Prefix-Adaptive and Time-Sensitive Personalized Query Auto Completion

Fei Cai, Shangsong Liang, and Maarten de Rijke

Abstract—Query auto completion (QAC) methods recommend queries to search engine users when they start entering a query. Current QAC methods mostly rank query completions based on their past popularity, i.e., on the number of times they have previously been submitted as a query. However, query popularity changes over time and may vary drastically across users. Accordingly, the ranking of query completions should be adjusted. Previous time-sensitive and user-specific QAC methods have been developed separately, yielding significant improvements over methods that are neither time-sensitive nor personalized. We propose a hybrid QAC method that is both time-sensitive and personalized. We extend it to handle long-tail prefixes, which we achieve by assigning optimal weights to the contribution from time-sensitivity and personalization. Using real-world search log datasets, we return top N query suggestions ranked by predicted popularity as estimated from popularity trends and cyclic popularity behavior; we rerank them by integrating similarities to a user's previous queries (both in the current session and in previous sessions). Our method outperforms state-of-the-art time-sensitive QAC baselines, achieving total improvements of between 3 and 7 percent in terms of mean reciprocal rank (MRR). After optimizing the weights, our extended model achieves MRR improvements of between 4 and 8 percent.

Index Terms—Query auto completion, personalization, time-sensitive, long-tail, web search

1 Introduction

Query auto completion (QAC) helps users formulate a query and improve the search quality. Currently, common search engines and popular online properties, e.g., online shopping sites and email services, provide a QAC service. As a user enters a prefix in the search box, matching completions may appear below the search box as a drop-down menu with the typed characters highlighted in each completion. These matching completions can be further reranked according to other metrics, e.g., the price of products in an online store [2].

In pre-computed auto completion systems, the list of matching completions for a prefix is generated in advance and stored in an efficient data structure for fast lookups. When needed, as shown in Fig. 1, continued typing characters can lead to dynamic refinements of the completions by exact prefix matching until an appropriate completion is found. Where offered, the facility is heavily used and highly influential on search results [3], [4].

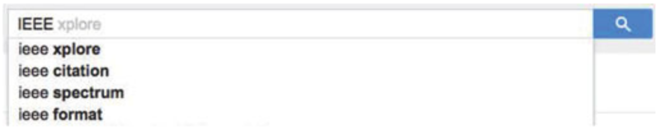
A common and useful approach in previous work on QAC is to extract past queries extending a prefix from query logs, and then rank them by their popularity [3], [4], i.e., by the number of times they have been submitted, which assumes that current or future query popularity is the same as past query popularity. This approach fails to take strong clues from time, trend and user-specific context into account while such information often influences the queries most likely to be entered. As illustrated in Fig. 2, personalized QAC may inject the most popular completions from a user as query completions for that particular user; see Fig. 2a (not personalized) and Fig. 2b (personalized). From Figs. 2c and 2d according to Google Trends,[1] we see that query popularity strongly depends on time (a clear burst for the query "*MH17*" around 18 July, 2014) and that it is subject to cyclic phenomena (yearly for the query "*New year*" and weekly for the query "*Movie*"), which can be explored to forecast future query popularity. This motivates a QAC approach that takes both the temporal aspect and the personal context into account.

In addition, some user inputs are easier to complete than others, depending on the "popularity" of the prefix, measured here by the number of returned query completions. For instance, as shown in Fig. 3, a search session contains three queries from the well-known AOL query log [5]. For the sake of the example, let us assume that we have not yet seen the last query (row 4 in Fig. 3a, query "*jsonline*"), and that there are three lists in Fig. 3b of query completions with the initial prefix "*js*," "*jso*" and "*json*," respectively, of this query "*jsonline*," for which we want to recommend completions. A regular baseline based on the most popular query is applied to return these completions [3]. More completions are returned for the first prefix (row 2 in Fig. 3b) than for the second and third prefixes (rows 3 and 4, respectively, in Fig. 3b). We treat a prefix as a long-tail prefix or a normal prefix, depending on the number of returned query completions rather than the number of characters in the prefix. The technical challenges for dealing with long-tail prefixes are:

• F. Cai is with the Science and Technology on Information Systems Engineering Laboratory, National University of Defense Technology, Changsha, Hunan 410073, China, and the Informatics Institute, University of Amsterdam, Amsterdam 1012, WX, The Netherlands. E-mail: f.cai@uva.nl.
• S. Liang is with the Department of Computer Science, University College London, London, WC1E 6BT, United Kingdom. E-mail: shangsong.liang@ucl.ac.uk.
• M. de Rijke is with the Informatics Institute, University of Amsterdam, Amsterdam 1012, WX, The Netherlands. E-mail: derijke@uva.nl.

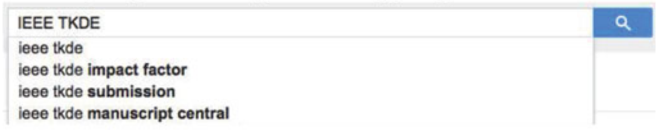
Manuscript received 24 Mar. 2015; revised 10 May 2016; accepted 10 May 2016. Date of publication 12 May 2016; date of current version 3 Aug. 2016. Recommended for acceptance by J. Pei.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TKDE.2016.2568179

1. http://www.google.com/trends



IEEE xplore
ieee xplore
ieee citation
ieee spectrum
ieee format

(a) **Query auto completion of typed prefix *"IEEE␣"*.**



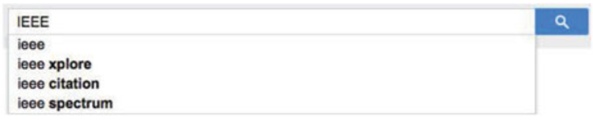
IEEE TKDE
ieee tkde
ieee tkde impact factor
ieee tkde submission
ieee tkde manuscript central

(b) **Query auto completion of typed prefix *"IEEE TKDE"*.**

Fig. 1. (Top) Query auto completion for the prefix *"IEEE␣"*, where ␣ indicates that a space follows after *"IEEE"*. (Bottom) The refined completions after continuing to type *TKDE* after *"IEEE␣"*. The snapshot was taken on Thursday, February 12, 2015.

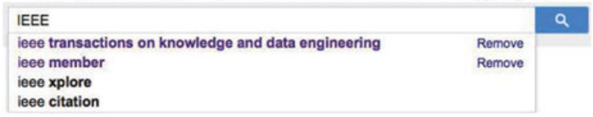
(1) only a limited number of query completions are matched; and (2) they are not easily distinguishable from others by their popularity, making it difficult to infer a user's search intent. However, for normal prefixes, the baseline popularity-based strategy generally works well. Thus, other resources, such as the search context, rather than query popularity could help address these challenges. This motivates us to work with a modified QAC model to adaptively deal with long-tail prefixes. In such a modified QAC model, the parameters controlling the contribution of search popularity and of search context for ranking query completions are optimized by a regression model.

Before we start, we differentiate query auto completion from query suggestion in Table 1: for QAC we tend to follow a strict matching policy while for query suggestions we do not. Accordingly, in QAC we rank a limited number of query completions and allow the user to complete their query before it has been fully entered without modifying their previously entered input. In contrast, query suggestions may include query suggestions that are semantically related to the user's input but may suffer a problem of ranking a large amount of query suggestions.



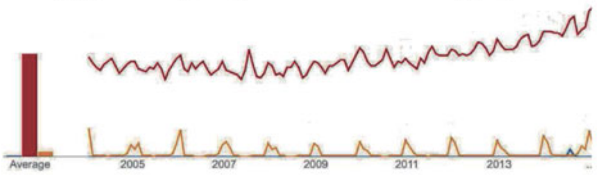
IEEE
ieee
ieee xplore
ieee citation
ieee spectrum

(a) **QAC of typed prefix *"IEEE"* without logging in.**

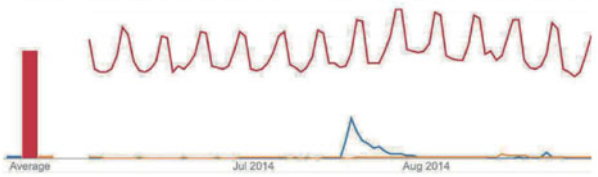


IEEE
ieee transactions on knowledge and data engineering Remove
ieee member Remove
ieee xplore
ieee citation

(b) **QAC of typed prefix *"IEEE"* after logging in.**

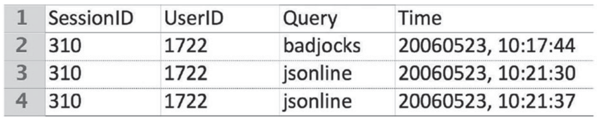


(c) **Relative query popularity in the past few years (2004–present).**



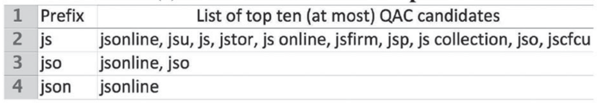
(d) **Relative query popularity within 3 months (June, 2014–August, 2014).**

Fig. 2. (Top) Query auto completions of the typed prefix "*IEEE*" under different settings. (Bottom) Relative query popularity for different queries over time. Queries: "*MH17*" in blue, "*Movie*" in red, and "*New year*" in yellow. The snapshot was taken on Thursday, February 12, 2015.



1	SessionID	UserID	Query	Time
2	310	1722	badjocks	20060523, 10:17:44
3	310	1722	jsonline	20060523, 10:21:30
4	310	1722	jsonline	20060523, 10:21:37

(a) **An AOL session example.**



1	Prefix	List of top ten (at most) QAC candidates
2	js	jsonline, jsu, js, jstor, js online, jsfirm, jsp, js collection, jso, jscfcu
3	jso	jsonline, jso
4	json	jsonline

(b) **Lists of top ten (at most) completions after typing corresponding prefix.**

Fig. 3. An AOL session example (Top) and lists of top 10 (at most) completions returned based on frequency after typing the corresponding prefix (Bottom).

Given a threshold N, we define a prefix p to be a *long-tail* prefix if the number of returned query completions is smaller than N. If we set $N = 10$, the prefixes "*jso*" and "*json*" in Fig. 3b are long-tail prefixes while the prefix "*js*" is not.

The QAC model that we propose first returns the top N query completions by predicted popularity based on their cyclic behavior and recent trend, and then reranks these completions by user-specific context to output a final query auto completion list. We show that, when compared against a state-of-the-art time-sensitive QAC baseline [6], the predicted popularity values produced by our time-sensitive approach are closer approximations to what will be observed later in the logs, and are more effective for QAC after integrating user query similarity and checking prefix popularity, with improvements in Mean Reciprocal Rank (MRR) scores between 4 and 8 percent on a web search log and on a query log from an audiovisual archive.

Our contributions in this paper can be summarized as follows:

(1) We address the challenge of query auto completion in a novel way by considering both time-sensitive query popularity and user-specific context.
(2) We propose a new query popularity prediction method that explores the cyclic behavior and recent trend of query popularity.
(3) We extend our hybrid QAC model to deal with long-tail prefixes by optimizing the contributions from query popularity and user-specific context.

We describe related work in Section 2. Our approach is described in Section 3, while Section 4 presents the experimental setup. We report our experimental results in Section 5 and conclude in Section 6.

TABLE 1
Query Auto Completion versus Query Suggestion

Task	Input	Output
QAC	Prefix p	Query set $Q = \{q_o : q_o \text{ starts with } p\}$
QS	Query q	Query set $Q = \{q_o : q_o \text{ is relevant to } p\}$

2 Related Work

In major web search scenarios, a common and straightforward approach to rank query completions is to use Maximum Likelihood Estimation (MLE) based on the past popularity of queries [3]. Bar-Yossef and Kraus [3] refer to this type of ranking as the Most Popular Completion (MPC) model

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

where $f(q)$ denotes the number of occurrences of query q in search log Q, and $C(p)$ is a set of query completions that start with prefix p. In essence, the MPC model assumes that the current query popularity distribution will remain the same as that previously observed, and hence completions are ranked by their past popularity in order to maximize QAC effectiveness for all users on average. As mentioned above, query popularity may change over time and the ranking of completions is user-dependent (see Fig. 2). Accordingly, query completions must be adjusted to account for time-sensitive and user-specific changes.

Next, we summarize recent QAC approaches into three parts: time-sensitive, context-aware, and interaction-based QAC.

2.1 Time-Sensitive Query Auto Completion

Time-sensitive query auto completion (TS-QAC) takes time information, such as recency [6], [7], [8] and seasonality [4], [9], into consideration for ranking query completions. It leverages time-series analysis techniques for classifying seasonal queries and forecasting their future popularity [4], [9].

Rather than ranking query completions by their previously observed popularity, Shokouhi and Radinsky [4] propose a long-term time-series modeling approach to forecast the query frequencies by applying a fixed moving time window. The forecasts obtained by such time-series modeling are substantially more reliable. Similarly, Strizhevskaya et al. [10] study actualization techniques for measuring prediction accuracy of various daily query popularity prediction models using query logs.

Another aspect of time-sensitive QAC is the problem of search trend prediction. Short-range query popularity prediction has seen little attention. Golbandi et al. [7] develop a regression model to detect bursting queries for enhancing trend detection. By analyzing query logs, they seek to accurately predict what the most trending query items on the Web are. Kulkarni et al. [11] classify queries into different categories based on the changes in popularity over time and show that monitoring query popularity can reveal strong signals for detecting trends in query intent.

Recently, considering both recent trends and past query popularity, Whiting and Jose [6] have proposed several practical approaches to ranking query completions, such as outputting query popularity evidence from a sliding window of the past 2 to 28 days or the query popularity distribution in a recent query chunk observed with a given prefix, as well as predicting query popularity based on recently observed trends. Cai and de Rijke [12] propose a learning-based QAC approach where features derived from similar queries and semantically related terms are taken into account.

Our TS-QAC approach (detailed below) differs from previous work as we consider both periodicity and recent trends in query frequency. Additionally, none of the publications listed so far caters for individual users, returning the same QAC list of typed prefixes for all users. We return a personalized QAC list to boost QAC effectiveness based on a time-sensitive ranking of query completions output by forecasted query popularity, which will specifically benefit revisiting search tasks.

2.2 Context-Aware Query Auto Completion

In most work mentioned so far, query completions are computed globally and for a given prefix: all users are presented with the same list of completions. But exploiting the user's personal context has led to increases in QAC effectiveness [3], [13], [14], [15], [16].

Bar-Yossef and Kraus [3] treat the user's recent queries as context and exploit users with shared search activity, considering the similarity of query completions with this context for ranking. Their hybrid model computes the final score of each completion by linearly combining the MPC score and a context-similarity score. Our approach to personalized QAC differs in the definition of context-similarity and in the way we measure it. Shokouhi [14] exploits profiles to extract user-based features to model the likelihood that a user will issue certain queries, and explores the effectiveness of considering a user's age, gender, location and longer search history in ranking query completions, to thereby personalize the ranking of query completions.

Guo et al. [17] propose a two-step approach, where the user's session context is matched against pre-generated topic models for ranking query completions. Similarly, Cao et al. [18] and Liao et al. [15] first cluster queries in the click graph into a smaller set of virtual concepts. They match the users' context captured based on their recent queries against these clusters for ranking query completions. Weber and Castillo [19] focus on showing differences in query likelihood across demographics. They predict the second term in a query based on an unsupervised probabilistic model. Building on temporal intuitions, Sengstock and Gertz [20] consider query completions that depend on the time of day, i.e., taking the search time as a user-specific context. Arias et al. [21] propose a QAC algorithm for mobile search; their completions are thesaurus-based concepts whose relatedness to the user's context is fixed and pre-determined by a rule-based mechanism.

Bhatia et al. [22] mine frequently occurring phrases and n-grams from indexed documents for generating and ranking query completions for partial queries in the absence of search logs. Fan et al. [23] propose a generative model that incorporates the topical coherence of terms based on Latent Dirichlet Allocation (LDA) for ranking query completions. Closely related to QAC, Bickel et al. [24] learn a linearly interpolated n-gram model for sentence completion based on lexical statistics of text collections. Grabski and Scheffer [25] deploy an index-based retrieval algorithm and a cluster-based approach for their sentence-completion task.

Our personalized QAC approach works in a different way; it scores query completions according to their

TABLE 2
Description of Query Popularity Prediction Methods and QAC Approaches

Approach	Description	Source
P_k	Predicting a query's future popularity by aggregating the frequency in past k days	[4]
MPC-ALL	Ranking query completions according to their past popularity on the whole log	[3]
MPC-R-TW	Ranking query completions according to their past popularity within a fixed time window	[6]
O-MPC-R	Ranking query completions according to their past popularity within an optimal time window	[6]
P_{trend}	Predicting a query's future popularity by its recent trend as described in (5)	This paper
λ^*-TS-QAC	A TS-QAC model (Algorithm 1) with an optimal parameter λ^* used in (2), achieved by (8)	This paper
Personalized QAC	Ranking query completions according to their similarity to previous queries, following (9)	This paper
λ-H-QAC	A hybrid QAC model (Algorithm 2) integrating λ-TS-QAC as (14) and personalized QAC as (15)	This paper
λ^*-H-QAC	A hybrid QAC model (Algorithm 2) integrating λ^*-TS-QAC by (8) and personalized QAC as (15)	This paper
G-QAC	A hybrid QAC model integrating MPC-ALL and n-gram based personalized QAC as (13)	This paper
λ^*-H$_G$-QAC	A hybrid QAC model integrating λ^*-TS-QAC and n-gram based personalized QAC as (13)	This paper
λ^*-H$'$-QAC	A modified λ^*-H-QAC model with long-tail prefix detection added	This paper

similarity both to frequent queries issued by the same user and to queries in the current search session.

2.3 Interaction-Based Query Auto Completion

Recently, user interactions have begun to play a more prominent role in algorithms for QAC [26], [27], [28]. Jiang et al. [27] 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. With fine-grained user interaction information, Li et al. [29] observe a horizontal skipping bias and vertical position bias exists in the QAC process, and propose a two-dimensional click model for modeling the QAC process. Mitra et al. [28] investigate user interaction patterns with QAC 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 interesting findings provide valuable insights for understanding user engagement with QAC.

In addition, Li et al. [30] pay attention to users' sequential interactions with a QAC engine in and across QAC sessions, rather than users' interactions at each keystroke of each QAC session. Based on an in-depth analysis of a high-resolution query log, they propose a probabilistic model that addresses the QAC task by capturing the relationship between users' sequential behaviors at different keystrokes. Zhang et al. [31] study implicit negative feedback from a user's interactions with a search engine, and propose an adaptive model that adapts query auto completion to feedback about skipped query completions. This model is based on the assumption that top ranked but skipped query completions are not likely to be submitted and encodes the strength of the implicit negative feedback to a query completion q_c from a user u with personalization. We refer the reader to [32] for a recent survey of the state-of-the-art in query auto completion.

Interaction-based QAC models need a special device to record explicit interactions as feedback to infer users' search intent. This setup differs from our QAC scheme, where user's previous queries both in the current search session and their former search sessions are used to model their specific interest. Our idea originates from the combination of time-sensitivity and personalization used in current QAC algorithms. Our approach to personalized QAC stands to gain from query repetitions in user search behavior. Additionally, to the best of our knowledge, there is no published work on QAC that specifically deals with long-tail prefixes.

3 Approach

In this section we describe our time-sensitive personalized query auto completion approach, a hybrid model that not only inherits the merits of time-sensitive query auto completion but also considers a user's personal context. Table 2 provides an overview of various QAC approaches; the baselines (rows 1–3) are described in the literature; we detail our models (rows 4–10) in three steps: time-sensitive QAC, personalized QAC, and hybrid QAC.

3.1 Periodicity and Trend Based QAC

We propose a time-sensitive QAC method that ranks query completions by predicted query popularity (i.e., its frequency) based on its periodicity and recent trend to detect both cyclicly and instantly frequent queries. TS-QAC not only inherits the merits of time-series analysis on long-term observations of query popularity, but also considers the recent variation of query counts. Specifically, we predict a query q's next-day popularity $\tilde{y}_{t_0+1}(q,\lambda)$ by both its recent trend and periodicity with a free parameter λ $(0 \leq \lambda \leq 1)$ controlling each contribution

$$\tilde{y}_{t_0+1}(q,\lambda) = \lambda \cdot \hat{y}_{t_0+1}(q)_{trend} + (1-\lambda) \cdot \bar{y}_{t_0+1}(q)_{peri}, \quad (2)$$

where $\lambda = 1$ for aperiodic queries and $0 \leq \lambda < 1$ for periodic queries. The term $\hat{y}_{t_0+1}(q)_{trend}$ is estimated via a linear aggregation of predictions from recent N_{days} observations

$$\hat{y}_{t_0+1}(q)_{trend} = \textstyle\sum_{i=1}^{N_{days}} norm(\omega_i) \cdot \hat{y}_{t_0+1}(q,i)_{trend}, \quad (3)$$

where $norm(\omega_i)$ normalizes the contributions from each day to ensure $\sum_i \omega_i = 1$. We introduce a temporal decay function to output the weight before normalizing as $\omega_i = f^{TD(i)-1}$, where f is a decay factor and $TD(i)$ refers to the interval from day i to the future day t_0+1. We identify the highest prediction accuracy parameter N_{days} for each query based on its past observations in the whole log using a multiple linear regression model, following [6]. The prediction $\hat{y}_{t_0+1}(q,i)_{trend}$ from each day i ($i = 1,\dots,N_{days}$) is derived from the first order derivative of q's daily count $C(q,t)$ as

$$\hat{y}_{t_0+1}(q,i)_{trend} = y_{t_0-TD(i)}(q,i) + \int_{t_0-TD(i)}^{t_0+1} \frac{\partial C(q,t)}{\partial t}\mathrm{d}t, \tag{4}$$

where $y_{t_0-TD(i)}(q,i)$ is the observed query count of q at day i.

The periodicity term $\bar{y}_{t_0+1}(q)_{peri}$ in (2) is smoothed by simply averaging the recent M observations y_{t_p} at preceding time points $t_p = t_0 + 1 - 1\cdot T_q, \dots, t_0 + 1 - M\cdot T_q$ in the log

$$\hat{y}_{t_0+1}(q)_{peri} = \tfrac{1}{M}\textstyle\sum_{m=1}^{M} y_{t_0+1-m\cdot T_q}(q), \tag{5}$$

where T_q denotes q's periodicity. For detecting cyclic aspects of query q's frequency, we use autocorrelation coefficients [33], which measure the correlation between N_s successive count observations $C(q,t)$ at different times $t = 1,2,\dots,N_s$ in the query log. The correlation is computed between a time series and the same series lagged by i time units as

$$r_i = \frac{\sum_{t=1}^{N_s-i}(C(q,t)-\bar{x}_1)(C(q,t+i)-\bar{x}_2)}{\left(\sum_{t=1}^{N_s-i}(C(q,t)-\bar{x}_1)^2\right)^{\frac{1}{2}}\left(\sum_{t=i+1}^{N_s}(C(q,t+i)-\bar{x}_2)^2\right)^{\frac{1}{2}}}, \tag{6}$$

where $\bar{x}_1$ (or $\bar{x}_2$) is the mean of the first (or last) $N_s - i$ observations. For N_s reasonably large, the denominator in (6) can be simplified by approximation. First, the difference between the 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. Accordingly, r_i can be approximated by

$$r_i \approx \frac{\sum_{t=1}^{N_s-i}(C(q,t)-\bar{x})(C(q,t+i)-\bar{x})}{\sum_{t=1}^{N_s}(C(q,t)-\bar{x})^2}, \tag{7}$$

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

In addition, we choose an optimal λ^* by minimizing the metric *Mean Absolute Error* (MAE) (described in Section 4.3) as

$$\lambda^* = \arg\min_{0\le\lambda\le1} \frac{1}{|Q|}\cdot\frac{1}{|L_v|}\sum_{q\in Q}\sum_{s=1}^{|L_v|}|\tilde{y}_s(q,\lambda) - y_s(q)|, \tag{8}$$

where $\tilde{y}_s(q,\lambda)$ and $y_s(q)$ are the predicted and the true query counts at day s in the validation period (L_v days), respectively.

Algorithm 1 details the major steps of our time-sensitive QAC method. We write λ-TS-QAC for the version with a fixed λ and λ^*-TS-QAC for the version where an optimal λ^* is chosen. We fix the optimal number of days for predicting the popularity trend by minimizing the MAE in row 10 in Algorithm 1.

Algorithm 1. Time-Sensitive Query Auto Completion

```
Input: All queries: $Q$; Length of training and validation days: $L_t$ and $L_v$; $t_0$; Number of returned completions: $N$;
Output: Predictions: $\bar{Q} = \{\bar{y}_{t_0+1}(q): q \in Q\}$;
        Top $N$ completions of each prefix of all queries;
1:  for each $q \in Q$ do
2:    $T_q \leftarrow autocor(Count(q))$;
3:    for $i = 1,\dots,L_t$ do
4:      for $j = 1,\dots,L_v$ do
5:        $\hat{y}_{t_0+1}(q)_{trend}[j] \leftarrow Regression(Count(q)[1:i])$;
6:        $AbsoluteError[j] \leftarrow \hat{y}_{t_0+1}(q)_{trend}[j] - y_{t_0+1}(q)|$;
7:      end for
8:      $MAE(i) \leftarrow mean(AbsoluteError)$;
9:    end for
10:   $N_{days} \leftarrow \arg\min_{1\le i\le L_t} MAE(i)$;
11:   Update $\hat{y}_{t_0+1}(q)_{trend}$ with optimal $N_{days}$ and Compute
      $\bar{y}_{t_0+1}(q)_{peri}$;
12: end for
13: Find an optimal $\lambda^*$ by (8);
14: $\lambda \leftarrow \lambda^*$;
15: for each $q \in Q$ do
16:   $\tilde{y}_{t_0+1}(q,\lambda) \leftarrow \lambda \times \hat{y}_{t_0+1}(q)_{trend} + (1-\lambda) \times \bar{y}_{t_0+1}(q)_{peri}$;
17: end for
18: for each $q \in Q$ do
19:   for each prefix $p$ of $q$ do
20:     Return top $N$ completions of $p$ ranked by $\tilde{y}_{t_0+1}(q,\lambda)$;
21:   end for
22: end for
```

3.2 Personalized QAC

Next, we extend our time-sensitive QAC described in Section 3.1 with personalized QAC. After sorting the queries with typed prefix p by predicted popularity following (2), we are given a ranked list of the top N query completions. Let $\mathcal{S}(p)$ represent the set of returned top N query completions of prefix p.

Our personalized QAC works by scoring completions $q_c \in \mathcal{S}(p)$ using a combination of two similarity scores $Score(Q_s, q_c)$ and $Score(Q_u, q_c)$, where Q_s relates to the recent queries in the current search session and Q_u refers to those of the same user issued before, if available, as

$$Pscore(q_c) = \omega \cdot Score(Q_s, q_c) + (1-\omega)\cdot Score(Q_u, q_c), \tag{9}$$

where ω controls the weight of the individual components. Personalized QAC works at the session-based and user-dependent level.

To compute the required similarity scores, we first consider how to represent queries in Q_s and Q_u. A naive approach would be to represent a query by n-grams or its terms as "a bag of words." The resulting similarity measure can capture syntactic reformulations. However, the problem is that queries are short, and thus their vocabulary is too sparse to capture semantic relationships. In order to overcome this sparsity problem, we use another solution to measure similarity. We observe in our datasets (see Table 3 and Fig. 5) that users often request the same query or reformulate the query by modifying previous ones within the same session. We treat a user's preceding queries Q_s in the current session and their preceding queries Q_u in the historical log as context to personalize QAC where we measure similarity at the character level.

We represent the query $q_s \in Q_s$ and $q_c \in S(p)$ by their query terms as $\{w_{s1}, w_{s2}, \ldots, w_{sm}\}$ and $\{w_{c1}, w_{c1}, \ldots, w_{cn}\}$ and let $N(w_*, q_*)$ denote the count of w_* appearing in q_*. We estimate the similarity between q_c and Q_s as a conditional probability

$$\begin{aligned} Score(Q_s, q_c) &= p(q_c|Q_s) \\ &= \sum_{q_s \in Q_s} norm(\omega_s) \cdot p(q_c|q_s), \end{aligned} \tag{10}$$

where $norm(\omega_s)$ introduces a decay function $\omega_s = f^{TD(s)-1}$ as in (3) except that here $TD(s)$ refers to the interval between q_c and q_s, and $p(q_c|q_s)$ is calculated following [34], [35] as

$$\begin{aligned} p(q_c|q_s) &= \prod_{w_{ci} \in q_c} p(w_{ci}|q_s)^{N(w_{ci}, q_c)} \\ &= \prod_{w_{ci} \in q_c} p(w_{ci}|W(w_{ci}))^{N(w_{ci}, q_c)}, \end{aligned} \tag{11}$$

where $W(w_{ci}) = \{w \in q_s \mid w[0] = w_{ci}[0]\}$ is a set of terms in q_s sharing the same start with w_{ci}, and define

$$\begin{aligned} p(w_{ci}|W(w_{ci})) &\equiv Similarity(w_{ci}, W(w_{ci})) \\ &= \frac{1}{|W(w_{ci})|} \sum_{w_j \in W(w_{ci})} Similarity(w_{ci}, w_j) \\ &= \frac{1}{|W(w_{ci})|} \sum_{j=1}^{|W(w_{ci})|} \frac{len(common(w_{ci}, w_j))}{\min(len(w_{ci}), len(w_j))}, \end{aligned}$$

where $len(common(w_{ci}, w_j))$ is the maximal length of common string appearing in w_{ci} and w_j from the beginning.

We compute $Score(Q_u, q_c)$ in a different manner from $Score(Q_s, q_c)$ in (10) because in this setting it is desirable to consider both query count and time interval. Specifically, we output $Score(Q_u, q_c)$ as

$$Score(Q_u, q_c) = p(q_c|Q_u) = \sum_{q_u \in Q_u} norm(\omega_u) \cdot p(q_c|q_u), \tag{12}$$

where $norm(\omega_s)$ only depends on the query count—we assume that frequent queries reflect a user's personal search clues.

3.3 Hybrid QAC

We introduce a hybrid QAC model that combines time-sensitive QAC with personalized QAC. First, TS-QAC produces a list of query completions $S(p)$ of prefix p. We assign $TSscore(q_c)$ to each completion $q_c \in S(p)$ using its predicted popularity, i.e., $\tilde{y}_{t_0+1}(q_c, \lambda)$ in (2). Like [3], we then define our hybrid models as convex combinations of two scoring functions

$$Hscore(q_c) = \gamma \cdot TSscore(q_c) + (1 - \gamma) \cdot Pscore(q_c). \tag{13}$$

As $TSscore(q_c)$ and $Pscore(q_c)$ use different units and scales, they need to be standardized before being combined. We standardize $TSscore(q_c)$ (used in [3]) as

$$TSscore(q_c) \leftarrow \frac{\tilde{y}_{t_0+1}(q_c, \lambda) - \mu_T}{\sigma_T}, \tag{14}$$

where μ_T and σ_T are the mean and standard deviation of predicted popularity of queries in $S(p)$. Similarly, we use (9) to obtain

$$Pscore(q_c) \leftarrow \frac{Pscore(q_c) - \mu_P}{\sigma_P}, \tag{15}$$

where μ_P and σ_P are the mean and standard deviation of similarity scores of queries in $S(p)$. Algorithm 2 describes our hybrid QAC model, which requires a ranked list of query completions along with their predicted popularity as produced by Algorithm 1; (13) provides the overall ranking score (see Algorithm 2, row 15).

Algorithm 2. Hybrid QAC Model

```
Input: Predictions: $\bar{Q}$; user: $u$; prefix $p$; $N$; $\alpha$;
Output: Ranked list of top $N$ query completions of $p$;
 1: Produce $S(p)$ consisting of top $N$ query completions by (2);
 2: List $u$'s queries $Q_u$ and $Q_s$;
 3: for each $q_c \in S(p)$ do
 4:     Compute $TSscore(q_c)$ based on (14);
 5:     for each $q_s \in Q_s$ do
 6:         $p(q_c|q_s) = Similarity(q_c, q_s)$;
 7:     end for
 8:     Compute $Score(Q_s, q_c)$ based on (11);
 9:     for each $q_u \in Q_u$ do
10:         $p(q_c|q_u) = Similarity(q_c, q_u)$;
11:     end for
12:     Compute $Score(Q_u, q_c)$ based on (12);
13:     Compute $Pscore(q_c)$ based on (9) and (15);
14: end for
15: Re-rank $S(p)$ by $HQscore(q_c)$ based on (13);
16: Return a reranked list of $S(p)$;
```

We write λ-H-QAC to refer to the hybrid combination of λ-TS-QAC (used at row 4 in Algorithm 2) and the personalization approach described in the previous section; we write λ^*-H-QAC for the variant where λ has been optimized according to (8).

For comparison, we also introduce other combined QAC models that consider the query popularity and personalization. For instance, the G-QAC model derives the $TSscore(q_c)$ score in (13) by MPC-ALL and $Pscore(q_c)$ by using an n-gram representation similarity. Similarly, the λ^*-H$_G$-QAC model generates $TSscore(q_c)$ by our λ^*-TS-QAC (in Section 3.1) and $Pscore(q_c)$ by the same n-gram representation similarity.

3.4 Modified λ^*-H-QAC (λ^*-H′-QAC)

The λ^*-H-QAC model assigns a fixed weight γ to $TSscore(q_c)$ and $1 - \gamma$ to $Pscore(q_c)$ in (13) when calculating the final ranking score for the completion q_c. All prefixes are handled equally when we score their completions associated with them. However, we observe that some typed prefixes are easier to complete than others. This depends on the prefix popularity discussed in Section 1. This observation motivates an extended ranking model, named λ^*-H′-QAC. Rather than using a fixed weight γ for all prefixes, we assign an optimal weight $\bar{\gamma}$ to long-tail prefixes after checking their prefix popularity.

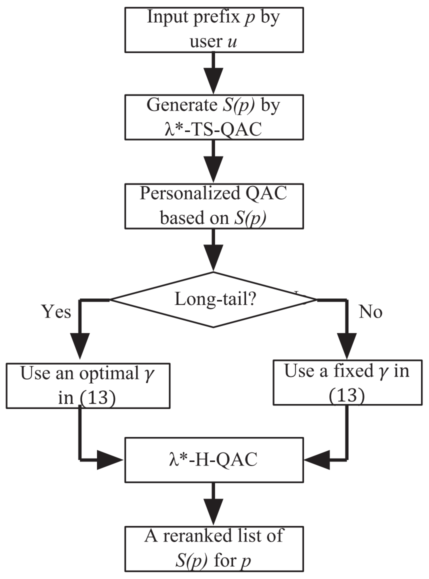

Input prefix p by user u
Generate $S(p)$ by λ*-TS-QAC
Personalized QAC based on $S(p)$
Long-tail?
Yes
No
Use an optimal γ in (13)
Use a fixed γ in (13)
λ*-H-QAC
A reranked list of $S(p)$ for p

Fig. 4. Main steps of the λ^*-H$'$-QAC model.

More specifically, to derive $\bar{\gamma}$, we first partition the prefixes in the training data into two buckets according to the number of query completions returned by TS-QAC: long-tail and the other, i.e., $\bar{P}_1$ and $\bar{P}_2$ in Algorithm 3; then we directly perform a linear regression on $\bar{P}_1$ to generate an optimal weight $\bar{\gamma}$ for long-tail prefixes. Finally, λ^*-H$'$-QAC coincides with λ^*-H-QAC for normal prefixes with a fixed weight γ in (13) but unlike λ^*-H-QAC, for long-tail prefixes it uses the optimal weight $\bar{\gamma}$. The details are described in Algorithms 3 and 4. We visualize the main steps of our model λ^*-H$'$-QAC in Fig. 4, where most steps are done offline, including generating $S(p)$ by λ^*-TS-QAC and finding the optimal $\bar{\gamma}$ for long-tail prefixes, etc.

Algorithm 3. Optimization on Long-Tail Prefixes (OLP)

```
Input: Predictions: $\bar{Q}$; prefix set $\bar{P}$; $N$; threshold $Num$;
Output: Subset $\bar{P}_1$ for long-tail prefixes and $\bar{P}_2$ for others;
 1: $\bar{P}_1 = \bar{P}_2 = \{\}$;
 2: for each prefix $p \in \bar{P}$ do
 3:   Produce $S(p)$ consisting of top $N$ query completions by (2);
 4:   if $N < Num$ then
 5:     $\bar{P}_1 = \bar{P}_1 \cup \{p\}$;
 6:   else
 7:     $\bar{P}_2 = \bar{P}_2 \cup \{p\}$;
 8:   end if
 9:   for each $q_c \in S(p)$ do
10:     Compute $TSscore(q_c)$ based on (14);
11:     Compute $Pscore(q_c)$ based on (9) and (15);
12:   end for
13: end for
14: Apply linear regression on $\bar{P}_1$, producing an optimal weight
    $\bar{\gamma}$ in (13);
15: Return $\bar{\gamma}$, $\bar{P}_1$ and $\bar{P}_2$;
```

4 Experimental Setup

Below, Section 4.1 lists the research questions that guide our experiments; Section 4.2 describes the datasets and lists some interesting observations; Section 4.3 provides details about our evaluation metrics and baselines; we detail our settings and parameters in Section 4.4.

Algorithm 4. λ^*-H$'$-QAC

```
Input: Predictions: $\bar{Q}$; prefix set $\bar{P}$; $N$; $\bar{\gamma}$;
Output: Ranked list of top $N$ query completions for each $p \in \bar{P}$;
1: for each prefix $p \in \bar{P}$ do
2:   List $S(p)$;
3:   if $p \in \bar{P}_1$ then
4:     Perform $\lambda^*$-H-QAC for $p$ with $\bar{\gamma}$ instead of $\gamma$ in (13);
5:   else
6:     Perform $\lambda^*$-H-QAC for $p$ with a fixed $\gamma$ in (13);
7:   end if
8: end for
9: Return a reranked list of $S(p)$;
```

4.1 Research Questions

The research questions guiding the remainder of the paper are:

RQ1 As a sanity check, what is the accuracy of query popularity prediction generated by various models? (See Sections 5.1 and 5.2.)

RQ2 How do our time-sensitive QAC models (λ-TS-QAC and λ^*-TS-QAC) compare against state-of-the-art time-sensitive QAC baselines? (See Section 5.3.)

RQ3 Does λ^*-H-QAC outperform time-sensitive QAC methods (O-MPC-R and λ^*-TS-QAC)? (See Section 5.4.)

RQ4 How does λ^*-H-QAC compare against a personalized QAC method using n-gram based query similarity (G-QAC)? (See Section 5.5.)

RQ5 Which ingredient makes a bigger contribution to effectively ranking query completions, predicted popularity or query similarity? (See Section 5.6.)

RQ6 How does λ^*-H$_G$-QAC compare against λ^*-H-QAC? (See Section 5.7.)

RQ7 How does λ^*-H$'$-QAC compare against λ^*-H-QAC on long-tail prefixes? And on all prefixes? (See Sections 5.8 and 5.9.)

4.2 Datasets

We use two query log[2] in our experiments: AOL [5] and one made available by the Netherlands Institute for Sound and Vision,[3] to which we will refer as "SnV" [37]. The AOL log is publicly available and sufficiently large to guarantee statistical significance and SnV is one of the largest audiovisual archives in Europe. The AOL queries were sampled between March 1, 2006 and May 31, 2006. In total there are 16,946,938 queries submitted by 657,426 unique users, while the SnV logs were recorded for one year between January 1, 2013 and December 31, 2013 using an in-house system tailored to the archive's online interface. For consistency, we partitioned each log into two parts: a training set consisting

2. Other popular query logs used in recent research, such as the MSN log [36] and the Sogou log (http://www.sogou.com/labs) were not used. The former lacks user IDs and the latter is too small.

3. http://www.beeldengeluid.nl

TABLE 3
Statistics of the Processed AOL and SnV Datasets

	AOL		SnV	
Variables	Training	Testing	Training	Testing
#Qs	6,904,655	3,609,617	291,392	154,770
#Unique Qs	456,010	456,010	86,049	86,049
#Ss	5,091,706	2,201,990	176,893	102,496
#Unique Us	466,241	314,153	1051	804
Qs/Session	1.36	1.63	1.65	1.51
Qs/User	14.81	11.49	277.25	192.50

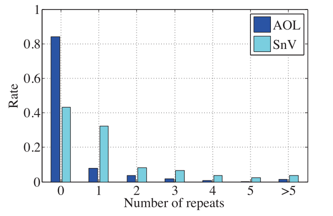
Queries: Qs, Sessions: Ss, Users: Us.

of 75 percent of the query log, and a test set consisting of the remaining 25 percent. Traditional k-fold cross-validation is not applicable to temporally ordered data since it would obviously mess up the order [38]. Queries in the training set were submitted before May 8, 2006 in the AOL dataset and before October 1, 2013 in the SnV dataset. We also use the last week of training data to generate optimal parameters: N_{days} in (3) and λ^* in (8).

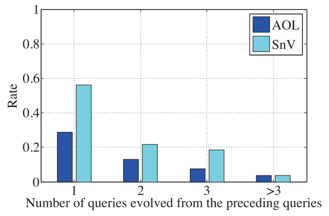
Moreover, we filtered out a large volume of navigational queries containing URL substrings (.com, .net, .org, http, .edu, www.) from the AOL dataset and removed queries starting with special characters such as &, \$ and # from both datasets. Additionally, only queries appearing in both partitions were kept. In total, 95,043 unique queries (21 percent) in the processed AOL and 6,023 (7 percent) in SnV show cyclic phenomena in terms of query frequency. Session boundaries are identified in the AOL dataset by 30 seconds of inactivity; in the SnV dataset a session boundary occurs when a query has no overlapping terms with the previous query as users routinely view audiovisual material during the search process; this can lead to periods of inactivity even though the user is still fully engaged in the search process [37]. Table 3 details the statistics of the datasets.

We display the overlap of queries with various ways of binning in Fig. 5. To begin, Fig. 5a shows the rates of unique $\langle user, query\rangle$ pairs posted at different numbers of repeats. A considerable number of queries are posted more than once by the same user within the training period (15.9 percent for AOL and 56.9 percent for SnV). The discrepancy between the rates can be explained by considering the type of user the search engine serves. Fig. 5b gives us the distribution of sessions containing queries that “evolved” from preceding queries within the session, where we say that query q_2 evolved from query q_1 if q_2 is issued after q_1 and shares at least one query term with q_1. Sessions with more than one query are considered. In total, there are 983,983 sessions in AOL and 35,942 in SnV left. Clearly, users reformulate a query very often. The difference between the sum of all rates (0.531 for AOL and 1 for SnV) is a consequence of different session segmentation methods.

In Fig. 6, we plot the the ratios of long-tail prefixes among all prefixes in the training and test periods of the AOL and SnV datasets, respectively. For AOL, nearly 13 percent of the prefixes return fewer than 10 query completions in both the training and test period. For SnV, less long-tail prefixes are detected, resulting in 12.5 and 12.7 percent for the training and test datasets, respectively. Hence, in general, we will



(a) **Distribution of unique pair $\langle$*user, query*$\rangle$ at various number of repeats.**



(b) **Distribution of sessions containing various number of “evolved queries.”**

Fig. 5. Query repeat rates (left) and variation rates (right) for AOL and SnV.

encounter at least one long-tail prefix among every 10 prefixes. Such long-tail prefixes often appear when a repeated submission of previous queries, typically unpopular ones, in the current session is observed. This finding motivates us to devote special attention to long-tail prefixes. In addition, we find that long-tail prefixes are often observed in search sessions where users resubmit queries that have been issued before in the current session, as shown in Fig. 3. Interestingly, for both datasets the percentage of prefixes with few query completions (e.g., 1 and 2) is higher than those with more query completions (e.g., 8 and 9).

4.3 Evaluation Metrics and Baselines

We first measure our forecast accuracy for the time-sensitive QAC model and then evaluate the effectiveness of the resulting ranked lists of query completions. For each task, we use metrics that are widely used in the literature on QAC task [3], [4], [6], [14].

Mean Absolute Error is widely used to measure the accuracy of forecasts and is defined as follows:

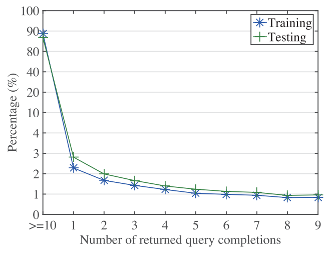
$$MAE = \frac{1}{n}\sum_{i=1}^{n} |\hat{y}_i - y_i|, \quad (16)$$

where y_i is the true value and $\hat{y}_i$ is the prediction. MAE is an unbounded measure and is not strongly resilient to outliers. Therefore, it is often used along with another metric such as Symmetric Mean Absolute Percentage Error (SMAPE) to diagnose the forecast variation. SMAPE is defined as

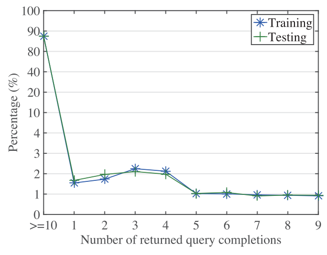
$$SMAPE = \frac{1}{n}\sum_{i=1}^{n} \frac{|\hat{y}_i - y_i|}{\hat{y}_i + y_i}. \quad (17)$$

In contrast to MAE, SMAPE is bounded between 0 and 1.

To evaluate the quality of rankings of query completions, Mean Reciprocal Rank is a standard measure. For a query q with prefix p in the query set Q associated with a list of



(a) **AOL**



(b) **SnV**

Fig. 6. Distribution of prefixes with varying numbers of returned query completions in the AOL and SnV datasets, tested on the training and test periods, respectively.

TABLE 4
Selecting Our Baseline

Model		AOL					SnV				
		$\#p=1$	$\#p=2$	$\#p=3$	$\#p=4$	$\#p=5$	$\#p=1$	$\#p=2$	$\#p=3$	$\#p=4$	$\#p=5$
MPC-ALL		0.1090	0.1903	0.3018	0.3996	0.4813	0.1573	0.2497	0.3281	0.4762	0.5438
	2 days	0.1093	0.1866	0.2989	0.3970	0.4681	0.2467	0.3526	0.4917	0.6096	0.6913
	4 days	0.1082	0.1814	0.2902	0.3875	0.4593	0.2281	0.3349	0.4751	0.5794	0.6681
MPC-R-	7 days	0.1120	0.1938	0.3107	0.4113	0.4830	0.2209	0.3158	0.4519	0.5528	0.6327
	14 days	0.1140	0.1994	0.3217	0.4254	0.4985	0.1953	0.2946	0.4318	0.5317	0.6108
	28 days	0.1147	0.2009	0.3233	0.4276	0.5076	0.1731	0.2690	0.3873	0.5167	0.5731
O-MPC-R		**0.1175**	**0.2027**	**0.3267**	**0.4318**	**0.5087**	**0.2519**	**0.3607**	**0.5034**	**0.6133**	**0.6992**

The performance of various baselines in terms of MRR, tested on the AOL and SnV datasets after typing 1 to 5 characters as prefix. The best performing baseline in each column is highlighted in boldface.

query completions $S(p)$ and the user's finally completed query q', Reciprocal Rank (RR) is computed as

$$RR = \begin{cases} \frac{1}{\text{rank of } q' \text{ in } S(p)}, & \text{if } q' \in S(p) \\ 0, & \text{else.} \end{cases} \quad (18)$$

Then MRR is computed as the mean of RR for all prefixes.

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 $\alpha = .01$ and $^\vartriangle$ ($^\triangledown$) for $\alpha = .05$.

We consider several QAC baselines: (1) the most popular completion QAC method based on the whole log, referred as MPC-ALL [3]; (2) an MPC-based QAC method within recent time windows (TW = 2, 4, 7, 14 and 28 days, respectively) denoted as MPC-R-TW [6]; (3) a recent QAC method with an optimal time window referred as O-MPC-R, which learns the optimal time window for each prefix and performs best on the AOL data in [6].

To select the best baseline against which we compare our newly introduced models, we compare the performance of the three approaches just listed; see Table 4. For both datasets, O-MPC-R outperforms the other two approaches at different prefix lengths, e.g., it results in close to 10 percent MRR improvements over MPC-ALL and MPC-R, respectively. Hence, we select O-MPC-R as the baseline for comparisons against our proposed models in latter experiments.

4.4 Experimental Settings

In our experiments, prefixes are simulated by considering all possible prefixes consisting of the first 1 to 5 characters of the finally submitted query. Following [39], we set the factor $f = 0.95$ in the decay function in Section 3.1. For time-sensitive prediction, we use a fixed $\lambda = 0.5$ in (2) to compare with the results produced with an optimal λ^* returned by (8). To detect periodicity, we count queries per hour for AOL and per day for SnV because of the difference in time spans of the collected data. This means that for SnV, we compute $\hat{y}_{t_0+1}(q)_{peri}$ in (5) directly by averaging the day-level predictions $y_{t_0+1-m\cdot T_q}$, while for AOL, we first generate predictions per hour and then accumulate them to produce $y_{t_0+1-m\cdot T_q}$. For identifying trends, we use per day counts to overcome sparsity. For smoothing in (5), we set $M = 3$, as it performs best when M changes from 1 to 10 in our trials. In our time-sensitive QAC experiments, we are given a list of top N query completions; we set $N = 10$ as this is commonly used by many web search engines.

We balance the contributions of Q_s and Q_u in (9), if available, by setting $\omega = 0.5$, and construct Q_u using the ten most frequent queries of the user while collecting all preceding queries in the current session to form Q_s (see Table 3). In particular, for users without long-term search history, i.e., for cold-start users, we only consider their short-term search history in the current session for personalization. It could help if we use the long-term search logs from similar users seen in the training period. For instance, based on the preceding queries within a current session issued by a new user, we can find a group of seen users in the training period who have often submitted the same queries before. By using the long-term search logs of users in this group, we can model the interests of the new user for personalization. For personalized QAC comparisons, we set the size of n-grams to be $n = 4$, which has been recommended in string search [40] to represent queries. For our hybrid models, we set $\gamma = 0.5$ in (13), which is also used by λ^*-H$'$-QAC for non-long-tail prefixes. In addition, we set the threshold $Num = 10$ when classifying prefixes in Algorithm 3.

5 Results and Discussions

In Section 5.1, we examine the performance of our time-sensitive QAC model in terms of its query popularity prediction performance, which we follow with a section about the trade-off of the parameter λ in Section 5.2. We examine the performance of various TS-QAC approaches in Section 5.3. Then, Section 5.4 details the effectiveness of our hybrid QAC model; Section 5.5 provides an analysis of the hybrid QAC model with various personalized QAC scenarios; Section 5.6 zooms in on the effect on ranking query completions by varying the contribution weight γ in the hybrid QAC model; Section 5.7 compares the performance of combined QAC models. Finally, Sections 5.8 and 5.9 detail the results of our model on long-tail prefixes.

5.1 Query Popularity Prediction Evaluation

Since the true popularity of query completions is unavailable at runtime, ranking models sort query completions according to their previously observed popularity [3] or predicted popularity inferred from previous logs [4]. We first evaluate the prediction accuracy on query popularity, and then measure the impact of these predictions on the quality of rankings of query completions in Section 5.3.

TABLE 5
The Forecast Metrics Produced by Different Methods on the AOL and SnV Dataset

	AOL		SnV	
Method	MAE	SMAPE	MAE	SMAPE
P_1	0.2906	<u>0.2278</u>	<u>1.2287</u>	<u>0.3104</u>
P_3	0.2944	0.2363	1.3739	0.3265
P_6	<u>0.2893</u>	0.2325	1.5751	0.3412
P_{trend}	0.2996	0.2313	1.2492	0.3117
λ-TS-QAC	$0.2848^{\triangle}$	$0.2197^{\blacktriangle}$	1.2291	$0.2959^{\blacktriangle}$
λ^*-TS-QAC	**0.2832**$^{\triangle}$	**0.2145**$^{\blacktriangle}$	**1.2067**$^{\blacktriangle}$	**0.2813**$^{\blacktriangle}$

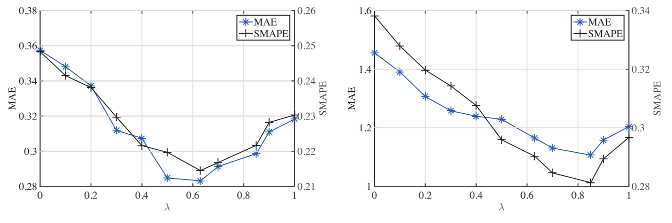
The best performer in each column is highlighted in boldface and the best performing baseline is underlined. Statistical significance of pairwise differences (λ-TS-QAC versus the best baseline P_* and λ^*-TS-QAC versus the best baseline P_*) are indicated.

Our time-sensitive prediction method considers both the recent and long-term query frequency to predict the future popularity. To compare, the predicted query frequencies are aggregated over a past query log (used in [4]) or only contributed over recent trend as described in (5). We denote the former by P_k where k is the number of previous days used for averaging ($k \in \{1, 3, 6\}$) and refer to the latter as P_{trend}. We do not take the prediction produced only by periodicity as a baseline because of the unavailability of sufficiently many periodic queries (21 percent in AOL and 7 percent in SnV, see Section 4.2). Table 5 includes the forecast error rates of different methods on datasets. The numbers show that λ^*-TS-QAC performs better in terms of MAE and SMAPE than all aggregation- and trend-based baselines, as well as λ-TS-QAC.

Still focusing on Table 5, we take a closer look at the error rates. The MAE achieved on AOL is much smaller than 1 due to the sparseness of query frequencies. Among the aggregated baselines, MAE favors P_6 and SMAPE prefers P_1 on AOL. However, for SnV, P_1 wins on both metrics. The numbers show that with the exception of P_1 on SnV, our predictions are better than all aggregated baselines on both metrics. The differences are statistically significant on SMAPE but not so according to MAE. Overall, the competitive performance on the AOL dataset can be explained by the fact that compared to the daily query frequency used in the SnV dataset, the data here is less sparse and has lower variance.

5.2 Impact of the Trade-Off Parameter λ

Next, we manually vary the parameter λ in (2) to determine the best prediction accuracy, with 0.01 increments. We show the results in Fig. 7. For AOL (Fig. 7a), λ^*-TS-QAC performs best in terms of prediction accuracy with $\lambda^* = 0.62$, suggesting the predictions should emphasize recent variations. We repeat our analysis on SnV and summarize the results in Fig. 7b. The results are consistent with the overall AOL numbers. SnV receives an optimal $\lambda^* = 0.83$ in our experiments. This is due to the fact SnV contains fewer periodic queries than AOL and hence it favors predictions from the trend.



(a) **Prediction accuracy at various λ on AOL.** (b) **Prediction accuracy at various λ on SnV.**

Fig. 7. Impact of the trade-off parameter λ in TS-QAC on the accuracy of query popularity prediction for AOL and SnV.

Another interesting finding on both datasets from Fig. 7 is that MAE and SMAPE favor a relatively larger λ and they share the same behaviors, e.g., MAE decreases as SMAPE comes down. This implies that (1) the recent trends are important to predict future popularity; and (2) periodic phenomena also contribute as the errors go up if their contribution is removed (i.e., with $\lambda = 1$).

Next, we take a close look at the optimal λ, i.e., λ^*. We find that for periodic queries, the optimal λ^* is often larger than 0.5. For instance, the mean of the optimal λ^* in the AOL dataset is close to 0.6. In other words, the contribution from the cyclic behavior of query popularity is relatively less important than that from recent trends. Neither of them is negligible for the prediction of query popularity as both weights are substantially larger than 0.

5.3 Performance of TS-QAC Ranking

For **RQ2**, we use MPC-based models to generate rankings of query completions for each prefix to compare with our results produced by time-sensitive QAC models, λ-TS-QAC and λ^*-TS-QAC. Table 6 contains the evaluation results for different QAC models in terms of MRR. On both two datasets, each prefix is used to generate ten rankings of query completions. For now, please ignore the λ^*-H-QAC row as we will get to it later. All pairwise differences are detected and marked if statistically significant.

TABLE 6
Performance in Terms of MRR at Prefix Length $\#p$ Ranging from 1 to 5 Characters on the AOL and SnV Datasets

	AOL					SnV				
Model	$\#p = 1$	$\#p = 2$	$\#p = 3$	$\#p = 4$	$\#p = 5$	$\#p = 1$	$\#p = 2$	$\#p = 3$	$\#p = 4$	$\#p = 5$
Baseline	0.1175	0.2027	0.3267	0.4318	0.5087	0.2519	0.3607	0.5034	0.6133	0.6992
λ-TS-QAC	0.1169	$0.1982^{\triangledown}$	0.3270	$0.4390^{\triangle}$	$0.5115^{\triangle}$	0.2536	$0.3726^{\blacktriangle}$	$0.5117^{\triangle}$	$0.6296^{\blacktriangle}$	$0.7103^{\blacktriangle}$
λ^*-TS-QAC	0.1208	$0.2056^{\triangle}$	$0.3317^{\triangle}$	$0.4455^{\blacktriangle}$	$0.5143^{\blacktriangle}$	$0.2637^{\blacktriangle}$	$0.3864^{\blacktriangle}$	$0.5193^{\blacktriangle}$	$0.6439^{\blacktriangle}$	$0.7203^{\blacktriangle}$
λ^*-H-QAC	**0.1224**$^{\blacktriangle}$	$0.2091^{\blacktriangle}$	$^{\triangle}0.3387^{\blacktriangle}$	$^{\triangle}0.4562^{\blacktriangle}$	$^{\triangle}0.5236^{\blacktriangle}$	**0.2662**$^{\blacktriangle}$	$0.3907^{\blacktriangle}$	$^{\blacktriangle}0.5355^{\blacktriangle}$	$^{\blacktriangle}0.6690^{\blacktriangle}$	$^{\blacktriangle}0.7491^{\blacktriangle}$
λ^*-H′-QAC	**0.1224**$^{\blacktriangle}$	**0.2103**$^{\blacktriangle}$	$^{\blacktriangle}$**0.3408**$^{\blacktriangle}$	$^{\blacktriangle}$**0.4594**$^{\blacktriangle}$	$^{\triangle}$**0.5278**$^{\blacktriangle}$	**0.2662**$^{\blacktriangle}$	**0.3913**$^{\blacktriangle}$	$^{\blacktriangle}$**0.5376**$^{\blacktriangle}$	$^{\blacktriangle}$**0.6702**$^{\blacktriangle}$	$^{\blacktriangle}$**0.7505**$^{\blacktriangle}$

The best performer in each column is highlighted in boldface. Statistically significant differences are determined against the baseline, i.e., O-MPC-R in Table 4, and marked in the upper right hand corner of the corresponding scores; statistically significant differences between λ^*-H-QAC and λ^*-H′-QAC versus λ^*-TS-QAC are also detected and marked in the upper left hand corner of the corresponding scores.

TABLE 7
MRR Changes Observed by Comparing O-MPC-R Against λ^*-H-QAC and λ^*-TS-QAC Against λ^*-H-QAC, Respectively, with a Query Prefix p of 1–5 Characters on AOL and SnV Query Logs

	AOL		SnV	
$\#p$	O-MPC-R	λ^*-TS-QAC	O-MPC-R	λ^*-TS-QAC
1	$-4.00\%^{\blacktriangledown}$	-1.31%	$-5.37\%^{\blacktriangledown}$	-0.94%
2	$-3.06\%^{\blacktriangledown}$	-1.67%	$-7.68\%^{\blacktriangledown}$	-1.10%
3	$-3.54\%^{\blacktriangledown}$	$-2.07\%^{\triangledown}$	$-5.99\%^{\blacktriangledown}$	$-3.03\%^{\blacktriangledown}$
4	$-5.35\%^{\blacktriangledown}$	$-2.35\%^{\triangledown}$	$-8.33\%^{\blacktriangledown}$	$-3.75\%^{\blacktriangledown}$
5	$-2.85\%^{\blacktriangledown}$	$-1.79\%^{\triangledown}$	$-6.67\%^{\blacktriangledown}$	$-3.84\%^{\blacktriangledown}$

The symbol "−" before MRR changes means λ^*-H-QAC outperforms the corresponding method. Statistical significance of pairwise differences (O-MPC-R versus λ^*-H-QAC and λ^*-TS-QAC versus λ^*-H-QAC) is indicated.

We find that λ^*-TS-QAC outperforms the baseline as well as λ-TS-QAC in terms of MRR, while λ-TS-QAC loses against the baseline for $\#p = 1$ and 2 on AOL. Specifically, λ^*-TS-QAC offers a maximal MRR increase against the baseline of 3.2 percent for $\#p = 4$, which is significant, and λ-TS-QAC brings an increase by up to 1.7 percent over the baseline for $\#p = 4$ on the AOL corpus. On the SnV dataset, we see the biggest performance improvements over the baseline: almost 7.1 percent for λ^*-TS-QAC and 3.3 percent for λ-TS-QAC both when expanding a two-character prefix. The limited improvement of λ-TS-QAC is probably due to predictions on occasional queries such as news search, whereas λ^*-TS-QAC smoothes it with cyclic phenomena for QAC.

With an optimal λ^* specifying the contributions from the recent trends and cyclic behavior, TS-QAC produces better rankings of query completions than with a fixed λ, as we can see by comparing λ-TS-QAC and λ^*-TS-QAC in Table 6. Generally, at different prefix lengths, λ^*-TS-QAC receives larger MRR gains over λ-TS-QAC on SnV than AOL. We attribute this to the volume of periodic queries in the different datasets as discussed in Section 5.2.

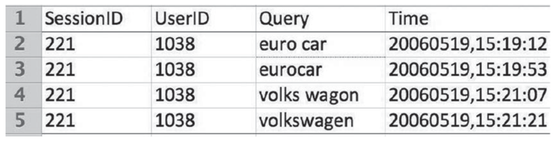
5.4 Hybrid QAC Ranking Performance

RQ3 is aimed at examining whether a user's personal query similarity helps generate better rankings of query completions. We first give the absolute MRR scores of λ^*-H-QAC in Table 6. For convenience, we report the MRR changes produced by comparing O-MPC-R against λ^*-H-QAC and λ^*-TS-QAC against λ^*-H-QAC in Table 7. With the appropriate regression model and query similarity measure, λ^*-H-QAC is able to marginally outperform the baselines on both query logs at each prefix length. Despite the additional overhead of scoring similarity between queries, λ^*-H-QAC presents relatively small (~2 percent) improvements over λ^*-TS-QAC on AOL. This is due to the fact that no strongly differential features have so far been explored for users.

Compared to AOL, λ^*-H-QAC on SnV achieves relative MRR gains over the baselines and the MRR differences are generally enlarged for longer prefixes. In part, this may be due to the following. First, AOL contains more queries than SnV queries, although these are spread sparsely over a three-month period. This could suggest that a search engine serving more queries is able to generate better query completions since it has a larger sample of similar behavior. Second, AOL is a more general search log across topics while SnV focuses on multimedia search. Third, there may be underlying demographic differences between users of the two search logs that lead to changes in query distributions, for example, AOL covers more public users while SnV mostly serves for media professionals. The higher performance of SnV as compared to AOL could be a consequence of the difference in user activity as *Qs/Us* in Table 3 indicates SnV users submit ∽20 times more queries than AOL users.

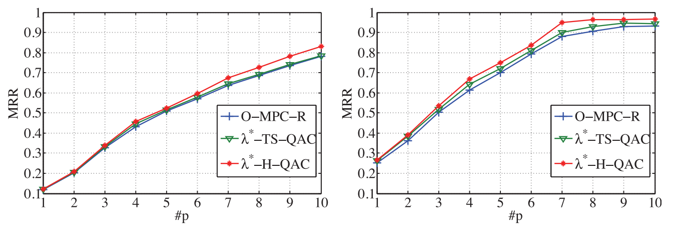
Clearly, for both query logs, λ^*-H-QAC is considerably more effective with a longer prefix, see Tables 6 and 7. To verify this, we examine the MRR scores with a longer prefix of up to 10 characters in Fig. 8. We find that effectiveness converges more quickly on SnV than AOL when the length of prefix increases, probably because QAC is constrained by how much evidence is available, and a slightly longer prefix hugely narrows the number of possible query completions, certainly on the SnV dataset.

To illustrate the effectiveness of our model, we consider an example from the AOL query log (Fig. 9). Assume that our user has entered the prefix *vo* of the last query in this session, so that we need to recommend query completions for this prefix. The results shown in Fig. 10 are generated by the O-MPC-R, λ^*-TS-QAC and λ^*-H-QAC approaches, respectively. Clearly, "volkswagon" and "volkswagen" benefit more from the search context than others as they are



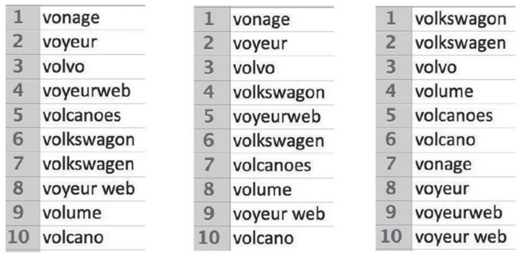
1	SessionID	UserID	Query	Time
2	221	1038	euro car	20060519,15:19:12
3	221	1038	eurocar	20060519,15:19:53
4	221	1038	volks wagon	20060519,15:21:07
5	221	1038	volkswagen	20060519,15:21:21

Fig. 9. An AOL test session.



(a) **AOL** (b) **SnV**

Fig. 8. QAC performance in terms of MRR observed for each approach, with a query prefix p of 1–10 characters for the AOL and SnV query logs.



	(a) O-MPC-R		(b) λ^*TS-QAC		(c) λ^*H-QAC
1	vonage	1	vonage	1	volkswagon
2	voyeur	2	voyeur	2	volkswagen
3	volvo	3	volvo	3	volvo
4	voyeurweb	4	volkswagon	4	volume
5	volcanoes	5	voyeurweb	5	volcanoes
6	volkswagon	6	volkswagen	6	volcano
7	volkswagen	7	volcanoes	7	vonage
8	voyeur web	8	volume	8	voyeur
9	volume	9	voyeur web	9	voyeurweb
10	volcano	10	volcano	10	voyeur web

(a) **Ranked by O-MPC-R.** (b) **Ranked by λ^*TS-QAC.** (c) **Ranked by λ^*H-QAC.**

Fig. 10. Ranked lists of query completions for the prefix *vo*.

TABLE 8
MRR Scores of G-QAC and Personalized QAC (Per. QAC), as well as MRR Changes in Bracket Produced by Comparing G-QAC Against λ^*-H-QAC (in Table 6), and Personalized QAC against λ^*-H-QAC, Respectively, with a Query Prefix p Length of 1–5 Characters Tested on AOL and SnV Query Logs

	AOL		SnV	
#p	G-QAC	Per. QAC	G-QAC	Per. QAC
1	0.1132▼ (−7.52%)	0.0174▼ (−85.78%)	0.2313▼ (−13.11%)	0.2427▼ (−8.83%)
2	0.1987▼ (−4.97%)	0.0688▼ (−67.10%)	0.3443▼ (−11.88%)	0.3619▼ (−7.35%)
3	0.3175▼ (−6.26%)	0.1371▼ (−59.52%)	0.4564▼ (−14.76%)	0.5018▼ (−6.29%)
4	0.4180▼ (−8.37%)	0.2256▼ (−50.55%)	0.5658▼ (−15.43%)	0.6197▼ (−7.37%)
5	0.4981▼ (−4.88%)	0.3312▼ (−36.75%)	0.6353▼ (−15.19%)	0.7098▼ (−5.25%)

Significant differences against λ^*-H-QAC are indicated.

ranked at the top by the λ^*-H-QAC approach; this is because they are closely related to the earlier query "volks wagon" (line 4 in Fig. 9). Hence, the queries "volkswagon" and "volkswagen" are more sensible completions. We could also explain it by introducing the semantic similarity between the query completion and the previous queries in session. We consider the query pair likelihood ratio [41, (LLR)] in (19) as

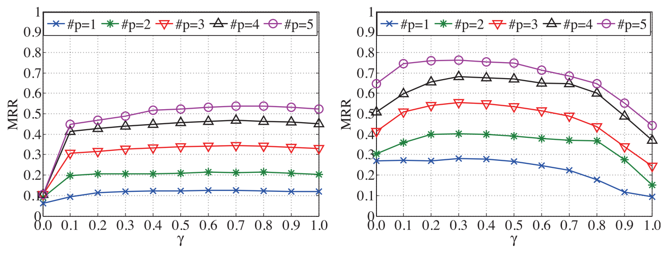
$$LLR(q_1, q_2) = -2\log\frac{L(q_2\,|\,\neg q_1)}{L(q_2\,|\,q_1)},$$

where $L(q_2\,|\,\neg q_1)$ denotes the number of queries containing q_2 but without q_1, and $L(q_2\,|\,q_1)$ indicates the volume of queries containing both q_1 and q_2, to see whether their co-occurrence in a search session is statistically significant. We find that, for instance, the pairs of query "volks wagon" and "volkswagon" or of query "volks wagon" and "volkswagen" co-occur relatively more often in sessions than other pairs.

5.5 Personalized QAC Performance Analysis

To help us answer **RQ4**, we compare the performance of λ^*-H-QAC with two personalized QAC scenarios (G-QAC and Personalized QAC listed in Table 2) and record the MRR scores of these two methods in Table 8. We also report the MRR changes produced by comparing G-QAC against λ^*-H-QAC, as well as Personalized QAC against λ^*-H-QAC in brackets in Table 8.

The model λ^*-H-QAC significantly outperforms G-QAC and Personalized QAC on both AOL and SnV in terms of MRR scores in all cases, which again confirms the above observations in Table 6. For AOL, Personalized QAC does not work well and its MRR scores are always substantially lower than those of G-QAC, suggesting that ranking query completions only according to query similarity on bigger dataset is not reliable because the number of query completions is very large and users often issue new queries. Interestingly, Personalized QAC outperforms G-QAC on SnV. We believe this can be attributed to (i) SnV users frequently



(a) **AOL** (b) **SnV**

Fig. 11. Performance of λ^*-H-QAC when varying the combination weight γ with a query prefix p length of 1–5 characters for the AOL and SnV query logs.

issue similar queries in the current search session or in a long-term period, yielding distinguishable similarity scores for query completions; and (ii) the average number of queries of SnV users is larger, resulting in a better estimation of Q_u in (9).

The MRR improvements of λ^*-H-QAC over G-QAC are still very high, indicating that MPC-ALL in G-QAC may often eliminate useful query completions. This may be strengthened due to the low volume of queries, as the relative changes on SnV (around 15 percent) are larger than those on AOL (around 7 percent). We conclude that a small dataset suffers more from uncertainty on query popularity for ranking query completions.

5.6 Effect of Contribution Weight γ

Next, we examine the effect on overall QAC performance by varying the contribution weight γ in (13) in our hybrid QAC model, λ^*-H-QAC, from 0 to 1, on AOL and SnV. See Fig. 11. For AOL (Fig. 11a), if the value of γ used in λ^*-H-QAC goes up from 0 to 0.4, the performance increases more dramatically compared with the results under other settings ($0.4 < \gamma \leq 1$). If we rank query completions only by query similarity, i.e., $\gamma = 0$, the performance is worse than any other result. The MRR value of λ^*-H-QAC reaches its peak around $\gamma = 0.7$ for all cases, which shows that λ^*-H-QAC favors time-sensitive popularity over user's query similarity on AOL. This finding is confirmed when we average MRR values produced under different settings: $0 \leq \gamma \leq 0.5$ and $0.5 \leq \gamma \leq 1$ for each length of prefix. The average MRR of the latter ($0.5 \leq \gamma \leq 1$) is higher for all cases.

In contrast to AOL, the optimal value of γ on SnV (Fig. 11b) is around 0.3, which indicates that ranking completions on SnV favors user's query similarity a bit more. The discrepancy between the optimal value of γ on SnV and the optimal value of γ on AOL can be explained by considering the number of issued queries of each user. Sufficient personal queries results in effective personalized QAC on SnV. The MRR of SnV tends to be more sensitive to γ than that of AOL as it varies dramatically with the increase of γ, especially when $0.5 \leq \gamma \leq 1$. The overall MRR score of λ^*-H-QAC is better than that produced by just setting $\gamma = 0$ or $\gamma = 1$, which is consistent with our findings for AOL.

5.7 Performance of Combined QAC Models

To answer **RQ5**, we compare λ^*-H$_G$-QAC (in Table 2) with λ^*-H-QAC (MRR scores reported in Table 6). The MRR scores of λ^*-H$_G$-QAC and the corresponding changes

TABLE 9
MRR Scores of λ^*-H$_G$-QAC, as well as MRR Changes Produced by Comparing λ^*-H$_G$-QAC Against λ^*-H-QAC (MRR Scores Presented in Table 6), with a Query Prefix p Length of 1–5 Characters Tested on the AOL and SnV Query Logs

	AOL		SnV	
$\#p$	MRR	change	MRR	change
1	0.1213	−0.90%	0.2650	−0.45%
2	0.2066$^{\triangledown}$	−1.21%$^{\triangledown}$	0.3891	−0.41%
3	0.3330$^{\triangledown}$	−1.68%$^{\triangledown}$	0.5309	−0.86%
4	0.4476$^{\blacktriangledown}$	−1.89%$^{\blacktriangledown}$	0.6617$^{\triangledown}$	−1.10%$^{\triangledown}$
5	0.5179	−1.09%	0.7398$^{\triangledown}$	−1.24%$^{\triangledown}$

Statistical differences (λ^*-H$_G$-QAC versus λ^*-H-QAC) are indicated.

against λ^*-H-QAC tested on AOL and SnV are recorded in Table 9. We find that λ^*-H$_G$-QAC performs better on SnV than on AOL, with higher MRR scores in all cases. However, λ^*-H-QAC consistently outperforms λ^*-H$_G$-QAC as the MRR changes produced by comparing λ^*-H$_G$-QAC against λ^*-H-QAC are always negative.

Another interesting finding is that λ^*-H$_G$-QAC performs very competitive with λ^*-H-QAC, especially on SnV, and the differences are limited (MRR changes $\sim$1 percent). This appears to be due to the fact that (i) λ^*-H$_G$-QAC scores the query similarity on a close character level but confronts the sparseness problem, and (ii) the number of grams n is artificially fixed, resulting in failure to rank query completions properly.

5.8 Performance on Long-Tail Prefixes

To answer **RQ6**, we examine the performance of λ^*-H$'$-QAC on the corresponding subset of the AOL and SnV datasets, only containing the long-tail prefixes to compare against the results produced by λ^*-H-QAC under setting $\gamma = 0.5$ in (13). We report the results in Table 10 in terms of MRR at various number of returned query completions (No.) ranging from 1 to 9, including the case of No. = 1 where λ^*-H$'$-QAC presents a draw with λ^*-H-QAC when only one query completion is returned.

We can see that from Table 10, generally, λ^*-H$'$-QAC outperforms λ^*-H-QAC in terms of MRR on both datasets. It achieves 1.94 and 2.45 percent MRR improvements on average over λ^*-H-QAC for all long-tail prefixes in the AOL and SnV subsets, respectively. Moreover, we checked the weights returned by a regression model, which control the contributions from the time-sensitive part and the personalized aspect in λ^*-H$'$-QAC, i.e., γ and $1-\gamma$ in (13), and found that γ is less than 0.5 on both datasets $\sim$0.42 for AOL and $\sim$0.31 for SnV), implying that personalization is more important for long-tail prefixes. For long-tail prefixes, the final submitted queries often occurred in the current session. In other words, for these cases, repeated query submissions in the same session are often observed.

One particularly interesting observation from Table 10 is that λ^*-H$'$-QAC achieves relatively larger MRR gains over λ^*-H-QAC when the median number of query completions (e.g., 4 or 5) are returned. This can be attributed to the following: (i) for cases with fewer query completions returned, e.g., 2 or 3, λ^*-H$'$-QAC achieves similar results, resulting in many draws; (ii) the cases with more query completions returned, e.g., 8 or 9, account for the minority of long-tail prefixes, as shown in Fig. 6, resulting in limited improvements.

TABLE 10
Performance of λ^*-H-QAC and λ^*-H$'$-QAC in Terms of MRR at Various Numbers of Returned Query Completions (No.) Ranging from 2 to 9 on the Subset of AOL and SnV Datasets Only Containing Long-Tail Prefixes

	AOL		SnV	
No.	λ^*-H-QAC	λ^*-H$'$-QAC	λ^*-H-QAC	λ^*-H$'$-QAC
1	1.0000	1.0000	1.0000	1.0000
2	0.7756	0.7842	0.8217	0.8305
3	0.6119	0.6207	0.6412	0.6547
4	0.4701	0.4863$^{\blacktriangle}$	0.5231	0.5398$^{\blacktriangle}$
5	0.4197	0.4318$^{\vartriangle}$	0.4729	0.4921$^{\blacktriangle}$
6	0.3558	0.3617	0.3927	0.4035$^{\vartriangle}$
7	0.2987	0.3014	0.3138	0.3198
8	0.2461	0.2492	0.2793	0.2816
9	0.2074	0.2098	0.2239	0.2267

Significance differences (λ^*-H$'$-QAC versus λ^*-H-QAC) are indicated.

5.9 Performance of Modified Hybrid QAC

Finally, we examine the overall performance of λ^*-H$'$-QAC on the whole datasets (AOL and SnV) to compare against other models. The results in terms of MRR scores are listed in Table 6, row 7.

We can see from Table 6 that (1) with long-tail prefix detection, our extended hybrid QAC model, i.e., λ^*-H$'$-QAC, receives the highest MRR scores among the five models at all lengths of prefix, suggesting that long-tail prefix detection helps boost QAC performance; (2) for some cases, e.g., $\#p = 3$ on AOL, significant improvements at level $\alpha = .01$ are observed by comparing λ^*-H$'$-QAC against λ^*-TS-QAC, however, which are not seen on the comparisons between λ^*-H-QAC and λ^*-TS-QAC; (3) λ^*-H$'$-QAC achieves the equal performance with λ^*-H-QAC at $\#p = 1$ because there is no long-tail prefix consisting of only one character.

In general, λ^*-H$'$-QAC achieves limited improvements over λ^*-H-QAC. This is because the majority of prefixes in the datasets are returned by more than ten query completions, in other words, they are not long-tail prefixes, and in such cases, λ^*-H$'$-QAC will degenerate to λ^*-H-QAC and then reports the same MRR scores with λ^*-H-QAC.

6 Conclusion

Most previous work on query auto completion focuses on either time-sensitive maximum likelihood estimation or context-aware similarity. In this paper we have adopted a combination of the two aspects of the QAC problem. To understand a user's personal search intent, we have extended our time-sensitive QAC method with personalized QAC, which infers the similarity between current requests and preceding queries in a current search session and previous search tasks at the character level. In addition, we have adjusted the model specific for long-tail prefixes. In particular, we assign an optimal weight $\bar{\gamma}$ in (13) to long-tail prefixes after checking their prefix popularity rather than using a fixed weight γ that is optimized for normal prefixes.

As to future work, we intend to have a closer look at the top N query completions returned by the popularity-based ranking method for $N > 10$: how much can we gain from good query completions that were ranked at lower ranks? Moreover, we aim to transfer our approach to other datasets with long-term query logs, which should help us benefit from queries with longer periodicity than we have access to in the AOL and SnV logs used in our current work. To which degree is it beneficial to diversify QAC results [42]? In addition, we could consider using an optimal weight for all prefixes in our proposal and study a cold-start problem where a user's long-term search logs are unavailable, which could be addressed by using the logs from a group of similar users seen in the training period. A further possible step is to model personalized temporal patterns for active users, especially professional searchers, requiring a generalization from actual query terms to topics or intents. This might help generate a better ranking of query completions for them.

ACKNOWLEDGMENTS

The authors thank our reviewers for their helpful and constructive feedback. This research was supported by the Innovation Foundation of NUDT for Postgraduate under No. B130503, the National Natural Science Foundation of China under No. 61202487, Amsterdam Data Science, the Bloomberg Research Grant program, 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 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, 652.002.001, 612.001.551, 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. An earlier version of this article appeared in the *Proceedings of CIKM '14* [1]. In this substantially extended version, we examine the performance on long-tail prefixes and produce a new model for query auto completion by extending the models presented in [1] with more research questions studied and answered. We also expand our discussion of related work and analysis of experimental results.

REFERENCES

[1] F. Cai, S. Liang, and M. de Rijke, "Time-sensitive personalized query auto-completion," in *Proc. 23rd ACM Int. Conf. Conf. Inf. Knowl. Manage.*, 2014, pp. 1599–1608.

[2] S. Chaudhuri and R. Kaushik, "Extending autocompletion to tolerate errors," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2009, pp. 707–718.

[3] Z. Bar-Yossef and N. Kraus, "Context-sensitive query auto-completion," in *Proc. 20th Int. Conf. World Wide Web*, 2011, pp. 107–116.

[4] M. Shokouhi and K. Radinsky, "Time-sensitive query auto-completion," in *Proc. 35th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2012, pp. 601–610.

[5] G. Pass, A. Chowdhury, and C. Torgeson, "A picture of search," in *Proc. 1st Int. Conf. Scalable Inf. Syst.*, 2006, Art. no. 1.

[6] S. Whiting and J. M. Jose, "Recent and robust query auto-completion," in *Proc. 23rd Int. Conf. World Wide Web*, 2014, pp. 971–982.

[7] N. G. Golbandi, L. K. Katzir, Y. K. Koren, and R. L. Lempel, "Expediting search trend detection via prediction of query counts," in *Proc. 6th ACM Int. Conf. Web Search Data Min.*, 2013, pp. 295–304.

[8] T. Miyanishi and T. Sakai, "Time-aware structured query suggestion," in *Proc. 36th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2013, pp. 809–812.

[9] M. Shokouhi, "Detecting seasonal queries by time-series analysis," in *Proc. 34th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2011, pp. 1171–1172.

[10] A. Strizhevskaya, A. Baytin, I. Galinskaya, and P. Serdyukov, "Actualization of query suggestions using query logs," in *Proc. 21st Int. Conf. World Wide Web*, 2012, pp. 611–612.

[11] A. Kulkarni, J. Teevan, K. M. Svore, and S. T. Dumais, "Understanding temporal query dynamics," in *Proc. 4th ACM Int. Conf. Web Search Data Min.*, 2011, pp. 167–176.

[12] F. Cai and M. de Rijke, "Learning from homologous queries and semantically related terms for query auto completion," *Inf. Proc. Manage.*, vol. 52, pp. 628–643, 2016, to be published.

[13] R. L. T. Santos, C. Macdonald, and I. Ounis, "Learning to rank query suggestions for adhoc and diversity search," *Inf. Retrieval*, vol. 16, pp. 429–451, 2013.

[14] M. Shokouhi, "Learning to personalize query auto-completion," in *Proc. 36th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2013, pp. 103–112.

[15] Z. Liao, D. Jiang, E. Chen, J. Pei, H. Cao, and H. Li, "Mining concept sequences from large-scale search logs for context-aware query suggestion," *ACM Trans. Intell. Syst. Technol.*, vol. 3, no. 1, 2011, Art. no. 17.

[16] F. Cai and M. de Rijke, "Selectively personalizing query auto-completion," in *Proc. ACM Int. Conf. Res. Develop. Inf. Retrieval*, 2016, pp. 993–996, to be published.

[17] J. Guo, X. Cheng, G. Xu, and X. Zhu, "Intent-aware query similarity," in *Proc. 20th ACM Int. Conf. Inf. Knowl. Manage.*, 2011, pp. 259–268.

[18] H. Cao, et al., "Context-aware query suggestion by mining click-through and session data," in *Proc. 14th ACM SIGKDD Int. Conf. Knowl. Discovery Data Min.*, 2008, pp. 875–883.

[19] I. Weber and C. Castillo, "The demographics of web search," in *Proc. 33rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2010, pp. 523–530.

[20] C. Sengstock and M. Gertz, "Conquer: A system for efficient context-aware query suggestions," in *Proc. 20th Int. Conf. Companion World Wide Web*, 2011, pp. 265–268.

[21] M. Arias, J. M. Cantera, and J. Vegas, "Context-based personalization for mobile web search," in *Proc. 34th Very Large Database Conf.*, 2008, pp. 33–39.

[22] S. Bhatia, D. Majumdar, and P. Mitra, "Query suggestions in the absence of query logs," in *Proc. 34th Int. ACM SIGIR Conf. Res. Develop. Retrieval*, 2011, pp. 795–804.

[23] J. Fan, H. Wu, G. Li, and L. Zhou, "Suggesting topic-based query terms as you type," in *Proc. 12th Int. Asia-Pacific Web Conf.*, 2010, pp. 61–67.

[24] S. Bickel, P. Haider, and T. Scheffer, "Learning to complete sentences," in *Proc. 16th Eur. Conf. Mach. Learn.*, 2005, pp. 497–504.

[25] K. Grabski and T. Scheffer, "Sentence completion," in *Proc. 27th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2004, pp. 433–439.

[26] K. Hofmann, B. Mitra, F. Radlinski, and M. Shokouhi, "An eye-tracking study of user interactions with query auto completion," in *Proc. 23rd ACM Int. Conf. Conf. Inf. Knowl. Manage.*, 2014, pp. 549–558.

[27] J.-Y. Jiang, Y.-Y. Ke, P.-Y. Chien, and P.-J. Cheng, "Learning user reformulation behavior for query auto-completion," in *Proc. 37th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2014, pp. 445–454.

[28] B. Mitra, M. Shokouhi, F. Radlinski, and K. Hofmann, "On user interactions with query auto-completion," in *Proc. 37th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2014, pp. 1055–1058.

[29] Y. Li, A. Dong, H. Wang, H. Deng, Y. Chang, and C. Zhai, "A two-dimensional click model for query auto-completion," in *Proc. 37th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2014, pp. 455–464.

[30] L. Li, H. Deng, A. Dong, Y. Chang, H. Zha, and R. Baeza-Yates, "Analyzing user's sequential behavior in query auto-completion via markov processes," in *Proc. 38th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2015, pp. 123–132.

[31] A. Zhang, et al., "adaQAC: Adaptive query auto-completion via implicit negative feedback," in *Proc. 38th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2015, pp. 143–152.

[32] F. Cai and M. de Rijke, "Query auto completion in information retrieval," *Found. Trends Inf. Retrieval*, 2016, submitted for publication.

[33] C. Chatfield, *The Analysis of Time Series: An Introduction*. Boca Raton, FL, USA: Chapman and Hall, 2004.

[34] F. Cai, S. Liang, and M. de Rijke, "Personalized document re-ranking based on bayesian probabilistic matrix factorization," in *Proc. 37th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2014, pp. 835–838.

[35] F. Cai, S. Wang, and M. de Rijke, "Behavior-based personalization in web search," *J. Assoc. Inf. Sci. Technol.*, 2016, to be published.

[36] N. Craswell, R. Jones, G. Dupret, and E. Viegas, Eds., *Proc. 2009 Workshop Web Search Click Data: WSCD'09*, ACM, 2009.

[37] B. Huurnink, L. Hollink, W. van den Heuvel, and M. de Rijke, "Search behavior of media professionals at an audiovisual archive: A transaction log analysis," *J. Amer. Soc. Inf. Sci. Techn.*, vol. 61, no. 6, pp. 1180–1197, Jun. 2010.

[38] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Comput. Surv.*, vol. 46, no. 4, pp. 44:1–44:37, Mar. 2014.

[39] P. N. Bennett, et al., "Modeling the impact of short-and long-term behavior on search personalization," in *Proc. 35th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2012, pp. 185–194.

[40] W. Litwin, R. Mokadem, P. Rigaux, and T. Schwarz, "Fast ngram-based string search over data encoded using algebraic signatures," in *Proc. 33rd Int. Conf. Very Large Data Bases*, 2007, pp. 207–218.

[41] R. Jones, B. Rey, O. Madani, and W. Greiner, "Generating query substitutions," in *Proc. 15th Int. Conf. World Wide Web*, 2006, pp. 387–396.

[42] F. Cai, R. Reinanda, and M. de Rijke, "Diversifying query auto-completion," *ACM Trans. Inf. Syst.*, vol. 34, 2016, Art. no. 25, to be published.

Fei Cai received the MSc degree in system engineering from the National University of Defense Technology, Changsha, China, in 2010. He is currently working toward the PhD degree at the Informatics Institute, University of Amsterdam, under the supervision of Maarten de Rijke. His research interests include information retrieval. He has several papers published in SIGIR and CIKM, and served as a PC member for CIKM 2015 and a reviewer for SIGIR, WWW, WSDM, CIKM, the *IEEE Transactions on Knowledge and Data Engineering*, the *Information Processing and Management*, the *Journal of the Association for Information Science and Technology*, etc.

Shangsong Liang received the MSc degree in computer science from Northwest A&F University, Xi'an, China, in 2010, and the PhD degree in computer science from the University of Amsterdam, Amsterdam, the Netherlands, in 2014. He is currently a postdoc at University College London, London, United Kingdom. His current research interests include information retrieval, expert finding, and search diversification. He has published papers in KDD, SIGIR and the *Information Processing and Management*.

Maarten de Rijke received the MSc degree in philosophy and mathematics in 1989 to 1990, and the PhD degree in theoretical computer science in 1993. He is a professor of computer science in the Informatics Institute at the University of Amsterdam. He worked as a postdoc at CWI, before becoming a Warwick research fellow at the University of Warwick, United Kingdom. He joined the University of Amsterdam in 1998, and was appointed full professor in 2004. His research focuses on intelligent information access, with projects on self-learning search engines and semantic search. He is the editor-in-chief of the *ACM Transactions on Information Systems and of Foundations* and the *Trends in Information Retrieval*.