the graph [1] (as shown in Figure 2) could be a way not only to reduce the size of the structure, but also to discover unseen query formulations that ultimately improve the prediction capabilities of our system. Also, we will extend our method to provide suggestions that may also include short phrases.

As a second research direction, we want to study the online user behavior of our auto-completion paradigm in the context of a commercial mobile search engine, observing real traffic and a large number of user interactions, to shed light on the particularities of different auto-completion methods in a live setting. Furthermore, we want to study the characteristics of queries formulated with the assistance of both standard and term-by-term QAC methods in terms of the quality of the retrieved search results.

8. REFERENCES

[1] P. Anchuri, M. J. Zaki, O. Barkol, S. Golan, and M. Shamy. Approximate graph mining with label costs. In *19th ACM Conference on Knowledge Discovery and Data Mining*, KDD '13, pages 518–526, 2013.

[2] R. Baeza-Yates, G. Dupret, and J. Velasco. A study of mobile search queries in Japan. In *16th International World Wide Web Conference*, WWW '07, 2007.

[3] Z. Bar-Yossef and N. Kraus. Context-sensitive query auto-completion. In *20th International World Wide Web Conference*, WWW '11, pages 107–116, 2011.

[4] S. Bhatia, D. Majumdar Debapriyo and P. Mitra. Query suggestions in the absence of query logs. In *34th ACM Conference on Research and Development in Information Retrieval*, SIGIR '11, pages 795–804, 2011.

[5] P. Boldi and S. Vigna. The Webgraph framework I: Compression techniques. In *13th International World Wide Web Conference*, WWW '04, pages 595–602, 2004.

[6] O. Chapelle, D. Metlzer, Y. Zhang, and P. Grinspan. Expected reciprocal rank for graded relevance. In *18th ACM Conference on Information and Knowledge Management*, CIKM '09, pages 621–630, 2009.

[7] K. Church, B. Smyth, K. Bradley, and P. Cotter. A large scale study of european mobile search behaviour. In *10th Conference on Human Computer Interaction with Mobile Devices and Services*, MobileHCI '08, pages 13–22, 2008.

[8] K. Church, B. Smyth, P. Cotter, and K. Bradley. Mobile information access: A study of emerging search behavior on the mobile internet. *ACM Transactions on the Web*, 1(1), May 2007.

[9] J. Darragh, I. H. Witten, and M. James. The reactive keyboard: a predictive typing aid. *Computer*, 23(11):41–49, Nov 1990.

[10] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, Jan. 2008.

[11] E. Fredkin. Trie memory. *Communications of the ACM*, 3(9):490–499, Sept. 1960.

[12] N. Garay-Vitoria and J. González-Abascal. Intelligent word-prediction to enhance text input rate. In *2nd Conference on Intelligent User Interfaces*, IUI '97, pages 241–244, 1997.

[13] J.-L. Gomez-Barroso, R. Compano, C. Feijoo, M. Bacigalupo, O. Westlund, S. Ramos, A. Jaokar, F. Alvarez, R. De Waele, G. Mateos-Barrado, and C. Garcia-Jimenez. *Prospects of Mobile Search*. Publications Office of the European Union, 2010.

[14] K. Hofmann, B. Mitra, F. Radlinski, M. Shokouhi. An eye-tracking study of user interactions with query auto completion. In *23rd ACM Conference on Information and Knowledge Management*, CIKM '14, pages 549–558, 2014.

[15] J.-Y. Jiang, Y.-Y. Ke, P.-Y. Chien, and P.-J. Cheng. Learning user reformulation behavior for query auto-completion. In *37th ACM Conference on Research and Development in Information Retrieval*, SIGIR '14, pages 445–454, 2014.

[16] M. Kamvar and S. Baluja. A large scale study of wireless search behavior: Google mobile search. In *ACM Conference on Human Factors in Computing Systems*, CHI '06, pages 701–709, 2006.

[17] M. Kamvar and S. Baluja. Deciphering trends in mobile search. *Computer*, 40(8):58–62, Aug. 2007.

[18] M. Kamvar and S. Baluja. Query suggestions for mobile search: understanding usage patterns. In *ACM Conference on Human Factors in Computing Systems*, CHI '08, pages 1013–1016, 2008.

[19] E. Kharitonov, C. Macdonald, P. Serdyukov, and I. Ounis. User model-based metrics for offline query suggestion evaluation. In *36th ACM Conference on Research and Development in Information Retrieval*, SIGIR '13, pages 633–642, 2013.

[20] Y. Li, A. Dong, H. Wang, H. Deng, Y. Chang, and C. Zhai. A two-dimensional click model for query auto-completion. In *37th ACM Conference on Research and Development in Information Retrieval*, SIGIR '14, pages 455–464, 2014.

[21] B. Mitra, M. Shokouhi, F. Radlinski, and K. Hofmann. On user interactions with query auto-completion. In *37th ACM Conference on Research and Development in Information Retrieval*, SIGIR '14, pages 1055–1058, 2014.

[22] B. Mitra, and N. Craswell. Query auto-Completion for rare prefixes. In *24th ACM Conference on Information and Knowledge Management* , CIKM '15, pages 1755–1758, 2015.

[23] J. Pasquero, J. Griffin, and D. Mckenzie. Method and apparatus for word prediction selection, Sept. 17 2014. EP Patent App. EP20,130,159,459.

[24] M. Shokouhi. Learning to personalize query auto-completion. In *36th ACM Conference on Research and Development in Information Retrieval*, SIGIR '13, pages 103–112, 2013.

[25] M. Shokouhi and K. Radinsky. Time-sensitive query auto-completion. In *35th ACM Conference on Research and Development in Information Retrieval*, SIGIR '12, pages 601–610, 2012.

[26] Y. Song, H. Ma, H. Wang, and K. Wang. Exploring and exploiting user search behavior on mobile and tablet devices to improve search relevance. In *22nd International World Wide Web Conference*, WWW '13, pages 1201–1212, 2013.

[27] S. Whiting and J. M. Jose. Recent and robust query auto-completion. In *23rd International World Wide Web Conference*, WWW '14, pages 971–982, 2014.