Information Processing and Management 000 (2016) 1–16

Contents lists available at ScienceDirect

Information Processing and Management

journal homepage: www.elsevier.com/locate/ipm

Learning from homologous queries and semantically related terms for query auto completion

Fei Cai[a,b,*], Maarten de Rijke[b]

[a] Science and Technology on Information Systems Engineering Laboratory, National University of Defense Technology, Hunan, China
[b] Informatics Institute, University of Amsterdam, Amsterdam, The Netherlands

A R T I C L E I N F O

Article history:
Received 27 April 2015
Revised 26 November 2015
Accepted 1 December 2015
Available online xxx

Keywords:
Query auto completion
Semantics
Query suggestion
Learning to rank

A B S T R A C T

Query auto completion (QAC) models recommend possible queries to web search users when they start typing a query prefix. Most of today's QAC models rank candidate queries by popularity (i.e., frequency), and in doing so they tend to follow a strict query matching policy when counting the queries. That is, they ignore the contributions from so-called homologous queries, queries with the same terms but ordered differently or queries that expand the original query. Importantly, homologous queries often express a remarkably similar search intent. Moreover, today's QAC approaches often ignore semantically related terms. We argue that users are prone to combine semantically related terms when generating queries.

We propose a learning to rank-based QAC approach, where, for the first time, features derived from homologous queries and semantically related terms are introduced. In particular, we consider: (i) the observed and predicted popularity of homologous queries for a query candidate; and (ii) the semantic relatedness of pairs of terms inside a query and pairs of queries inside a session. We quantify the improvement of the proposed new features using two large-scale real-world query logs and show that the mean reciprocal rank and the success rate can be improved by up to 9% over state-of-the-art QAC models.

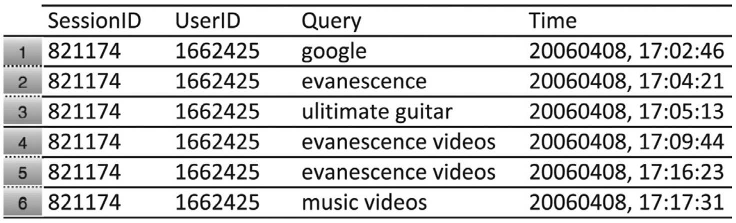
1. Introduction

Query auto completion (QAC), a popular feature of modern search engines, is offered to help users formulate a query when they have an intent in mind but not a clear way to express it. The typical query completion service of a modern search engine takes a few initial characters entered by the user and returns matching queries to automatically complete the search clue. Where offered, query completion is heavily used by visitors and highly influential on search results (Mitra, Shokouhi, Radlinski, & Hofmann, 2014).

Unlike query recommendation or query suggestion, auto-completed queries strictly start with an initially typed prefix (Cai, Liang, & de Rijke, 2014b). Most previous work on QAC is centered around the Most Popular Completion (MPC) approach, which ranks QAC candidates by query popularity, i.e., frequency, collected either from historical logs (Bar-Yossef & Kraus, 2011; Whiting & Jose, 2014) or from future predictions (Shokouhi & Radinsky, 2012; Whiting & Jose, 2014). In the latter

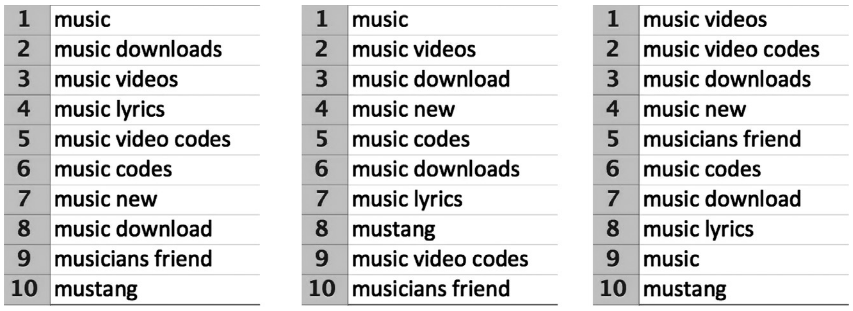
[*] Corresponding author at: Science and Technology on Information Systems Engineering Laboratory, National University of Defense Technology, Hunan, China. Tel.: +31 687591248.
E-mail address: f.cai@uva.nl, caifei1104@gmail.com (F. Cai).

http://dx.doi.org/10.1016/j.ipm.2015.12.008
0306-4573/© 2015 Elsevier Ltd. All rights reserved.



	SessionID	UserID	Query	Time
1	821174	1662425	google	20060408, 17:02:46
2	821174	1662425	evanescence	20060408, 17:04:21
3	821174	1662425	ulitimate guitar	20060408, 17:05:13
4	821174	1662425	evanescence videos	20060408, 17:09:44
5	821174	1662425	evanescence videos	20060408, 17:16:23
6	821174	1662425	music videos	20060408, 17:17:31

Fig. 1. An AOL session example.



1	music	1	music	1	music videos
2	music downloads	2	music videos	2	music video codes
3	music videos	3	music download	3	music downloads
4	music lyrics	4	music new	4	music new
5	music video codes	5	music codes	5	musicians friend
6	music codes	6	music downloads	6	music codes
7	music new	7	music lyrics	7	music download
8	music download	8	mustang	8	music lyrics
9	musicians friend	9	music video codes	9	music
10	mustang	10	musicians friend	10	mustang

(a) Ranked by frequency of candidates. **(b) Reranked by frequency of homologous queries.** **(c) Reranked by semantic similarity.**

Fig. 2. Ranked lists of QAC candidates for the prefix “mus”.

case, methods from time series analysis are put to work to predict the query frequency (Cai et al., 2014b; Shokouhi & Radinsky, 2012; Whiting & Jose, 2014).

We propose to complement these time- and popularity-based QAC models with two methods based on lexical variations. First of all, popularity-based QAC models invariably count the query volumes following a strict query matching policy, thereby ignoring the contributions from so-called *homologous queries*, i.e., (1) queries with the same terms as the candidate query but in a different order and (2) queries that extend the candidate query. Formally, we define the following two types of homologous queries for a given query $q = (term_1, term_2, \ldots, term_m)$: (1) Given q, a *super query* of q is a query $s_q = (term_1, term_2, \ldots, term_m, term_{m+1}, \ldots, term_L)$ that extends q; (2) A *pseudo-identical query* for q is a query p_q that is a permutation of q. To a certain extent, homologous queries express similar search intents. For instance, at the time of writing (late 2014), for the two queries “*Chile SIGIR*” and “*SIGIR Chile*” (a pseudo-identical query of “*Chile SIGIR*”), the same SERPs should probably be returned. And the SERPs for “*Chile SIGIR*” and “*Chile SIGIR 2015*” (a super query of “*Chile SIGIR*”) should probably overlap to a very large degree. Based on these examples, we hypothesize that it is advantageous to consider homologous queries as a context resource for QAC.

QAC features inferred from homologous queries are one important innovation that we study in this paper. A second way of using lexical variations for QAC that we propose is based on semantically related terms. As discussed in the literature, a user’s search history usually reveals their search intent, often expressed by the queries or clicked documents. For instance, Shokouhi (2013) studies the similarity between a QAC candidate and previous queries in both the short-term and long-term history for reranking QAC candidates. And Jiang, Ke, Chien, and Cheng (2014) infer features from users’ reformulation behavior for reranking QAC candidates. We exploit a similar intuition but operationalize it differently, by considering the semantic relatedness of terms in a QAC candidate and of terms from a QAC candidate and queries previously submitted in the same session. Let us give an example. Consider Fig. 1, which contains a session from the well-known AOL query log.

For the sake of the example, let us assume that we have not yet seen the last query (query 6, “music videos”) and that it is in fact the initial segment “mus” of this query for which we want to recommend completions. A regular baseline based on query frequency is likely to rank the completion “music” first, as shown in Fig. 2a. If we consider the observed frequency of homologous queries for a candidate, we would return the list seen in Fig. 2b, which is a reranked version of the list in Fig. 2a. Clearly, the queries “music” and “music video” gain more benefits from homologous queries than others as they are now ranked at the top. But if we look in the user’s search session (e.g., at query 4 and 5 in Fig. 1), we would see that “videos” is semantically closely related to earlier queries. Based on this insight, the query “music videos” in Fig. 2a is a more sensible completion. Considering the semantic similarity of terms both inside a candidate and of queries inside a session can generate another reranked QAC list shown in Fig. 2c. We can see semantically close queries, e.g., “music videos” and “music video codes,” have now been to the top of the list.

Motivated by the examples above, we study the potential of homologous queries and semantic relatedness for improving state-of-the-art QAC methods. In particular, in addition to effective popularity-based features of QAC candidates, extended with time-based features and features of user reformulation behavior, we consider time- and popularity-based features for homologous queries as well as semantic features based on semantic relatedness of terms in a candidate and of pairs of terms from a candidate and from previous queries in the same session. Building on LambdaMART (Burges, Svore, Bennett, Pastusiak, & Wu, 2011), we propose a learning to rank (L2R) based QAC model, called L2R-QAC, along with a number of variations depending on different groups of features used to rerank the top N QAC candidates returned by popularity. We evaluate the effectiveness of these models using two large publicly available query logs.

Our contributions in this paper can be summarized as

1. We propose a learning to rank based query auto completion model (L2R-QAC) that exploits contributions from so-called homologous queries for a QAC candidate, in which two kinds of homologous queries are taken into account.
2. We propose semantic features for QAC, using the semantic relatedness of terms inside a query candidate and of pairs of terms from a candidate and from queries previously submitted in the same session.
3. We analyze the effectiveness of our L2R-QAC model with newly added features, and find that it significantly outperforms state-of-the-art QAC models, either based on learning to rank or on popularity.

We describe related work in Section 2. Features for the specific QAC learning problem are described in Section 3. Then, Section 4 presents our experimental setup. In Section 5 we report our experimental results. We conclude in Section 6, where we suggest future research directions.

2. Related work

Since the first query auto completion (QAC) system in web search was launched by Google,[1] also known as "Google Autocomplete" in 2004, this particular feature is known as "type-ahead" or "auto-complete." QAC has subsequently been studied by Bar-Yossef and Kraus (2011) with the well-known *Most Popular Completion* (MPC) approach. Completions can be sorted by popularity or any other metric identified by the customer, so that the most useful or advantageous query completions are listed first. In recent years, QAC has been studied extensively in the literature (Cai et al., 2014b; Jiang et al., 2014; Li et al., 2014; Mitra et al., 2014; Shokouhi, 2013; Shokouhi & Radinsky, 2012; Whiting & Jose, 2014) and it has been widely adopted as a prominent feature of common search engines. We briefly summarize two aspects of recent work on QAC, i.e., popularity-based QAC and learning-based QAC, as these approaches are closely related to our work in this paper. Other work on query suggestion, like Liu, Song, Chen, Nie, and Wen (2012) and Ozertem, Chapelle, Donmez, and Velipasaoglu (2012), is related to but different from QAC, hence it is not discussed here as we only focus on the task of QAC.

2.1. Popularity-based query auto completion

A useful and straightforward approach to rank QAC candidates is to use Maximum Likelihood Estimation (MLE) based on the past popularity of queries. This approach relies on query logs to rank QAC candidates by frequency. Bar-Yossef and Kraus (2011) refer to this type of ranking as the *Most Popular Completion* (MPC) model:

$$MPC(p) = \arg\max_{q \in C(p)} w(q),\ w(q) = \frac{f(q)}{\sum_{i \in Q} f(i)}, \tag{1}$$

where $f(q)$ denotes the frequency of query q in query log Q, and $C(p)$ is a set of QAC candidates starting with prefix p. Essentially, the MPC model assumes that the current query popularity distribution is the same as what was previously observed, and hence it is reasonable to rank QAC candidates by their previous popularity in order to maximize the QAC effectiveness for all users on average.

A popularity-based QAC approach can be combined with other criteria. For instance, after returning a QAC ranked list sorted by popularity, Bar-Yossef and Kraus (2011) treat the user's recent queries as context and exploit users with common search activities for a QAC re-ranking task by considering the similarity between QAC candidates and this context.

However, query popularity may change over time. Consequently, QAC rankings must be adjusted to account for time-sensitive changes. Two options have been considered to this end, using: (i) the observed query frequency within a fixed time window as $f(q)$ in (1) rather than the observation within the whole log; (ii) the predicted query frequency $f'(q)$ to replace $f(q)$ in (1). For the former, Whiting and Jose (2014) propose several practical QAC ranking approaches, generating QAC candidates by query popularity observed within a sliding time window ranging from the past 2 to 28 days or within a recent query chunk observed with a given prefix. For the latter, methods from time series analysis have been used for predicting query frequency. Shokouhi and Radinsky (2012) propose a long-term time series modeling approach to forecast the query frequency by applying a fixed moving time window. Queries recurring within specific temporal intervals, e.g., day/night, workday/weekend and summer/winter, are modeled differently for predicting the future popularity at different times. Another approach to predict query popularity is based on search trend prediction. Golbandi, Katzir, Koren, and Lempel (2013) develop a regression model to detect bursting queries for enhancing trend detection. By analyzing query logs, they seek to accurately predict the most trending query terms on the Web. Various attempts have been made to make search trend prediction more accurate with low latency relative to the actual event that sparked the trend.

[1] http://googleblog.blogspot.nl/2004/12/ive-got-suggestion.html .

Kulkarni, Teevan, Svore, and Dumais (2011) classify queries into different categories based on the changes in popularity over time, reporting that monitoring query popularity can reveal strong signals for detecting trends in query intent. Cai et al. (2014b) rank QAC candidates by predicted query popularity and Whiting and Jose (2014) also propose a short-range query popularity prediction approach based on recently observed trends.

None of the work listed above has considered possible contributions from homologous queries for a given query candidate. However, we hypothesize that homologous queries for a query q likely address the same search intent as q. To the best of our knowledge, we are the first to develop features from such homologous queries for QAC.

2.2. Learning-based query auto completion

In most QAC work mentioned so far, for a given prefix, QAC candidates are identified based on popularity. Learning-based QAC approaches have gathered less attention so far. This may be due to the fact that limited features can be developed to capture the relation between a prefix and a query. This problem can be tackled by exploiting the user's personal context (Jiang et al., 2014; Liao et al., 2011; Santos, Macdonald, & Ounis, 2013; Shokouhi, 2013).

Shokouhi (2013) exploits user profiles to extract user-based features to model the likelihood of query candidates and explores the effectiveness of the user's age, gender, location and search history for learning to personalize query auto completion. In other words, this model is learnt to find similar users who share common search activities with the current user. Guo, Cheng, Xu, and Zhu (2011) propose a two-step approach for ranking QAC candidates by learning a user's search sessions as context to match against pre-generated topic models. Similarly, Cao et al. (2008) and Liao et al. (2011) match the user's context, captured by their recent queries, against the query clusters learnt from the click graph for ranking QAC candidates. Other aspects of user context building on temporal intuitions, e.g., taking the search time as a user-specific context, have also been considered for query completion (Sengstock & Gertz, 2011).

Recently, user interactions have begun to play a more prominent role in algorithms for QAC. Jiang et al. (2014) investigate the feasibility of exploiting the context to learn user reformulation behavior and propose a supervised approach for query auto completion, where term-, query- and session-level features of user reformulation behavior are developed. This approach can significantly improve the performance of existing context-aware QAC methods. In addition, with fine-grained user interaction information, Li et al. (2014) observe a horizontal skipping bias and a vertical position bias in the QAC process; they propose a two-dimensional click model for modeling the QAC process Chuklin, Markov, and de Rijke (2015). Interestingly, Mitra et al. (2014) investigate user interaction patterns with QAC in Bing and suggest that users are most likely to engage with auto-completion at word boundaries. They also notice that the likelihood of using auto-completion varies with the distance of query characters on the keyboard. These findings provide valuable insights for understanding user QAC engagement.

Most learning-based QAC approaches (re-)rank QAC candidates by measuring their similarity with the content in the search logs, overlooking the semantic relatedness of term pairs inside a query itself while these terms are not randomly combined when a user generates his query. To the best of our knowledge, there is no published work on QAC that considers the semantic relatedness of term pairs within a query. We hypothesize that injecting semantic features into a learning to rank approach to QAC boosts the QAC performance. Below, we introduce a learning to rank approach for QAC that uses features inferred from both homologous queries and semantic relatedness as well as user reformulation behavior features previously studied in the literature for QAC.

3. Learning to rank query auto completions

Here, we will formally describe the problem of learning to rank (L2R) for query auto completion (QAC), and then propose a number of L2R-based QAC models. Four categories of query features that are likely to affect QAC rankings are taken into account: query popularity, user reformulation behavior, features inferred from homologous queries, and semantic features. See Tables 2–4 below for a tabular overview. We use LambdaMART (Burges et al., 2011) to re-rank the top N QAC candidates initially returned by an MPC baseline. Any reasonable L2R algorithm can be employed to obtain our ranking model and will likely yield similar results; we choose LambdaMART because it has been shown to be one of the best algorithms for L2R tasks (Shokouhi, 2013).

3.1. Overview

We begin by describing the L2R problem for QAC. The input of a typical QAC task is a prefix $p_i = \{char_1, char_2, ..., char_{n_c}\}$, i.e., a string of n_c characters. Generally, there is a pool $Q_c(p_i)$ consisting of query candidates that start with the prefix p_i, and each candidate could be issued after typing p_i. These candidates could be straightforwardly collected by sorting them by popularity. Each candidate in $Q_c(p_i)$ can be represented as a prefix-query pair feature vector qf.

In a typical feature-based L2R framework for IR, the training data for a specific learning algorithm (whether pointwise, pairwise or listwise) consists of a set of query-document pairs with relevance labels, and the goal is to learn a ranking model by optimizing a loss function (Liu, 2003). Similarly, in a L2R framework for QAC, coupled with a slight difference in assigning labels to query candidates for training, the model is trained on prefix-query pairs with binary labels, i.e., "submitted" or

Table 1
Comparison of learning to rank for document retrieval (DR) vs. for query auto completion (QAC). In a document retrieval task, for a given query q_i, each of its associated documents d can be represented by a feature vector $df = \Phi(d, q)$, where Φ is a feature extractor; $m^{(i)}$ is the number of documents associated with query q_i, i.e., D. In a QAC task, for a given prefix p_i, each of its auto completed query candidates q can be represented by a feature vector $qf = \phi(p, q)$, where ϕ is a feature extractor; $n^{(i)}$ is the number of query candidates associated with prefix p_i), i.e., $Q_c(p_i)$.

DR	Input	Query $q_i = \{term_1, term_2, \ldots, term_m\}$
	Ranking candidates	Documents $d^{(i)} = \{df_j^{(i)} \in D\}_{j=1}^{m^{(i)}}$
	Candidate features	TF, IDF, BM25, etc.
	Output	A ranked list of documents $df_u^{(i)} \succ df_v^{(i)} \succ \cdots \succ df_l^{(i)}$
	Ground truth	Mostly multi-level relevance labels, i.e., 0, 1, 2.
	Major metrics	MAP, P@K, NDCG@K, etc.
QAC	Input	Prefix $p_i = \{char_1, char_2, \ldots, char_n\}$
	Ranking candidates	Queries $q^{(i)} = \{qf_j^{(i)} \in Q_c(p_i)\}_{j=1}^{n^{(i)}}$
	Candidate features	Popularity, length, position in session, etc.
	Output	A ranked list of queries $qf_u^{(i)} \succ qf_v^{(i)} \succ \cdots \succ qf_l^{(i)}$
	Ground truth	Binary labels: 1 for submitted query and 0 for the others.
	Major metrics	MRR, SR@K, etc.

Table 2
Summary of popularity features of QAC candidates (10 features) and the corresponding formulas. The periods considered for the popularity features are *whole*, 1-*day*, 2-*day*, 4-*day*, 7-*day* for observations and predictions.

Period	Description	Formula
whole, 1-, 2-, 4-, 7-*day*	Observation	*fre(q, period)* in Section 3.2.1
1-, 2-, 4-, 7-*day*	Predicted by trend	$\hat{y}_{t_0}(q, i)_{trend}$ as in (2)
whole	Predicted by periodicity	$\hat{y}_{t_0}(q)_{peri}$ as in (3)

Table 3
Summary of popularity features of homologous queries for QAC candidates (60 features) and the corresponding formulas. There are two categories of homologous queries, *super queries* and *pseudo-identical queries*. The weight of super queries can be determined using either query term overlap (Section 3.3.1) or common prefix weighting (Section 3.3.2). We consider two choices for each feature: maximum or summation as described in Section 3.3.

Period	Impact	Description	Formula
Super query			
whole, 1-, 2-, 4-, 7-*day*	w in (6) or (7)	Observation	*fre*$(q'$, *period*$)$ in Section 3.2.1
1-, 2-, 4-, 7-*day*	w in (6) or (7)	Predicted by trend	$\hat{y}_{t_0}(q', i)_{tre}$ in (2)
whole	w in (6) or (7)	Predicted by periodicity	$\hat{y}_{t_0}(q')_{per}$ in (3)
Pseudo-identical query			
whole, 1-, 2-, 4-, 7-*day*	$w = 1$	Observation	*fre*$(q'$, *period*$)$ in Section 3.2.1
1-, 2-, 4-, 7-*day*	$w = 1$	Predicted by trend	$\hat{y}_{t_0}(q', i)_{tre}$ in (2)
whole	$w = 1$	Predicted by periodicity	$\hat{y}_{t_0}(q')_{per}$ in (3)

Table 4
Summary of semantic features of candidate (14 features) and the corresponding formulas. We consider two choices for each feature: maximum or summation and there are 2 sources for determining the word2vec scores, the GoogleNews corpus or one of our query logs (AOL or MSN).

Description	Formula
Word2vec score from GoogleNews or query logs	word2vec in Section 3.4.1
Lexical similarity score from query logs	$f_{WordSimMax}$ and $f_{WordSimSum}$ in Section 3.4.1
Term pair likelihood ratio from query logs	f_{LLRMax} and f_{LLRSum} as (8)
Term pair cooccurrence frequency from query logs	f_{cof_max} and f_{cof_sum} in Section 3.4.1
Lexical query similarity score from query logs	$f_{QueSimMax}$ and $f_{QueSimSum}$ in 3.4.2
Temporal relation from Query logs	$f_{TemRelMax}$ and $f_{TemRelSum}$ as in (9)

"non-submitted," similar to the "relevance" label of query-document pairs in L2R for document retrieval. For reference, Table 1 compares L2R for document retrieval to L2R for QAC.

3.2. Popularity-based features

Popularity-based features are extracted from two sources: the query candidate (Table 2) and its homologous queries (Table 3). Both the observed and the predicted query frequency are introduced.

3.2.1. Popularity from observations

We observe the query popularity within various periods, producing pairs of time-sensitive features, denoted as *fre*(*q, period*). The period, identified by a changing time window, can be chosen from the set {*1-day, 2-day, 4-day, 7-day, whole*}, where *whole* indicates that the popularity is determined based on the entire training query log while the others are collected from logs for the recent 1, 2, 4, or 7 day(s) only. We only focus on these five period options as: (i) our available datasets for learning are not big enough to sensibly consider longer intervals, e.g., the MSN log that we use only covers a one month period; (ii) the sliding windows chosen as part of the time-sensitive approach to QAC by Whiting and Jose (2014) can successfully reveal recent trends of query popularity.

3.2.2. Popularity from predictions

Following Cai et al. (2014b), we generate features based on predicted query popularity according to the recent trend and according to cyclic phenomena. Specifically, we first detect the trend of query *q*'s popularity from the first-order derivative of its daily count $C(q, t)$ observed at different time points *t*, and then predict its future popularity at day t_0, denoted by $\hat{y}_{t_0}(q,i)_{tre}$, based on the observation of each preceding *i*th day as:

$$\hat{y}_{t_0}(q,i)_{tre} = y_{t_0-i}(q,i) + \int_{t_0-i}^{t_0} \frac{\partial C(q,t)}{\partial t}\mathrm{d}t, \tag{2}$$

where $y_{t_0-i}(q,i)$ is the observed frequency of *q* at the preceding *i*-th day. For simplicity, similar to the choices of *period* in Section 3.2.1, we consider four options for *i*, i.e., $i \in \{1, 2, 4, 7\}$.

In addition, we predict the query popularity according to its periodicity of query volume, denoted by $\bar{y}_{t_0}(q)_{per}$. We detect cyclic phenomena of query popularity at a per hour level and then produce an aggregated query popularity at a per day level. We smooth $\bar{y}_{t_0}(q)_{per}$ by simply averaging the recent *M* observations y_{t_p} at the preceding time points $t_p = t_0 - 1 \cdot T_q, \ldots, t_0 - M \cdot T_q$ in the log:

$$\hat{y}_{t_0}(q)_{per} = \frac{1}{M}\sum_{m=1}^{M} y_{t_0-m\times T_q}(q), \tag{3}$$

where T_q denotes query *q*'s periodicity. For detecting cyclic aspects of *q*'s frequency, we use autocorrelation coefficients (Chatfield, 2004), which measure the correlation between N_s successive count observations $C(q, t)$ at different times $t = 1, 2,$ ..., N_s in the query log. The correlation is computed between a time series and the same series lagged by *i* time units:

$$r_i = \frac{\sum_{t=1}^{N_s-i}(C(q,t)-\bar{x}_1)(C(q,t+i)-\bar{x}_2)}{(\sum_{t=1}^{N_s-i}(C(q,t)-\bar{x}_1)^2)^{\frac{1}{2}}(\sum_{t=i+1}^{N_s}(C(q,t+i)-\bar{x}_2)^2)^{\frac{1}{2}}}, \tag{4}$$

where $\bar{x}_1$ is the mean of the first $N_s - i$ observations and $\bar{x}_2$ is the mean of the last $N_s - i$ observations. For reasonably large N_s, the denominator in (4) can be simplified by approximation. First, the difference between the sub-period means $\bar{x}_1$ and $\bar{x}_2$ can be ignored. Second, the difference between summations over observations 1 to $N_s - i$ and $i + 1$ to N_s can be ignored. Consequently, r_i can be approximated as follows:

$$r_i \approx \frac{\sum_{t=1}^{N_s-i}(C(q,t)-\bar{x})(C(q,t+i)-\bar{x})}{\sum_{t=1}^{N_s}(C(q,t)-\bar{x})^2}, \tag{5}$$

where $\bar{x} = \sum_{t=1}^{N_s} C(q,t)$ is the overall mean.

3.3. Weighting homologous queries

Now, given a query *q*, *Hom*(*q*), the set of homologous queries for *q*, consists of all super queries s_q and pseudo-identical queries p_q of *q* found in the query logs, which have been issued before *q* was submitted. We extract the same popularity features discussed in Section 3.2 for each homologous query $q'_c \in Hom(q_c)$ for a candidate q_c; so q'_c is either a super query s_q or a pseudo-identical queries p_q of *q*. We are not going to use all homologous queries q'_c for a candidate query q_c with the same weight when generating the final popularity features of homologous queries. Instead, we measure how similar q'_c and q_c are with weight *w* to be able to weigh the contribution of q'_c—this will allow us to capture the popularity of q_c from a homologous query q'_c.

After discounting the popularity of homologous queries, we calculate the maximal and aggregated values from all homologous queries as features. Next, we will introduce two approaches for discounting.

3.3.1. Query term overlap weighting

Clearly, the more terms the homologous query q'_c and q_c share, the more relevant they could be to each other. Let $q'_c \cap q_c$ refer to the set of common terms in q'_c and q_c. We multiply the term overlap ratio as a discount $Discount(q'_c, q_c)$ with the popularity of super query s_q of candidate q_c:

$$w \leftarrow Discount(s_q, q_c) = \frac{|s_q \cap q_c|}{|s_q|}, \tag{6}$$

where $|\cdot|$ returns the number of terms of the input term set.

For pseudo-identical queries p_q for q_c, we set $Discount(p_q, q_c) = 1$ as we assume that they enjoy the same search popularity.

3.3.2. Common prefix weighting

In addition to the query term set overlap discounting, we may also posit that a longer common prefix matters more than shorter ones. This can be captured by introducing an impact factor based on the length of the overlapping prefix. Thus, we discount the popularity of super query s_q with $Impact(s_q, q_c)$ as:

$$w \leftarrow Impact(s_q, q_c) = \frac{\|CommonPrefix(s_q, q_c)\|}{\|s_q\|}, \tag{7}$$

where $\| \cdot \|$ returns the number of characters of the input string. Rather than emphasizing the overlapping terms, this emphasizes the longest common prefix. For pseudo-identical queries p_q of q_c we assign an impact $Impact(p_q, q_c) = 1$.

3.4. Semantic features

We use semantic information in two ways. Our example in the introduction to the paper suggests that queries semantically related to queries submitted earlier in a session may be more likely to be good completion candidates. We also estimate the semantic relatedness of words occurring in a query on the assumption that semantically more coherent queries may be better query candidates. We start with the latter.

3.4.1. Semantic relatedness in queries

We use several ways of computing semantic relatedness between terms in a query. To begin, we use a lexical similarity method that combines POS tagging, Latent semantic analysis and WordNet to determine term-level similarity (Han, L. Kashyap, Finin, Mayfield, & Weese, 2013). Given two words, it returns a string representing a number between 0.0 and 1.0 with 1.0 indicating absolutely similar. We use it to capture the word similarity, resulting in two features $f_{WordSimMax}$ and $f_{WordSimSum}$ obtained by maximizing and summing all similarity scores of possible term pairs in a query, respectively.[2]

Next, we consider the query term pair likelihood ratio. Jones, Rey, Madani, and Greiner (2006) observe that frequent query pairs from search sessions can be found by statistical hypothesis testing. Given two queries, the likelihood ratio (LLR) between them can be calculated and the LLR testing is performed to see whether their co-occurrence in a search session is statistically significant. Similarly, we argue that frequently co-occurring term pairs in a query could be semantically related and introduce the LLR score to capture the semantic relatedness of term pairs inside a query. More specifically, assuming that there are two terms $term_1$ and $term_2$ in a query, we calculate the LLR score for this term-pair as:

$$LLR(term_1, term_2) = -2\log\frac{L(term_2 \mid \neg term_1)}{L(term_2 \mid term_1)}, \tag{8}$$

where $L(term_2|\neg term_1)$ denotes the number of queries containing $term_2$ but without $term_1$, and $L(term_2|term_1)$ indicates the volume of queries containing both $term_1$ and $term_2$. High LLR scores are assumed to indicate semantic relatedness. We calculate LLR scores for all possible term pairs in a query q (excluding stop words) to measure the semantic relatedness. Then, we return the maximal and aggregated LLR scores of all term pairs as semantic features, resulting in f_{LLRMax} and f_{LLRSum}. In addition, we use the larger LLR score of a term pair $(term_i, term_j)$ in a query q in spite of the term order, and then produce the maximal and aggregated LLR scores as partial semantic features, indicated as

$$f_{cof\_max} = \max_{term_{i,j} \in q}(\max(L(term_i \mid term_j), L(term_j \mid term_i)))$$

and

$$f_{cof\_sum} = \sum_{term_{i,j} \in q} \max(L(term_i \mid term_j), L(term_j \mid term_i)).$$

Our third way of operationalizing semantic relatedness between query terms builds on the method described by Mikolov, Chen, Corrado, and Dean (2013a) and known as word2vec, where vector representations of words are learned from large amounts of unstructured text resources, e.g., GoogleNews. Representations of words as continuous vectors have been shown to capture meaningful semantic word regularities (Mikolov, Sutskever, Chen, Corrado, & Dean, 2013b). We use the idea to capture semantic relatedness between query terms. The training objective of the Skip-gram model is to learn word vector representations that are good at predicting the surrounding words by maximizing the average log probability

$$\frac{1}{T_r}\sum_{t=1}^{T_r} \sum_{-c_s <= j <= c_s, j \neq 0} \log P(term_{t+j} \mid term_t),$$

[2] We apply a web API released by UMBC that can be used to return the semantic similarity score between words, http://swoogle.umbc.edu/SimService/api.html.

with inputting a sentence of terms $term_1$, $term_2$, ..., $term_{T_r}$, where c_s is the size of training context. This setup allows us to assign a word2vec score, learnt from GoogleNews, to each term pair in a query. In this manner we produce $f_{W2vGooMax}$ and $f_{W2vGooSum}$ by maximizing and summing scores of all term pairs.

As the trained word representation is a data-driven approach, highly dependent on the text resource and limited by its inability to represent idiomatic queries or rare words unseen in the text resources, we also train a local Skip-gram model on a query log dataset to represent query terms as a compensation and then calculate a local log-based semantic relatedness score. In other words, we input queries in the query logs one by one as a sequence of training terms instead of sentences in the unstructured text resource. Based on the word2vec model learnt from the query logs, the maximal and aggregated word2vec scores of term pairs are returned, resulting in features $f_{W2vLogMax}$ and $f_{W2vLogSum}$.

3.4.2. Semantic relatedness in sessions

So far, we have considered the semantic relatedness between words inside a query. Next, we consider the semantic relatedness between query candidates and previous queries in the same session. We use alternative methods to capture this relation.

First, we capture query relatedness between query candidates and previous queries in a session at the lexical level, by using the combination of latent semantic analysis, POS tagging and WordNet mentioned before; this results in the features $f_{QueSimMax}$ and $f_{QueSimSum}$.

Second, following Chien and Immorlica (2005), we investigate query relatedness using temporal correlations. In other words, two queries are assumed to be semantically related in this sense if their popularities behave similarly over time. We employ Pearson's correlation coefficient, commonly represented by the letter r, to capture this notion of similarity between candidate q_c and previous query q_x. Similar to (5), we can obtain a formula for $r(q_c, q_x)$ as:

$$r(q_c, q_x) = \frac{1}{n_u}\sum_{i=1}^{n_u}\left(\frac{q_{c,i}-\mu(q_c)}{\delta(q_c)}\right)\left(\frac{q_{x,i}-\mu(q_x)}{\delta(q_x)}\right), \quad (9)$$

where the frequency of query q_c (or q_x) over n_u days is an n_u-dimensional vector $q_c = (q_{c,1}, q_{c,2}, \ldots, q_{c,n_u})$ with $\mu(q_c)$ and $\delta(q_c)$ indicating the mean frequency and the standard deviation of its frequency, respectively. The correlation $r(q_c, q_x)$ of two queries q_c and q_x is a standard measure of how strongly two queries are linearly related. It always lies between $+1$ and -1, with $+1$ implying an exactly positive linear relationship between them and -1 for an absolutely negative linear relationship. Again, we calculate the maximal and aggregated query relatedness scores of all query pairs as temporal semantic features, resulting in $f_{TemRelMax}$ and $f_{TemRelSum}$, respectively.

3.5. Summary

We also make use of user reformulation behavior features following Jiang et al. (2014), derived at three levels: term-, query- and session-level. Please refer to Jiang et al. (2014) for details. The use of features of homologous queries and semantic relatedness for QAC is a new contribution of this paper. In total, we use 127 features including those from Jiang et al. (2014).

Our features are slotted into three categories: popularity, user reformulation behavior, and semantic relatedness. For query popularity, we collect evidence from the candidate and its homologous queries, based on observations and predictions. For user behavior features, we directly use those from Jiang et al. (2014). Regarding homologous queries, there are 2 categories, 5 period options for observation (4 for prediction by trend), 2 weighting schemes (only for super queries) and 2 calculations, accounting for a total of 60 features, i.e., 30 features by observation and 30 features by prediction (24 from trend and 6 from periodicity). Referring to the candidate's popularity, recent observations and predictions are collected by simply changing the time period, yielding $5+4+1=10$ features. For semantic relatedness features, 2 sources are used to calculate the word2vec score: GoogleNews and a local query log, i.e., AOL (MSN) queries are scored by the corresponding word2vec model trained on the AOL (MSN) logs; 2 calculations are adopted for generating popularity features of homologous queries and semantic features: maximization and summation; 2 term-pair significant scores are included; semantic relatedness is measured on word- and query-level, resulting in 14 features.

Accordingly, in total, we develop 70 ($=60+10$) features for popularity, 14 features for semantic relatedness and 43 features for user reformulation behavior, yielding a total of $70+14+43=127$ features for our L2R-based QAC approach.

4. Experimental setup

Section 4.1 summarizes the proposed models trained on different features and lists the research questions to guide our experiments; Section 4.2 describes our datasets; Section 4.3 gives details about our evaluation metrics and baselines; we detail further experimental settings and parameters in Section 4.4.

4.1. Model summary and research questions

To examine the contribution from each specific feature source, we specify six L2R-QAC variations depending on the features used: L2R-U, -UP, -UPS, -UPH, -ALL and -TOP; see Table 5.

Table 5
An overview of the QAC models discussed in the paper.

Model	Description	No. of features	Source
Baselines			
MPC-ALL	Ranking QAC candidates according to their past popularity in the whole log	–	(Bar-Yossef & Kraus, 2011)
MPC-R	Ranking QAC candidates according to their past popularity within recent R days	–	(Whiting & Jose, 2014)
L2R-U	Learning to rank QAC candidates using user reformulation behavior features	43	(Jiang et al., 2014)
Our models			
L2R-UP	Extending L2R-U by adding 10 popularity features of the same query	$43 + 10 = 53$	This paper
L2R-UPS	Extending L2R-UP by adding 14 semantic features of the same query	$53 + 14 = 67$	This paper
L2R-UPH	Extending L2R-UP by adding 60 popularity features of homologous queries	$53 + 60 = 113$	This paper
L2R-ALL	All features are considered for learning to rank QAC candidates	$53 + 14 + 60 = 127$	This paper
L2R-TOP	Extending L2R-UP by only adding top ten important features newly developed here	$53 + 10 = 63$	This paper

Our research questions guiding the remainder of the paper are:

RQ1 Do the features that describe the observed and predicted popularity of a QAC candidate help boost QAC performance without negatively impacting the effectiveness of user behavior related features proposed by Jiang et al. (2014)? That is, how does L2R-UP compare against L2R-U? (See Section 5.1.)
RQ2 Do semantic features help improve QAC performance? That is, how does L2R-UPS compare against L2R-UP? (See Section 5.2.)
RQ3 Do homologous queries help improve QAC performance? That is, how does L2R-UPH compare against L2R-UP? (See Section 5.3.)
RQ4 How does L2R-UPS compare against L2R-UPH? What is the performance gain, if any, if all features are added for learning (L2R-ALL)? (See Section 5.4.)
RQ5 What are the principal features developed here for a learning to rank based QAC task? (See Section 5.5.)

4.2. Dataset

We use two publicly available query log datasets in our experiments: AOL (Pass, Chowdhury, & Torgeson, 2006) and MSN (Craswell, Jones, Dupret, & Viegas, 2009). The AOL queries were sampled between March 1, 2006 and May 31, 2006, while the MSN logs were recorded for one month in May 2006. For consistency, we partition each log into two parts: a training set consisting of 75% of the query log and a test set consisting of the remaining 25% in terms of time period. We also use the temporally last 10% samples in the training set as validation set for LambdaMART. Traditional k-fold cross-validation is not applicable to a streaming sequence since it would negate the temporal order inherent in the data (Gama, Žliobaitė, Bifet, Pechenizkiy, & Bouchachia, 2014). Queries in the training set were submitted before May 8, 2006 in the AOL dataset and before May 24, 2006 in the MSN dataset.

We filter out a large volume of navigational queries with URL substrings (.com, .net, .org, http, .edu, www, etc.) and remove queries starting with the special characters (&, $, #, etc.) from both datasets. We divide the queries into sessions by 30 minutes' inactivity[3] and only English queries appearing in both two partitions were kept. Importantly, in our experimental design we follow Jiang et al. (2014) and focus on sessions with at least two queries. By doing so, we can extract user behavior related features. Like previous session context-based QAC approaches (Bar-Yossef & Kraus, 2011; Jiang et al., 2014), we set the prefix in our experiments to be the first 1–5 character(s) of queries in a session. To obtain our training and test sets, we remove input prefixes for which the ground truth (see below) is not included in the top ten QAC candidates returned by MPC at the time point of querying; this too follows previous QAC work and is a commonly used methodology in QAC tasks (Cai et al., 2014b; Jiang et al., 2014; Shokouhi, 2013). Table 6 details the statistics of our processed datasets.

The ground truth for a QAC task is defined as follows. Given a search session with T queries, i.e., $\{q_1, q_2, \ldots, q_{T-1}, q_T\}$, we want to predict each intended query $q_i, i = \{1, 2, \ldots, T\}$ at position i of a session after typing its prefix p. Then, a query q is a *correct completion* of this prefix p of q_i if, and only if, $q = q_i$.

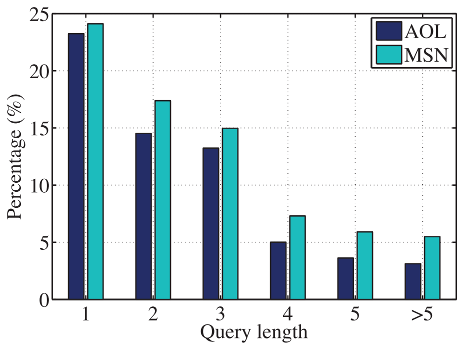
Next we take a closer look at the processed datasets so to be able to report the ratio of queries at various lengths in words and of queries that have homologous queries. As shown in Fig. 3a, generally, for both datasets, nearly one quarter of the one-term queries have homologous queries, solely contributed by their super queries. For two- and three-term queries, super queries and pseudo-identical queries contribute similarly, however for longer queries (>3 words), the latter dominate the contribution.

Additionally, as plotted in Fig. 3b, more than half of all queries contain more than one word, which supports the feasibility for semantic features between words inside a query.

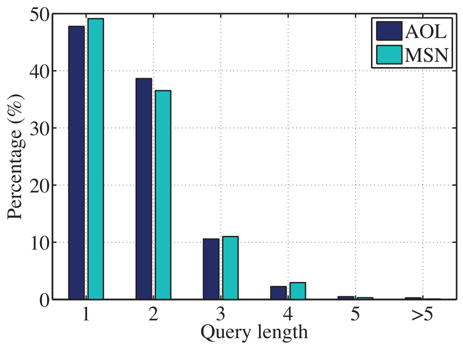
[3] Only applied on the AOL dataset as the MSN dataset provides session IDs.

Table 6
Statistics of the AOL and MSN datasets used. The "⋆" in # Prefix-⋆ indicates the length of the prefix in characters.

	AOL		MSN	
Variables	Training	Test	Training	Test
# Queries	3,808,083	1,571,346	3,784,925	1,402,308
# Unique queries	452,980	452,980	304,943	304,943
# Sessions	1,149,528	465,302	674,717	256,512
Queries / session	3.31	3.38	5.60	5.46
# All prefixes	8,783,957	3,260,130	4,995,213	1,751,158
# Prefix-1	605,710	209,650	427,502	141,925
# Prefix-2	1,175,087	405,857	749,821	249,065
# Prefix-3	1,954,285	707,580	1,134,539	387,633
# Prefix-4	2,433,385	916,976	1,320,529	470,286
# Prefix-5	2,615,490	1,020,067	1,362,822	502,249



(a) The ratio of queries possessing homologous queries.



(b) Distribution of queries with various number of terms.

Fig. 3. Statistics of queries at various query lengths (in words) for the AOL and MSN datasets, respectively.

4.3. Evaluation metrics and baselines

To evaluate the effectiveness of QAC rankings, Mean Reciprocal Rank (MRR) is a standard measure. For a query q with prefix p in the query set Q associated with a list of QAC candidates $S(p)$ and the user's finally submitted query q', the Reciprocal Rank (RR) is computed as:

$$RR = \begin{cases} \frac{1}{\text{rank of } q' \text{ in } S(p)}, & \text{if } q' \in S(p) \\ 0, & \text{else.} \end{cases} \tag{10}$$

Then, MRR is computed as the mean of RR scores for all queries in Q. The choice of MRR as a performance metric is common in settings that are concerned with finding a single known solution. However, because of the issues reported by Hofmann, Mitra, Radlinski, and Shokouhi (2014) on the formulation of MRR (averaging over a non ratio-scale), we introduce a less problematic metric. The *success rate at top K* (SR@K), denoting the average ratio of the actual query that can be found in the top K queries over the test data, can be used for tasks whose ground truth consists of only one instance such as query completion (Jiang et al., 2014).

For comparison, we consider several QAC baselines: (1) the Most Popular Completion (MPC) model based on the frequency in the whole log, referred to as MPC-ALL (Bar-Yossef & Kraus, 2011); (2) an MPC based QAC method, using frequency within a recent time window, denoted as MPC-R (Whiting & Jose, 2014), which is state-of-the-art. Here we set $R = 7$ days as the time window because performance peaks with this setting (Whiting & Jose, 2014); (3) a recent L2R-based QAC model considering user reformulation behavior features (Jiang et al., 2014), denoted as L2R-U. The former two baselines are popularity-based while L2R-U is a L2R-based QAC model.

Note that we do not compare our approach with query suggestion methods, e.g., (Liao et al., 2011; Ma, Yang, King, & Lyu, 2008; Mei, Zhou, & Church, 2008), because such methods focus on the task of query suggestion, which is a subtly different task from QAC (Cai et al., 2014b). For instance, in the case of query suggestion, the user input typically is a full *query* but it is only a *prefix* in QAC. Also, for generating the original query completion list to rerank in our QAC task, only the candidates matching the typed prefix are taken into consideration. However, for a query suggestion task, each query in the query log could be included in the candidate list to rerank given that it is deemed *relevant* to the user input. With

Table 7
Selecting our baseline. The performance of various baselines in terms of MRR and SR@K, tested on the AOL and MSN datasets after typing one character as prefix. The best performing baseline in each row is highlighted.

Dataset	Metric	MPC-ALL	MPC-R	L2R-U
AOL	MRR	0.6157	0.6348	**0.6682**
	SR@1	0.4532	0.4643	**0.4815**
	SR@2	0.5914	0.6038	**0.6256**
	SR@3	0.7016	0.7121	**0.7304**
MSN	MRR	0.6305	0.6498	**0.6821**
	SR@1	0.4702	0.4757	**0.4876**
	SR@2	0.6083	0.6276	**0.6385**
	SR@3	0.7251	0.7368	**0.7437**

this limitation we follow a standard practice in research on QAC (Bar-Yossef & Kraus, 2011; Cai et al., 2014b; Shokouhi & Radinsky, 2012; Whiting & Jose, 2014), where only appropriate QAC approaches are used as baselines rather than query suggestion approaches. In addition, QAC is different from query correction oriented tasks such as those discussed by Duan and Hsu (2011), where the main focus is centered on correctly modifying the misspelled word or query.

The statistical significance of observed differences between the performance of two approaches is tested using a two-tailed paired *t*-test and is denoted using $^{\blacktriangle}/^{\blacktriangledown}$ for significant differences for $\alpha = .01$, or $^{\vartriangle}/^{\triangledown}$ for $\alpha = .05$.

To select a single baseline against which we compare our newly introduced models, we compare the performance of the three baselines just listed and report the results in Table 7. For both datasets, L2R-U outperforms the other two baselines. For instance on AOL, it results in more than 8% and 5% improvements in MRR over MPC-ALL and MPC-R, respectively, after typing one character as prefix. Similar results can be found on the MSN dataset. Hence, we select L2R-U as the single baseline for comparison with our proposed models in later experiments.

4.4. Settings and parameters

Following Cai et al. (2014b), for time-sensitive query popularity prediction, we count queries per hour to detect the periodicity and aggregate the hour-predictions within the same day to generate the day-volume. For smoothing in (3), we set $M = 3$ as it performs best when M changes from 1 to 5 in our trials. Before we run our L2R-based QAC experiments, we are given a list of top N QAC candidates by the traditional MPC approach; we set $N = 10$ as this is commonly used by many web search engines and published QAC work (Cai, Liang, & de Rijke, 2014a; 2014b; Jiang et al., 2014; Shokouhi, 2013). We use the LambdaMART learning algorithm for ranking QAC candidates across all experiments (Burges et al., 2011).

5. Results and discussions

In Section 5.1, we examine the performance of L2R-UP, which we follow with a section about the contribution of semantic features. We examine the performance of L2R-UPH in Section 5.3 with features of homologous queries added to L2R-UP. Next, Section 5.4 details the performance of L2R-ALL learnt from all features above and Section 5.5 provides an analysis of feature importance. Finally, Section 5.6 zooms in on the impact of query position on QAC performance.

5.1. Effect of query popularity

Since information about the past popularity of QAC candidates can be generated offline before ranking while the true popularity is unavailable at runtime, we develop the popularity features according to their previously observed frequency within various periods in the query logs, known as time-sensitive popularity features. In contrast, we produce the predicted query popularity as features based either on recent trends or on cyclic patterns. In this section, we compare the performance of L2R-UP with that of the baseline.

Table 8 includes the results on two datasets, i.e., the AOL and MSN datasets, after entering prefixes consisting of 1 to 5 characters. On each dataset, L2R-UP generally performs better than the baseline (L2R-U) in terms of MRR.

When we take a closer look at the results across all prefixes, reported in MRR scores in Table 8, L2R-UP is considerably more effective on longer prefixes as it produces larger MRR improvements over the baseline on long prefixes, e.g., $\# = 4$ or 5. Longer prefixes notably reduce the space of possible QAC candidates, which simplifies the problem. In addition, L2R-UP shows a monotonous increase in MRR as the prefix length goes up. However, the baseline shows a bit fluctuation in terms of MRR as the prefix length changes. Expectedly, L2R-UP always receives the higher MRR scores compared to the baseline.

Next, we compare L2R-UP against the baseline (L2R-U) in terms of SR@1 and plot the results in Fig. 4. We find that, at each prefix length, for more than half of the test prefixes, L2R-UP returns the final submitted query at the first position in the QAC ranking list because all SR@1 scores achieved by L2R-UP are higher than 0.5; L2R-U receives a lower SR@1 score than 0.5 on both datasets; long prefixes invariably result in higher SR@1 scores than short ones. From these findings, we

Table 8
Performance in terms of MRR for the QAC task, at a prefix length #*p* ranging from 1 to 5 characters on the AOL and MSN datasets. For each dataset the best performer per row is highlighted. Statistically significant differences are determined against the baseline.

#*p*	AOL					
	Baseline	L2R-UP	L2R-UPS	L2R-UPH	L2R-ALL	L2R-TOP
1	0.6682	0.6764	0.6871$^{\vartriangle}$	0.6847$^{\vartriangle}$	**0.6977**$^{\blacktriangle}$	0.6913$^{\vartriangle}$
2	0.6631	0.6815$^{\vartriangle}$	0.6939$^{\blacktriangle}$	0.6898$^{\vartriangle}$	**0.7024**$^{\blacktriangle}$	0.6980$^{\blacktriangle}$
3	0.6654	0.6853$^{\vartriangle}$	0.7001$^{\blacktriangle}$	0.6910$^{\vartriangle}$	**0.7081**$^{\blacktriangle}$	0.7042$^{\blacktriangle}$
4	0.6673	0.6921$^{\vartriangle}$	0.7094$^{\blacktriangle}$	0.6981$^{\blacktriangle}$	**0.7144**$^{\blacktriangle}$	0.7127$^{\blacktriangle}$
5	0.6704	0.6986$^{\blacktriangle}$	0.7186$^{\blacktriangle}$	0.7059$^{\blacktriangle}$	**0.7215**$^{\blacktriangle}$	0.7201$^{\blacktriangle}$
#*p*	MSN					
	Baseline	L2R-UP	L2R-UPS	L2R-UPH	L2R-ALL	L2R-TOP
1	0.6821	0.6933	0.7028$^{\vartriangle}$	0.7011$^{\vartriangle}$	**0.7136**$^{\blacktriangle}$	0.7084$^{\vartriangle}$
2	0.6847	0.6971	0.7112$^{\vartriangle}$	0.7048$^{\vartriangle}$	**0.7204**$^{\blacktriangle}$	0.7183$^{\blacktriangle}$
3	0.6915	0.7080$^{\vartriangle}$	0.7225$^{\blacktriangle}$	0.7135$^{\vartriangle}$	**0.7287**$^{\blacktriangle}$	0.7251$^{\blacktriangle}$
4	0.6873	0.7113$^{\vartriangle}$	0.7260$^{\blacktriangle}$	0.7164$^{\blacktriangle}$	**0.7314**$^{\blacktriangle}$	0.7300$^{\blacktriangle}$
5	0.6895	0.7212$^{\blacktriangle}$	0.7366$^{\blacktriangle}$	0.7263$^{\blacktriangle}$	**0.7416**$^{\blacktriangle}$	0.7487$^{\blacktriangle}$

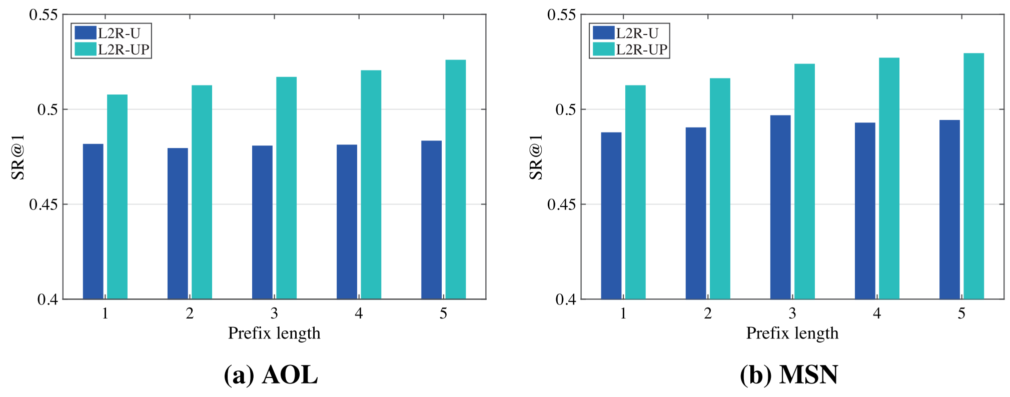


Fig. 4. QAC performance in terms of SR@1 observed for L2R-U and L2R-UP, tested on the AOL and MSN datasets.

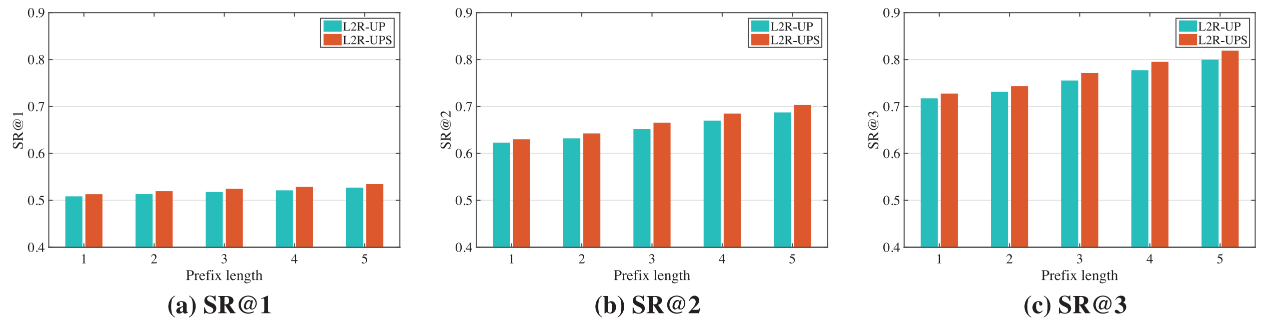


Fig. 5. QAC performance of L2R-UP and L2R-UPS tested on the AOL dataset at various prefix lengths (in characters), in terms of SR@1, SR@2 and SR@3, respectively.

conclude that the observed and predicted popularity features of query candidates indeed help generate better QAC rankings when embedded into a learning to rank framework.

5.2. Effect of semantic features

To answer **RQ2**, we learn our L2R-UPS model by extending L2R-UP with additional 14 semantic features. The MRR scores of L2R-UPS are listed in Table 8; we also plot the scores in terms of other metrics (SR@K, $K = 1, 2, 3$) of L2R-UP and L2R-UPS in Fig. 5 and 6, tested on the AOL and MSN datasets, respectively, with varying lengths of query prefix from 1 to 5.

Generally, we find that L2R-UPS beats L2R-UP for all cases on both datasets in terms of MRR and SR@K ($K = 1, 2, 3$). In particular, on the AOL dataset, the MRR improvements of L2R-UPS over L2R-UP are statistically significant (at level $\alpha = .05$)

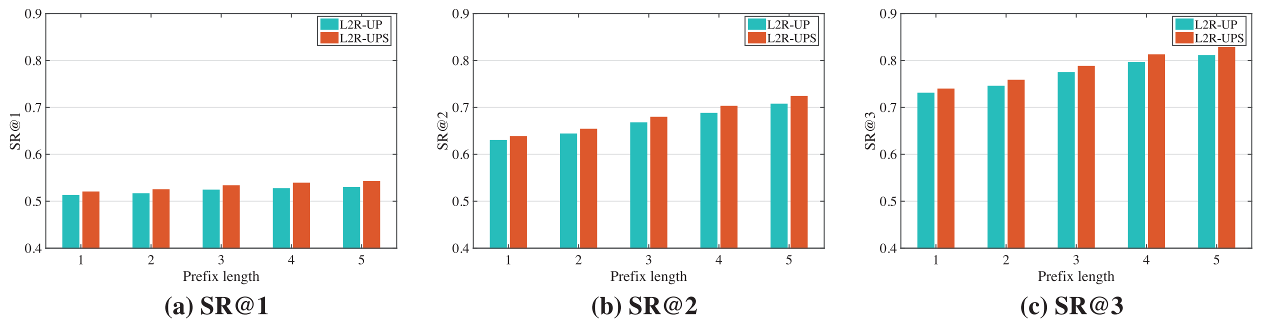


Fig. 6. QAC performance of L2R-UP and L2R-UPS tested on the MSN dataset at various prefix lengths (in characters), in terms of SR@1, SR@2 and SR@3, respectively..

Table 9
Changes in MRR and SR@1 scores between L2R-UPH and L2R-UP at varying prefix lengths on the AOL and MSN datasets.

#*p*	AOL		MSN	
	MRR	SR@1	MRR	SR@1
1	+1.23%	+1.30%	+1.13%	+1.21%
2	+1.22%	+1.19%	+1.10%	+1.09%
3	+0.83%	+0.93%	+0.78%	+0.90%
4	+0.87%	+0.86%	+0.72%	+0.82%
5	+1.04%	+1.07%	+0.71%	+0.79%

for most cases, e.g., at $\#p = 4$ and 5; however, on the MSN dataset, most of the improvements are not significant except at $\#p = 5$ (at level $\alpha = .05$). This is due to the fact that compared to the AOL dataset, the MSN dataset contains far more short queries, resulting in difficulties in capturing the term-pair semantic relatedness. In contrast, compared to the baseline (L2R-U), L2R-UPS shows significant MRR improvements for all cases on both datasets. For instance, on the AOL dataset, significant MRR improvements are observed at level $\alpha = .01$ for prefix length $\#p = 2$ to 5 and at level $\alpha = .05$ for prefix length $\#p = 1$ by comparing L2R-UPS against the baseline, respectively.

Clearly, the gains in MRR of L2R-UPS over L2R-UP are larger for longer prefixes. E.g., on the AOL dataset, L2R-UPS achieves a 2.86% improvement over L2R-UP at $\#p = 5$ and only a 1.58% improvement at $\#p = 1$ in terms of MRR. Similar observations can be made for other metrics. The results on the MSN dataset show a similar behavior even though the gaps are smaller. With longer prefixes, query candidates are more likely composed of multiple terms, which helps to extract semantic features. Regarding the outcomes in terms of SR@K, as shown in Figs. 5 and 6, for the majority of cases, L2R-UPS can return the correct query in top 3 of the QAC list as the scores in terms of SR@3 are higher than 0.8 on both datasets. Hence, we conclude that the semantic relatedness features can help generate "good" queries, in which terms are semantically close to each other.

5.3. Effect of homologous queries

Next, we turn to **RQ3** and examine the contribution from features of homologous queries for the candidate. Recall that the resulting model is called L2R-UPH. We generate the QAC rankings for each prefix; the MRR scores are included in Table 8, column 5. We see that L2R-UPH significantly outperforms the baseline, on both datasets, resulting in near 5% improvements in terms of MRR for long prefixes, e.g., for $\#p = 4$ or 5, but less for short prefixes, e.g., $\#p = 1$. Generally, L2R-UPH achieves 4.4% and 4.1% improvements over the baseline on the AOL and MSN datasets, respectively, in terms of MRR.

Additionally, we compare L2R-UPH against L2R-UP in terms of MRR and SR@1 and report the relative changes in Table 9.

Across the board, L2R-UPH outperforms L2R-UP in terms of MRR and SR@1. L2R-UPH achieves an average improvement in MRR scores around 1.2% on AOL and 0.9% on MSN over L2R-UP, respectively. For all cases, the improvement of L2R-UPH over L2R-UP is not statistically significant. Interestingly, the gains in MRR of L2R-UPH over L2R-UP are larger for shorter prefixes (e.g., $\#p = 1$ or 2), which differs from the outcomes of comparing L2R-UPS against L2R-UP where obvious MRR gains are found for long prefixes. We believe that this is due to the fact that shorter prefixes result in more ambiguous and shorter candidates, leading to a higher probability for QAC candidates to possess homologous queries from which more information can be gleaned.

Next, we zoom in on the difference between L2R-UPS and L2R-UPH. L2R-UPS tends to outperform L2R-UPH in terms of MRR at all prefix lengths (see Table 8, column 4 vs. 5), resulting in an average improvement over L2R-UPH of around 1.5% on the AOL dataset and 1.3% on the MSN dataset. The differences increase as the prefix becomes longer. Hence, even though

Table 10
Per prefix bakeoff, in terms of reciprocal rank: L2R-ALL vs. other models. The ratios (%) of test prefixes at all lengths for which L2R-ALL loses against the corresponding model listed in column 1 have a red background, ratios with equal performance have a yellow background, and those of prefixes for which L2R-ALL wins have a green background.

Model	AOL			MSN		
Baseline	🟥 4.21	🟨 52.17	🟩 43.62	🟥 5.48	🟨 54.30	🟩 40.22
L2R-UP	🟥 9.13	🟨 54.92	🟩 35.95	🟥 10.10	🟨 56.53	🟩 33.37
L2R-UPS	🟥 18.37	🟨 54.40	🟩 27.23	🟥 12.24	🟨 54.79	🟩 32.97
L2R-UPH	🟥 9.35	🟨 53.11	🟩 37.54	🟥 10.46	🟨 55.91	🟩 33.63

most differences are not statistically significant, semantic features are probably more important than those of homologous queries for QAC tasks in our setting.

In sum, homologous queries help improve the ranking of QAC candidates, reflecting the fact that searchers occasionally modify queries by changing the term order or adding terms. In addition, compared to the contribution from homologous queries, semantic features provide a bigger contribution to L2R-based QAC tasks as L2R-UPS is more effective than L2R-UPH on both datasets (AOL and MSN).

5.4. Performance of L2R-ALL

For research question **RQ4** we examine whether our L2R-ALL model learnt from all discussed features can help boost QAC ranking performance. The MRR scores achieved by L2R-ALL are listed in Table 8, column 6, on the AOL and MSN datasets. Clearly, for both datasets, L2R-ALL is considerably more effective than L2R-UPS and L2R-UPH, especially for short prefixes.

In addition, we examine the difference in MRR scores between L2R-ALL, L2R-UPS and L2R-UPH, respectively. For both datasets, the improvement of L2R-ALL over L2R-UPS is not significant. However, for all cases on AOL except $\#p = 1$ and 2, L2R-ALL significantly outperforms L2R-UPH at the $\alpha = .05$ level; for MSN, significant improvements of L2R-ALL over L2R-UPH are observed, except for $\#p = 1$ at level $\alpha = .05$. Generally, L2R-ALL achieves a 1.2% improvement in terms of MRR over L2R-UPS on both datasets. Compared to L2R-UPH, L2R-ALL shows a 2.3% MRR improvement in general. Regarding the comparisons to the baseline, L2R-ALL achieves significant improvements in terms of MRR at the $\alpha = .01$ level for all prefix lengths on both datasets. In particular, L2R-ALL achieves an average 6.8% and 6.3% MRR improvement on AOL and MSN, respectively.

Next, we check the QAC ranking performance per prefix and list the ratio of test prefixes at all lengths for which L2R-ALL loses against, equals or outperforms the corresponding models; see Table 10. We can see that, on both datasets, L2R-ALL presents a majority of draws with the other models. Actually, the draws are often observed on prefixes for which all models return the correct query submission at the top positions, e.g., 1 or 2. That is why these models receive high MRR scores (see Table 8). Additionally, compared with the other three models in Table 10, i.e., Baseline, L2R-UP and L2R-UPH, the L2R-UPS model beats L2R-ALL more often, especially on the AOL dataset, usually on long prefixes, e.g., $\#p = 4$ or 5.

5.5. Feature sensitivity analysis

Finally, we analyze the relative importance of our newly developed features to answer **RQ5**. Following (Agichtein, Castillo, Donato, Gionis, & Mishne, 2008), the top ten most significant features on each dataset used for learning, according to a χ^2 test, are reported in Table 11.

We see that the word2vec score returned by the word2vec model on the query logs, with the maximal value of all term pairs in a query, appears to be the most important feature. Generally, semantic relatedness features are more important than features of homologous queries, as they are ranked higher and account for the majority of the top ten features. Popularity features of pseudo-identical (PI) queries are notably more helpful for QAC than super queries (SQ). Also, the observations and predictions from the recent 2 days are effective. These results are consistent with the findings from previous work (e.g., (Cai et al., 2014a; Jiang et al., 2014)) that the predicted popularity dominates the QAC rankings. However, other signals also contribute many useful features, e.g., semantic query similarity represented by temporal relation and term pairwise occurrence frequency, e.g., $f_{TemRelSum}$ and f_{LLRSum}. Two particularly interesting observations from Table 11 are that: (1) word2vec features are ranked very high, suggesting that the developed semantic relatedness among term pairs inside a query does indeed help to generate appropriate queries; (2) the majority of important features use maximization rather than summation of values.

To verify the effectiveness of features deemed to be important for QAC, we create a model called L2R-TOP that extends L2R-UP with the features in Table 11 (on the corresponding datasets). The results in terms of MRR scores are listed in Table 8, column 7. We see that (1) compared to features selected from a sole source into L2R-UP, i.e., semantic relatedness or homologous queries, the important features according to Table 11 boost the performance; L2R-TOP receives higher

Table 11
Ten most important features by χ^2 test on the AOL and MSN datasets; *PI*(q) and *SQ*(q) are placeholders for pseudo-identical queries and super queries of query q, respectively.

Rank	AOL	MSN
1	$f_{W2vLogMax}$	$f_{W2vLogMax}$
2	$f_{TemRelSum}$	$fre(PI(q), 2)_{tre\_max}$
3	$fre(PI(q), 2)_{tre\_max}$	$f_{W2vGooMax}$
4	$f_{W2vGooMax}$	$fre(PI(q), 2)_{obs\_max}$
5	$fre(PI(q), 4)_{obs\_max}$	$f_{TemRelSum}$
6	$f_{TemRelMax}$	$fre(PI(q), 4)_{tre\_max}$
7	f_{cof_sum}	$fre(PI(q), 1)_{tre\_max}$
8	$fre(PI(q), 2)_{obs\_max}$	$f_{W2vLogSum}$
9	$f_{W2vLogSum}$	f_{LLRSum}
10	$fre(SQ(q), 2)_{tre\_sum}$	$fre(SQ(q), 4)_{tre\_max}$

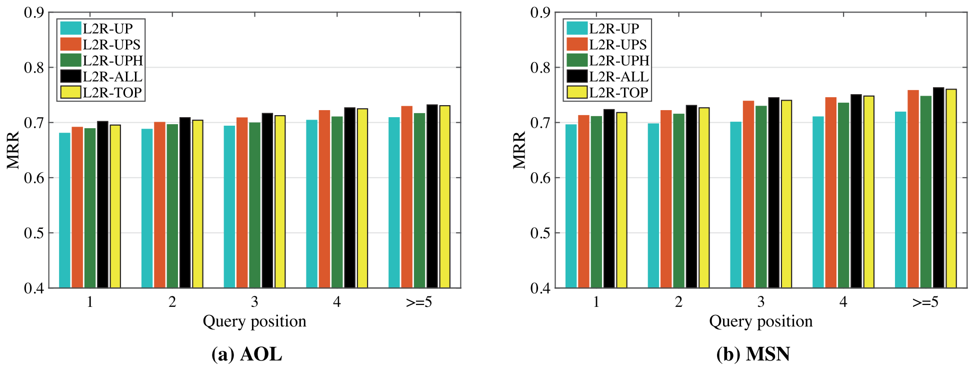


Fig. 7. Performance of L2R-based QAC models in terms of MRR at various query positions, tested on the AOL and MSN datasets, respectively.

MRR scores than L2R-UPS and L2R-UPH; (2) L2R-ALL invariably performs the best among all models, suggesting that L2R-based models not only learn from the important features, but also from the less important ones. Overall, L2R-TOP produces competitive results, implying that its 63 features (the ten most important plus 53 from L2R-UP) are highly informative for producing high quality QAC rankings.

5.6. Impact of query position

Previous work mainly focuses on the last query in a session for QAC evaluation (Cai et al., 2014b; Jiang et al., 2014). Instead, we implement tests on all queries in a session, which helps us to examine the performance of our L2R-based QAC models at various query positions in a search session. We plot the results in terms of MRR in Fig. 7 for all prefixes at each specific query position in session, tested on the AOL and MSN datasets, respectively.

We can see from Fig. 7 that: (1) for all L2R-based QAC models, the QAC performance in terms of MRR is improved when the user continues querying in a session because the MRR scores are increased as the query position changes from the beginning to the end of a session; (2) among these models, the performance of L2R-UP seems to be less sensitive to the query position than other models, especially on the MSN dataset, as the MRR scores of L2R-UP are relatively stable; (3) generally, the L2R-ALL model invariably performs best among these models at each specific query position. These findings could be due to: (1) at the end of a search session, the information of user behaviors makes more sense for learning than that at the beginning of a search session, which helps boost the QAC ranking performance; (2) semantic features are more reliably extracted at the end of a session rather than at the beginning, especially for those depending on the search context, e.g., $f_{TemRelSum}$ in (9) and f_{cof_sum} in Section 3.4.1, which are important to those models, e.g., L2R-UPS, L2R-ALL and L2R-TOP.

6. Conclusion

In this paper we follow a supervised learning to rank approach to address the problem of ranking query auto completion (QAC) candidates. We develop new features of homologous queries (i.e., those with the same terms but different orders and those extending the initial query) and semantic relatedness of terms inside a query and of pairs of terms from a query candidate and from earlier queries in the same session. We have verified the effectiveness of our models on two public datasets, showing significant improvements over state-of-the-art QAC baselines. Our analysis reveals that features of semantic relatedness and homologous queries are important and do indeed help boost QAC performance.

As to future work, we will have a closer look at the top N candidates returned by popularity based candidate ranking (MPC) for $N > 10$: how much can we gain from good candidates that were ranked low by MPC? Additionally, we want to study efficiency aspects of our approaches: parallel processing is likely to boost the efficiency of our models on feature extraction, and the addition of more, potentially expensive ways of generating homologous queries or semantic features could produce better QAC rankings. Finally, we aim to apply our approach to larger datasets than we considered in this paper, especially datasets that cover longer periods of time than AOL and MSN, as we believe that QAC can benefit from periodicity-based features.

Acknowledgments

This research was partially supported by the Innovation Foundation of NUDT for Postgraduate under No. B130503, Amsterdam Data Science, the Dutch national program COMMIT, Elsevier, the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement nr 312827 (VOX-Pol), the ESF Research Network Program ELIAS, the HPC Fund, the Royal Dutch Academy of Sciences (KNAW) under the Elite Network Shifts project, the Microsoft Research Ph.D. program, the Netherlands eScience Center under project number 027.012.105, the Netherlands Institute for Sound and Vision, the Netherlands Organisation for Scientific Research (NWO) under project nrs. 727.011.005, 612.001.116, HOR-11-10, 640.006.013, 612.066.930, CI-14-25, SH-322-15, the Yahoo! Faculty Research and Engagement Program, and Yandex. All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

References

Agichtein, E., Castillo, C., Donato, D., Gionis, A., & Mishne, G. (2008). Finding high-quality content in social media. In *WSDM '08* (pp. 183–194).
Bar-Yossef, Z., & Kraus, N. (2011). Context-sensitive query auto-completion. In *WWW '11* (pp. 107–116).
Burges, C. J., Svore, K. M., Bennett, P. N., Pastusiak, A., & Wu, Q. (2011). Learning to rank using an ensemble of lambda-gradient models. *J. Mach. Learn. Res., 14*, 25–35.
Cai, F., Liang, S., & de Rijke, M. (2014a). Personalized document re-ranking based on bayesian probabilistic matrix factorization. In *Sigir '14* (pp. 835–838).
Cai, F., Liang, S., & de Rijke, M. (2014b). Time-sensitive personalized query auto-completion. In *CIKM '14* (pp. 1599–1608).
Cao, H., Jiang, D., Pei, J., He, Q., Liao, Z., Chen, E., & Li, H. (2008). Context-aware query suggestion by mining click-through and session data. In *KDD '08* (pp. 875–883).
Chatfield, C. (2004). *The Analysis of Time Series: An Introduction*. New York: Chapman and Hall.
Chien, S., & Immorlica, N. (2005). Semantic similarity between search engine queries using temporal correlation. In *WWW '05* (pp. 2–11).
Chuklin, A., Markov, I., & de Rijke, M. (2015). Click Models for Web Search. *Synthesis Lectures on Information Concepts, Retrieval, and Services*. Morgan & Claypool Publishers.
Craswell, N., Jones, R., Dupret, G., & Viegas, E. (Eds.) (2009). *Proc. 2009 workshop on web search click data*. ACM.
Duan, H., & Hsu, B.-J. P. (2011). Online spelling correction for query completion. In *WWW '11* (pp. 117–126).
Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM Comput. Surv., 46*(4), 44:1–44:37.
Golbandi, N. G., Katzir, L. K., Koren, Y. K., & Lempel, R. L. (2013). Expediting search trend detection via prediction of query counts. In *WSDM '13* (pp. 295–304).
Guo, J., Cheng, X., Xu, G., & Zhu, X. (2011). Intent-aware query similarity. In *CIKM '11* (pp. 259–268).
Han, L., L. Kashyap, A., Finin, T., Mayfield, J., & Weese, J. (2013). UMBC EBIQUITY-CORE: Semantic textual similarity systems. In *2nd joint conference on lexical and computational semantics* (pp. 1–9). ACL.
Hofmann, K., Mitra, B., Radlinski, F., & Shokouhi, M. (2014). An eye-tracking study of user interactions with query auto completion. In *CIKM '14* (pp. 549–558).
Jiang, J.-Y., Ke, Y.-Y., Chien, P.-Y., & Cheng, P.-J. (2014). Learning user reformulation behavior for query auto-completion. In *SIGIR '14* (pp. 445–454).
Jones, R., Rey, B., Madani, O., & Greiner, W. (2006). Generating query substitutions. In *WWW '06* (pp. 387–396).
Kulkarni, A., Teevan, J., Svore, K. M., & Dumais, S. T. (2011). Understanding temporal query dynamics. In *WSDM '11* (pp. 167–176).
Li, Y., Dong, A., Wang, H., Deng, H., Chang, Y., & Zhai, C. (2014). A two-dimensional click model for query auto-completion. In *SIGIR '14* (pp. 455–464).
Liao, Z., Jiang, D., Chen, E., Pei, J., Cao, H., & Li, H. (2011). Mining concept sequences from large-scale search logs for context-aware query suggestion. *ACM Trans. Intell. Syst. Technol., 3*(1), Article17.
Liu, T.-Y. (2003). Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval, 3*(3), 225–331.
Liu, Y., Song, R., Chen, Y., Nie, J.-Y., & Wen, J.-R. (2012). Adaptive query suggestion for difficult queries. In *SIGIR '12* (pp. 15–24).
Ma, H., Yang, H., King, I., & Lyu, M. R. (2008). Learning latent semantic relations from clickthrough data for query suggestion. In *CIKM '08* (pp. 709–718).
Mei, Q., Zhou, D., & Church, K. (2008). Query suggestion using hitting time. In *CIKM '08* (pp. 469–478).
Mikolov, T., Chen, K., Corrado, G. S., & Dean, J. (2013a). Efficient estimation of word representations in vector space. In *Proceedings of workshop at ICLR* (pp. 1–12).
Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In *NIPS 26* (pp. 3111–3119).
Mitra, B., Shokouhi, M., Radlinski, F., & Hofmann, K. (2014). On user interactions with query auto-completion. In *SIGIR '14* (pp. 1055–1058).
Ozertem, U., Chapelle, O., Donmez, P., & Velipasaoglu, E. (2012). Learning to suggest: A machine learning framework for ranking query suggestions. In *SIGIR '12* (pp. 25–34).
Pass, G., Chowdhury, A., & Torgeson, C. (2006). A picture of search. In *InfoScale '06* (pp. 1–7).
Santos, R. L. T., Macdonald, C., & Ounis, I. (2013). Learning to rank query suggestions for adhoc and diversity search. *Inf. Retr., 16*, 429–451.
Sengstock, C., & Gertz, M. (2011). Conquer: A system for efficient context-aware query suggestions. In *WWW '11* (pp. 265–268).
Shokouhi, M. (2013). Learning to personalize query auto-completion. In *SIGIR '13* (pp. 103–112).
Shokouhi, M., & Radinsky, K. (2012). Time-sensitive query auto-completion. In *SIGIR '12* (pp. 601–610).
Whiting, S., & Jose, J. M. (2014). Recent and robust query auto-completion. In *WWW '14* (pp. 971–982).