

Comparing Approaches for Query Autocompletion

Giovanni Di Santo
giovi.disanto@gmail.com
Università degli studi dell'Aquila

Richard McCreadie, Craig Macdonald,
and Iadh Ounis
{firstname.lastname}@glasgow.ac.uk
School of Computing Science
University of Glasgow
G12 8QQ, Glasgow, UK

ABSTRACT

Within a search engine, query auto-completion aims to predict the final query the user wants to enter as they type, with the aim of reducing query entry time and potentially preparing the search results in advance of query submission. There are a large number of approaches to automatically rank candidate queries for the purposes of auto-completion. However, no study exists that compares these approaches on a single dataset. Hence, in this paper, we present a comparison study between current approaches to rank candidate query completions for the user query as it is typed. Using a query-log and document corpus from a commercial medical search engine, we study the performance of 11 candidate query ranking approaches from the literature and analyze where they are effective. We show that the most effective approaches to query auto-completion are largely dependent on the number of characters that the user has typed so far, with the most effective approach differing for short and long prefixes. Moreover, we show that if personalized information is available about the searcher, this additional information can be used to more effectively rank query candidate completions, regardless of the prefix length.

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

Keywords: Query Completion, Information Retrieval

1. INTRODUCTION

Query auto-completion is a relatively recent functionality offered by search engines, which, given a prefix already typed by a user, tries to guess the final query the user is going to type. Using this functionality, the user does not need to type the whole query and in some cases the auto-completion suggestions may help the user formulate a more precise query. From an information retrieval perspective, this can be seen as a ranking task, where the aim is to rank all possible candidate (complete) queries given the current prefix typed by the user. The top ranked candidate queries will then be displayed as the user types. Figure 1 illustrates

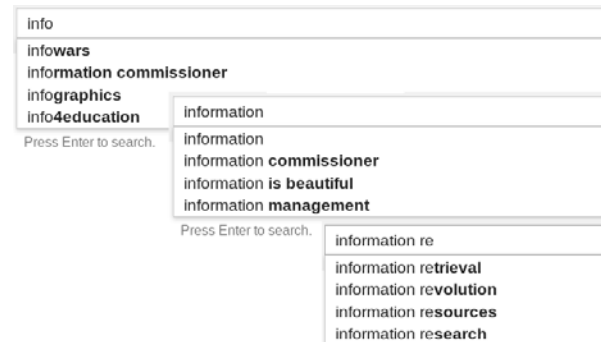


Figure 1: Illustration of query auto-completion for the query 'information retrieval'.

query auto-completion as implemented in a current search engine for the query 'information retrieval'.

Importantly, query auto-completion is a challenging task. In particular, given only a few characters that can be quite ambiguous, there are potentially thousands of candidate queries that the user might type. To tackle this challenge, a variety of approaches have been proposed [1, 7, 8, 11]. Early works examined probability-based ranking, where queries are ranked by their occurrence probability within a background corpus [2]. Meanwhile, other studies have focused on leveraging evidence from query logs to rank the candidate queries. For instance, Bar-Yossef and Kraus [1] ranked the candidate queries by textual similarity to a user profile, while Shokouhi [8] leveraged user history and demographic features within a learning-to-rank model to rank the candidate queries. However, of important note is that these prior studies typically report performance using different proprietary datasets (e.g. for privacy reasons) and often do not share any common baselines. Hence, it is difficult to compare the wealth of approaches to query auto-completion to see which is the most effective, and, as a result, it is unclear what the state-of-the-art in query autocompletion is.

As a step toward tackling this knowledge gap, in this paper, we perform a comparison of 11 different approaches to rank query auto-completions, with the aim of determining those that are the most effective. In particular, using the same dataset comprised of medical articles and associated query/click logs from a commercial medical search engine, we perform a comparison of query auto-completion approaches taken from and inspired by the literature, covering both query-log independent approaches and personalized approaches that are primarily driven by query logs. From analysis of the results, we show that in general, approaches

that personalize to the specific user outperform approaches that either do not use query-log evidence at all, or only consider past queries across multiple users in aggregate. Furthermore, we show that there is a dependence between the number of characters that the user has typed and the performance of the approaches tested - with the most effective approach changing based on the prefix length.

2. TASK DEFINITION

A query auto-completion engine takes as input a prefix p of length l typed by the user in the search engine box and a set of *candidate queries* Q_p to rank. A prefix p is a sequence of characters belonging to the query the user is about to submit. The candidate queries Q_p are a set of possible completions for the current prefix. These should be of high quality and represent the space of possible information needs that the user could have. The aim is to rank the candidate queries Q_p for a given prefix p , such that the query the user was typing appears within the top ranks.

Importantly, for the purposes of this paper, we assume that the set of candidate queries for each prefix are pre-provided. In this way, we can compare different query ranking strategies over the same set of candidate queries. We leave an examination of different approaches to generate these candidate queries to future work.

3. APPROACHES TO RANK CANDIDATE QUERIES

In this section, we summarize and formally define the 11 different candidate query ranking approaches that we compare in our subsequent experiments. In particular, we formalize each approach as a scoring function $score(q)$, where $q \in Q_p$. Note that we assume that we have access to a query-log containing past queries Q' and document clicks C' from which we can extract training evidence. We also naturally assume that we have access to the collection of documents D that the user is searching. Finally, for personalized approaches, we refer to all queries previously entered by the current user as the *user context*.

Most popular ranker (MP) is the ranking algorithm used as a *baseline* in the majority of the works present in the literature. Query candidates are ranked according to their past popularity, calculated as the query’s frequency (cf) inside the query log. Usually, this frequency is normalized by the total sum of query frequencies. Notably, there are different variants of this approach. Shokouhi and Radinsky [9] replaced the query’s actual frequency with a predicted frequency, while Strizhevskaya *et al.* [10] model the query frequencies using a time-series. We use the actual query frequency within the query-log.

$$MP(q) = \frac{cf(q)}{\sum_{i \in Q'} cf(i)} \quad (1)$$

Sentence occurrence ranker (SO) scores each query based upon the number of documents that match one or more of the query terms plus the number of documents that match all of the query terms. SO is inspired by [2].

$$SO(q) = \alpha \cdot |exactMatch(q, D)| + |termMatch(q, D)| \quad \alpha > 1 \quad (2)$$

Time ranker (TR) is a simple ranking algorithm that scores each candidate completion q by the time difference

between the current time and the most recent past occurrence of it in the query log Q' , thereby promoting queries that have been issued recently. LRD is the largest time difference between q and any query in Q' .

$$TR(q) = LRD - \operatorname{argmin}_{q_i \in match(q, Q')} (date(q) - date(q_i)) \quad (3)$$

Most popular time ranker (MT) mixes *most popular ranker* and *time ranker* together, inspired by the *hybrid approach* in [1]. In our work, we set α to 0.7.

$$MT(q) = \alpha \cdot MP(q) + (1 - \alpha) \cdot TR(q) \quad (4)$$

Term occurrence ranker (TO) ranks candidate queries based on the background popularity of those candidate’s terms. Term popularity is calculated as the mean of the frequency of the term inside the query log and the *TF-IDF* of the term within the corpus of documents being searched. The score for a candidate query is the mean of the scores for its terms.

$$TO(q) = \frac{\sum_{t \in q} \frac{(tfidf_{q_l}(q) \cdot tfidf_c(q))}{2}}{|q|} \quad (5)$$

Near words (NW) takes into consideration the distance among the terms typed by the user in the candidate completions. It promotes completions where the query terms are next to one another. The *frequency difference* (fd) among the completions having the highest and lowest frequency in Q_p is calculated and divided by the number of terms present in the longest query, obtaining the *term difference* (td). For each pair of user terms a value is assigned and summed to the overall score. That value is equal to the fd if one term directly follows the other and becomes lower as the distance increases. The td is subtracted from the fd for each term between the user terms pair. T_u in Equation (6) is defined as the set of terms typed by the user.

$$NW(q) = \frac{\sum_{(t_1, t_2) \in T_u} fd - td \cdot distance(t_1, t_2)}{|T_u|} \quad (6)$$

String similarity ranker (SS) scores a query based upon the similarity between that query and all previous queries issued by the user who submitted that query. The similarity measure used in this paper is the *Jaro Winkler edit distance* [13].

$$SS(q) = \frac{\sum_{q_i \in (Q')} JW(q, q_i)}{|Q'|} \quad (7)$$

WordNet similarity ranker (WR) uses *WordNet* [3] in order to capture semantic similarities between the query being ranked and the previous queries entered by the same user. The similarity is calculated for every pair of terms present in the completion being ranked and the user context. Mean similarity over the user’s previous queries is used for scoring the completion. We use the Wu & Palmer [12] measure that calculates the relatedness of two terms by considering the depths of the two related synsets in the WordNet taxonomies.

$$WR(q) = \frac{\sum_{q_i \in (Q')} \sum_{(q_t, q_{it}) \in (q, q_i)} JW(q, q_i)}{|Q'|} \quad (8)$$

N-Gram similarity ranker (NR) scores a candidate query based upon the syntactic relation between it and the user

context. In particular, we use the N-Gram similarity algorithm presented in [4]. We chose to use the *positional n-gram similarity* as the n-gram matching strategy.

$$NR(q) = \frac{\sum_{q_i \in (Q')} N - \text{gramSim}(q, q_i)}{|Q'|} \quad (9)$$

Kernel similarity ranker (KR) is based on the work presented in [5] and [6] that aims to estimate the similarity between two small strings by taking into account their semantic similarities. In particular, collection enrichment using the document corpus is applied to each candidate query and the user context. A candidate query is scored based upon the similarity between its expanded form and the expanded form of the user context (using the same kernel similarity measure as in [6]).

$$KK(q) = \frac{\sum_{q_u \in (C)} \text{KernSim}(q, q_u)}{|C|} \quad (10)$$

Clicked documents ranker (CR) models the user’s interests as the content of documents previously clicked (D_c), where this set is represented by the terms present inside the document titles. The candidate query is represented in the same way, but considering all of the documents previously clicked for that candidate query in the query log (D_q). The *cosine similarity* measure is used to score the candidate query representation by its similarity to the representation of the user’s interests.

$$CR(q) = \text{cosine}(q_d, c_d) \quad \begin{aligned} q_d &= \{t | t \in d_{\text{title}}, d \in D_q\} \\ c_d &= \{t | t \in d_{\text{title}}, d \in D_c\} \end{aligned} \quad (11)$$

The first three columns of Table 1 list the 11 approaches discussed above, highlighting whether each uses query-log information and/or the user context (e.g. queries previously entered by the user for personalization).

4. EXPERIMENTAL SETUP

Dataset: To evaluate query auto-completion, we use a sample of 1,417,880 unique queries issued by 37,806 different users between November 2010 to April 2013, provided by the *TRIPDatabase.com* medical search engine.¹ This dataset additionally contains 1,418,996 medical articles. Note that TRIP has basic in-built query suggestion based only on the corpus statistics, similar to SO approach.

Training/Testing Split: To facilitate experimentation, we divide the dataset into training and testing splits. In particular, all queries issued before April 2013 are used for training and extracting the user context information (past queries and clicks). For testing, we use a sub-set of the query sample, composed of 1405 queries by 968 users who had used TRIP for at least 3 months since March 2013 and had submitted at least one query during the month of April 2013.

Candidate Query Generation: To produce the candidate query set for a prefix, we use the query log from the TRIP medical search engine. Given a prefix, we first use a weighted compressed ternary search tree (*WCT*) that returns the first k most likely completions for the last term in the current prefix.² For instance, the prefix ‘diab’ might return ‘diabetes’ and ‘diabetic’ as term completions. We then

¹<http://tripdatabase.com/>

²Often the prefix will be only a single incomplete term, e.g. ‘weig’ but for longer prefixes it may be a second term that is incomplete, such as ‘weight lo’.

use these terms to retrieve all queries, from the aforementioned search engine query-log, matching them and all the other terms already fully typed by the user.

Evaluation Metrics: As a ranking task, we evaluate query auto-completion performance in terms of the mean reciprocal rank (MRR) of the candidate query ranking produced by each approach over all of the test queries. To do so, we crop each query to the first k characters ($k \in \{2, 4, 6, 8$ and $10\}$), representing the user at different stages of typing that query. Only the final query that the user issued is considered the correct completion. We report MRR for all queries when concatenated to each of the target lengths.

5. QUERY AUTO-COMPLETION PERFORMANCES

In this section, we aim to answer the research question, ‘RQ: which query auto-completion ranking approaches are the most effective?’. To do so, we compare the performance of each query auto-completion ranking approach described in Section 3. In particular, Table 1 reports the MRR performance of each of these approaches when ranking candidate query (completions) for prefixes of lengths $\{2, 4, 6, 8, 10\}$.

From Table 1, we observe the following. First, as a sanity check, comparing the ‘sentence occurrence ranker’ (SO) that does not use any query-log evidence to the other approaches that leverage query logs, we see that the query-log-based approaches all outperform the ‘sentence occurrence ranker’ (SO) for all prefix lengths. This confirms our expectations that approaches that rely on corpus statistics [2] are outperformed by query-log-based approaches [8].

Second, comparing the approaches that use query-log evidence without considering the current user in particular, we see that the most effective of these approaches when the user first starts typing (prefix length of 2), is the ‘most popular ranker’ (MP) approach. Recall that this approach simply chooses the most frequently occurring candidate query that starts with the target prefix. However, we also see that as the user continues to type (prefix lengths 4-10) the approach that combines query popularity with recency, i.e. the ‘most popular time ranker’ (MT) becomes incrementally more effective. To explain this behaviour, it is first important to note that for all approaches, as the prefix length increases, the ranking performance of all approaches also increases. This is to be expected, since as the prefix becomes longer, there are less possible candidate queries that match the prefix. For a very short prefix, considering the candidate recency can be misleading, since there may be hundreds of candidate queries that were issued close to the current search time, but that are not likely to be good query candidates. Hence, the recency factor dilutes the more effective query frequency signal. However, as the candidate query set becomes smaller (due to a longer prefix length), the recency signal becomes more useful, as it down-weights candidate queries that have not been issued for months or years, and hence are not likely candidates.

Third, comparing the approaches that use the query-log evidence without considering the current user (MP, TR, MT, TO, and NW) to those approaches that additionally employ the query log to personalize to the current user (WR, NR, KR, CR and SS), we see that, in general, personalized approaches outperform unpersonalized approaches. For example, at prefix length 2, the approach that personalizes based on clicked documents (‘clicked document ranker’ (CR)) out-

Approaches			MRR				
Ranking	Query-log Evidence	Personalized	2	4	6	8	10
Sentence occurrence ranker (SO)	No	No	0.005▼	0.0456▼	0.0696▼	0.1003▼	0.1546▼
Most Popular ranker (MP)	Yes	No	0.0964	0.2146	0.2851	0.3248	0.3641
Time Ranker (TR)	Yes	No	0.0324▼	0.1236▼	0.1995▼	0.2707▼	0.3281▼
Most Popular Time ranker (MT)	Yes	No	0.0961	0.2249▲	0.3112▲	0.3684▲	0.4153▲
Terms occurrence ranker (TO)	Yes	No	0.0021▼	0.0326▼	0.0773▼	0.1163▼	0.1617▼
Near Words Ranker (NW)	Yes	No	0.0611▼	0.1576▼	0.2347▼	0.2972▼	0.3611
String Similarity Ranker (SS)	No	Yes	0.0137▼	0.0711▼	0.1628▼	0.1149▼	0.2069▼
WordNet Similarity Ranker (WR)	Yes	Yes	0.089▼	0.0302▼	0.0711▼	0.0908▼	0.1055▼
N-Gram Similarity Ranker (NR)	Yes	Yes	0.0837	0.2927▲	0.3693▲	0.4207▲	0.4602▲
Kernel Similarity Ranker (KR)	Yes	Yes	0.907	0.2876▲	0.3356▲	0.3923▲	0.4121▲
Clicked Documents Ranker (CR)	Yes	Yes	0.1442▲	0.2952▲	0.3462▲	0.3938▲	0.4183▲

Table 1: Query auto-completion performance over the queries issued during the month of April '13 in our dataset, using the 11 presented ranking approaches. Statistically significant improvements/reductions in performance over the Most Popular ranker (MP) ($p < 0.05$ paired t-test) are denoted ▲ and ▼, respectively.

performs the query frequency-based ranker discussed earlier ('most popular ranker' (MP)) by a large and statistically significant margin (0.0964 MAP to 0.1442 MAP). This indicates that personalization to each individual user is preferable.³ Next, comparing approaches within the class of personalized approaches, we see that for small prefix lengths (2-4) ranking the candidate queries via similarity to previously clicked documents by the user ('clicked documents ranker' (CR)) is the most effective. However, as the prefix length increases (6-10), the n-gram similarity approach that compares against the past queries issued by the user ('n-gram similarity ranker' (NR)) becomes more effective. These results indicate that for short prefix lengths, where a user has likely issued multiple queries that are valid candidates previously, using past queries is less effective as it is difficult to break ties between those candidates. On the other hand, as the prefix length increases, there is often only one previous candidate query that the user has entered that matches, which is likely to be the correct candidate (e.g. if the user is re-finding).

To answer our research question RQ, on our dataset there is no single approach that illustrates a top performance for all query lengths. Rather, the correct candidate query ranking approach is dependent on how many characters the user has typed so far. For short prefix lengths, simple past query frequency (MP ranker) is a good feature if no personalization information is available, while ranking based on the content of previously clicked documents is effective otherwise (CR ranker). For longer prefix lengths, mixing query candidate recency with its past usage frequency can result in performance gains (MT ranker) in the unpersonalized setting. Meanwhile, if personalized information is available to the search system, then ranking query suggestions by n-gram similarity to the users past queries is more effective (NR ranker).

6. CONCLUSIONS

In this paper, we presented a comparison study of current approaches to rank candidate query completions for the user query as they type. Using a query-log and document corpus from the TRIP medical search engine, we evaluated the performance of 11 candidate query ranking approaches from the

³Note that this information may not be available if users are searching anonymously however.

literature and analysed where they are effective. From this analysis, we conclude that the most effective approach to query auto-completion is largely dependent on the number of characters that the user has typed so far and that personalized information can be used to more effectively rank the query candidate completions. This indicates selective approaches that apply different query-autocompletion models depending on the prefix length and user context are a promising direction for future work.

7. ACKNOWLEDGEMENT

We thank Jon Brassey from TRIPDatabase.com for providing the data used in this research.

8. REFERENCES

- [1] Z. Bar-Yossef and N. Kraus. Context-sensitive query auto-completion. In *Proc. of WWW*, 2011.
- [2] S. Bhatia, D. Majumdar, and P. Mitra. Query suggestions in the absence of query logs. In *Proc. of SIGIR*, 2011.
- [3] C. Fellbaum. Wordnet and wordnets. *Encyclopedia of Language and Linguistics*, 2005.
- [4] G. Kondrak. N-gram similarity and distance. In *Proc. of SPIRE*, 2005.
- [5] M. Sahami and T. D. Heilman. Mining the web to determine similarity between words, objects, and communities. In *Proc. of FLAIRS*, 2006.
- [6] M. Sahami and T. D. Heilman. A web-based kernel function for measuring the similarity of short text snippets. In *Proc. of WWW*, 2006.
- [7] C. Sengstock and M. Gertz. Conquer: A system for efficient context-aware query suggestions. In *Proc. of WWW*, 2011.
- [8] M. Shokouhi. Learning to personalize query auto-completion. In *Proc. of SIGIR*, 2013.
- [9] M. Shokouhi and K. Radinsky. Time-sensitive query auto-completion. In *Proc. of SIGIR*, 2012.
- [10] A. Strizhevskaya, A. Baytin, I. Galinskaya, and P. Serdyukov. Actualization of query suggestions using query logs. In *Proc. of WWW*, 2012.
- [11] S. Whiting and J. M. Jose. Recent and robust query auto-completion. In *Proc. of WWW*, 2014.
- [12] Z. Wu and M. Palmer. Verbs semantics and lexical selection. In *Proc. of ACL*, 1994.
- [13] Winkler, William E String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage, 1990.