Time-sensitive Personalized Query Auto-Completion

Fei Cai[†‡]
f.cai@uva.nl

Shangsong Liang[‡]
s.liang@uva.nl

Maarten de Rijke[‡]
derijke@uva.nl

[†]Key lab of Information System Engineering, National University of Defense Technology, Hunan, China
[‡]University of Amsterdam, Amsterdam, The Netherlands

ABSTRACT

Query auto-completion (QAC) is a prominent feature of modern search engines. It is aimed at saving user's time and enhancing the search experience. Current QAC models mostly rank matching QAC candidates according to their past popularity, i.e., frequency. However, query popularity changes over time and may vary drastically across users. Hence, rankings of QAC candidates should be adjusted accordingly. In previous work time-sensitive QAC models and user-specific QAC models have been developed separately. Both types of QAC model lead to important improvements over models that are neither time-sensitive nor personalized. We propose a hybrid QAC model that considers both of these aspects: time-sensitivity and personalization.

Using search logs, we return the top N QAC candidates by predicted popularity based on their recent trend and cyclic behavior. We use auto-correlation to detect query periodicity by long-term time-series analysis, and anticipate the query popularity trend based on observations within an optimal time window returned by a regression model. We rerank the returned top N candidates by integrating their similarities with a user's preceding queries (both in the current session and in previous sessions by the same user) on a character level to produce a final QAC list. Our experimental results on two real-world datasets show that our hybrid QAC model outperforms state-of-the-art time-sensitive QAC baseline, achieving total improvements of between 3% and 7% in terms of MRR.

Categories and Subject Descriptors

H.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

Keywords

Query auto-completion; personalization; time-sensitive

1. INTRODUCTION

To improve the search quality, common search engines and popular online properties such as online shopping and email, all provide a query auto-completion (QAC) service, where the goal is to help users formulate queries by providing possible completions matching the first few keystrokes typed. As a user enters a query in search box, matching completions appear below the search box as a drop-down menu with the typed characters highlighted in each completion. Once the matching candidates are filtered, they can be ranked according to different criteria. For instance, in an online store such completions may be ordered according to the price of products [12]. In pre-computed auto-completion systems, the list of matching candidates for each prefix are generated in advance and stored in efficient data structure for fast lookups. When needed, as shown in Fig. 1, continuing typing more characters can dynamically refine the completions by exact prefix matching until you find a more appropriate completion. Where offered, the facility is heavily used and highly influential on search results [3, 32].

(a) Query auto-completion of typed prefix *cikm*.

(b) Query auto-completion of typed prefix *cikm conference*.

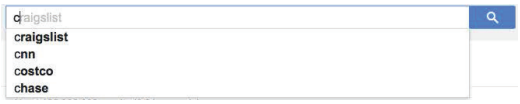
Figure 1: (Top) Query auto-completion by Google for the prefix *cikm*. (Bottom) The refined completions with after continuing to type *conference* after *cikm*. The snapshot was taken on Thursday, April 10, 2014.

Clearly, query auto-completion, or "type ahead" functionality, just takes a few initial keystrokes as input and returns matching queries to auto-complete the search clue. QAC candidates can be weighted by previous popularity and can either be influenced or hard-sorted by this factor or any other criterion, so that the most possible completions are listed first. Generally, queries can be completed from one or more of the following sources: (i) previous query popularity; (ii) search behavior; (iii) for email search, sender/recipient names using the field aliases (to:, from: and cc:).

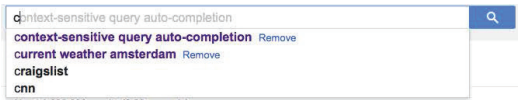
A common approach in previous work on QAC is to extract past queries with each prefix from a period of query logs, and rank them by their past popularity [3, 32, 33], which assumes that current or future query popularity is the same as past query popularity. Although this approach results in satisfactory QAC performance on average, it is far from optimal since it fails to take strong clues from time, trend and user-specific context into consideration while such information often influences the queries most likely to be typed. As illustrated in Fig. 2, personalized QAC may inject the most popular completions from a user as query completions to that user; see Fig. 2a (not personalized) and 2b (personalized). From Fig. 2c and Fig. 2d according to Google Trends, we can also find the query popularity strongly depends on the time (a clear burst for *MH370*

CIKM'14, November 3–7, 2014, Shanghai, China.

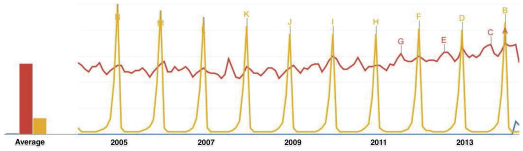
ACM 978-1-4503-2598-1/14/11 ...$15.00.
http://dx.doi.org/10.1145/2661829.2661921 .



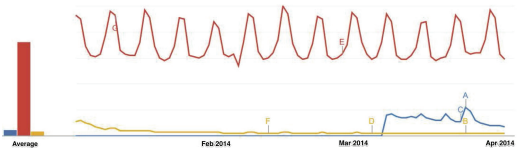
(a) Google QAC of typed prefix c without logging in.



(b) Google QAC of typed prefix c after logging in.



(c) Relative query popularity in the past 10 years (2004-2013).



(d) Relative query popularity within the past 90 days (from 10-Jan-2014 to 9-Apr-2014).

Figure 2: (Top) Google QAC of the typed prefix c under different logging settings. (Bottom) Relative query popularity for different queries over time. Queries: *MH370* in blue, *movie* in red and *christmas* in yellow. Among the three queries, *movie* is more popular weekly on weekends, while *christmas* is issued more commonly by users yearly, and *MH370* presents a sharp increase from a relatively low level near March 8, 2014. The snapshot was taken on Thursday, April 10, 2014.

around 8 March, 2014), and it presents cyclic phenomena (yearly for *christmas* and weekly for *movie*), which can be explored to forecast the future query popularity.[1] This motivates a QAC approach that takes both the temporal aspect and the personal context into account. This work is an attempt towards this objective.

To begin with, we differentiate query auto-completion (QAC) with query suggestion (QS) as follows:

Definition Let a string s with length l be keystrokes typed by user u in search box. A query suggestion QS of s for u with the form $QS(s,u)$ is a set of candidate queries q, where $q \in QS(s,u)$ is considered to be $relevant$ to s. Similarly, a query auto completion QAC of s for u with the form $QAC(s,u)$ is a set of candidate queries q, where $QAC(s,u) = \{q : q\ StartsWith(s)\}$, namely $q[i] = s[i]$ for i from 1 to l.

From this definition, $QS(s,u)$ may cover more general queries than $QAC(s,u)$. Also, we classify queries that need to be considered for completion into two categories. The first category corresponds to periodic queries that are: (i) consistently popular with short-term periodicity (e.g., *movie*); (ii) temporally recurring with long-term periodicity (e.g., *christmas*). The second category corresponds to aperiodic queries related to entirely unforeseeable breaking events and phenomena (e.g., *MH370*). Therefore, achieving optimal QAC effectiveness for a user, on average, is attributed to two factors: time-sensitive query popularity and personal context.

In our QAC model we first return the top N query completions by predicted popularity, not only based on the recent trend but also based on cyclic phenomena; we then rerank the returned top N completions by user-specific context to output a final query completion list. Predicted query popularity is based on two aspects, i.e., periodicity of a query detected by long-term time-series analysis plus its recent trend as indicated by observations during an optimal time window returned by a regression model. Rather than summarizing recent variations of query popularity in a fixed time span, we anticipate the trend from different periods of observations for each query. Secondly, we exploit each user's previous queries, both during the current session and from historical logs as user-specific context for reranking the top N QAC candidates. We combine the contributions from the predicted popularity and user-specific query similarity to produce a personalized list of QAC candidates.

We show that 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, with improvements in Mean Reciprocal Rank (MRR) scores by 3% on the AOL log and 7% on a search log from an audiovisual archive when compared to a state-of-the-art time-sensitive baseline [36].

Our contributions in this paper can be summarized as follows:

1. We tackle 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 achieves a better QAC ranking in terms of Mean Reciprocal Rank (MRR).
3. We analyze the effectiveness of our hybrid QAC model which considers predicted query popularity and user-specific context and find that it significantly outperforms state-of-the-art time-sensitive QAC methods.

2. RELATED WORK

Query auto-completion (QAC) [4, 5, 19, 32] is a prominent feature of common search engines. It relies on query logs to generate QAC candidates, and is among the first services that users interact when using an information retrieval system as they search and formulate their queries [31]. In major web search scenarios, the common and straightforward approach to rank QAC candidates 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 completion candidates 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 earlier, query popularity may change over time and the ranking of completions is also user-dependent (see Fig. 2). Accordingly, the QAC candidates must be adjusted consequently to account for time-sensitive and user-specific changes.

2.1 Time-sensitive query auto-completion

Time-sensitive query auto-completion (TS-QAC) takes time information, such as recency [16, 26, 36] and seasonality [30, 32], into consideration for ranking QAC candidates. It leverages time-series analysis techniques for classifying seasonal queries and forecasting their future popularity [30, 32]. Alfonseca et al. [1] cluster

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

queries based on time-series features; they suggest that their approach can be used for query completion and query categorization.

Rather than ranking QAC candidates by their previously observed popularity, Shokouhi and Radinsky [32] propose a long-term time-series modeling approach to forecast the query frequencies via applying a fixed moving time window. Queries recurring during specific temporal intervals, such as day/night, workday/weekend, summer/winter, etc. are modeled differently to predict future popularity for QAC ranking at different times. The forecasts obtained by such time-series modeling are substantially more reliable. However, the detailed analysis of the performance impact of the time window period selection and how to choose the optimal length of time window is unsolved. Similarly, Strizhevskaya et al. [33] 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. [16] develop a regression model to detect bursting queries for enhancing trend detection. By analyzing query logs, they seek to accurately predict what are the most trending query items 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. Kulkarni et al. [21] classify queries into different categories based on the changes in popularity over time, and show that monitoring the query popularity can reveal strong signals for detecting the trend in query intent.

White and Marchionini [35] propose a query completion model that produces an updated list of additional terms as a searcher enters his query. Real-time query completion is found to help users form better queries. Chien and Immorlica [13] demonstrate that queries with similar temporal patterns can be semantically related for query completion despite no lexical overlap. Liu et al. [24] introduce a unified model for forecasting query frequency, in which the forecast for each query is influenced by the frequencies predicted for semantically similar queries. Michail et al. [25] develop a compressed representation for time-series and propose a model for detecting bursts in query frequencies; these approaches can be utilized to boost QAC effectiveness.

Recently, considering both recent trend and past query popularity, Whiting and Jose [36] propose several practical QAC ranking approaches to deal with both robust and time-sensitive QAC task (as baselines), such as outputting query popularity evidence from a sliding window of 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.

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

2.2 User-specific query auto-completion

In most work mentioned so far, QAC candidates are computed globally and for a given prefix: all users are presented with the same list of candidates. But exploiting the user's personal context has led to increases in QAC effectiveness [3, 22, 28, 31].

Bar-Yossef and Kraus [3] treat the user's recent queries as context and exploit users with shared search activity, considering the similarity of QAC candidates with this context for ranking. Their hybrid model computes the final score of each candidate by linearly combining the MPC score and a context-similarity score. Our approach to personalized QAC differs in the definition of and in the way we measure context-similarity. Shokouhi [31] 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 QAC ranking, to therefore personalize the QAC ranking.

Guo et al. [18] propose a two-step approach, in which the user's session context is matched against pre-generated topic models for ranking QAC candidates. Similarly, Cao et al. [10] and Liao et al. [22] 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 QAC candidates. Weber and Castillo [34] 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 [29] consider query completions that depend on the time of day, i.e., taking the search time as a user-specific context. Arias et al. [2] 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. [7] mine frequently occurring phrases and n-grams from indexed documents for generating and ranking QAC candidates for partial queries in the absence of search logs. Fan et al. [14] propose a generative model that incorporates the topical coherence of terms based on Latent Dirichlet Allocation (LDA) for ranking QAC candidates. Closely similar to QAC, Bickel et al. [8] learn a linearly interpolated n-gram model for sentence completion based on lexicon statistics of text collections. Grabski and Scheffer [17] deploy an index-based retrieval algorithm and a cluster-based approach for their sentence-completion task.

Most user specific QAC approaches (re-)rank QAC candidates by measuring their similarity with the content in search logs, overlooking the updated query popularity. To the best of our knowledge, there is no published work on QAC considering both time-sensitivity and user-specificity. Previous work either focused on one or the other of these two aspects of QAC. Combining them is our goal here: time-sensitive personalized query auto-completion. By doing so our approach to personalized QAC stands to gain from repetitions in user search behavior.

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 considers a user's personal context. Table 1 provides an overview of the QAC approaches we discuss; 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 (TS-QAC) method that ranks QAC candidates 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 $\tilde{y}_{t_0+1}(q,\lambda)$ at day t_0+1 before day t_0 by both its recent trend and periodicity with a free parameter λ $(0 \leq \lambda \leq 1)$ controlling each contribution:

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

Table 1: Description of various QAC methods.

Approach	Description	Source
MPC-ALL	A QAC model that ranks QAC candidates according to their past popularity on the whole log	Reference [3]
MPC-R-TW	A QAC model that ranks QAC candidates according to their past popularity within a fixed time window	Reference [36]
O-MPC-R	A QAC model that ranks QAC candidates according to their past popularity within an optimal time window	Reference [36]
λ-TS-QAC	A time-sensitive QAC model (see Algorithm 1) with a fixed parameter $\lambda = 0.5$ used in (2)	This paper
λ^*-TS-QAC	A time-sensitive QAC model (see Algorithm 1) with an optimal parameter λ^* used in (2), achieved by (8)	This paper
Personalized QAC	A QAC model ranking QAC candidates according to their similarity to previous queries, following (9)	This paper
λ-H-QAC	A hybrid QAC model (see Algorithm 2) integrating λ-TS-QAC as (14) and our personalized QAC as (15)	This paper
λ^*-H-QAC	A hybrid QAC model (see Algorithm 2) integrating λ^*-TS-QAC achieved by (8) and our personalized QAC as (15)	This paper
G-QAC	A combined QAC model integrating MPC-ALL and personalized QAC using n-gram based query similarity as (16)	This paper
λ^*-H$_G$-QAC	A combined QAC model integrating λ^*-TS-QAC and personalized QAC using n-gram based query similarity as (16)	This paper

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 linear aggregation of predictions from recent N_{days} observations:

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

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 [36]. The prediction $\hat{y}_{t_0+1}(q,i)_{trend}$ from each day i ($i = 1, \ldots, N_{days}$) is derived from the first order derivative of q's daily count $C(q,t)$ as:

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

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 $t_p = t_0 + 1 - 1 \cdot T_q, \ldots, t_0 + 1 - M \cdot T_q$ in the log:

$$\hat{y}_{t_0+1}(q)_{peri} = \frac{1}{M} \sum_{m=1}^{M} y_{t_0+1-m\times T_q}(q), \quad (5)$$

where T_q denotes q's periodicity. For detecting cyclic aspects of query q's frequency, we use the autocorrelation coefficients [11], which measure the correlation between N_s successive count observations $C(q,t)$ at different times $t = 1, 2, \ldots, N_s$ in the query log. The correlation is computed between a time series and the same series lagged by i time units:

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

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 N_s reasonably large, the denominator in (6) 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. Accordingly, r_i can be approximated by:

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

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

Algorithm 1 Time-sensitive query auto-completion (TS-QAC).

Input: All queries: Q; Length of training and validation days: L_t and L_v; t_0; Number of returned completions: N;
Output: Predictions: $\bar{Q} = \{\bar{y}_{t_0+1}(q)\colon q \in Q\}$; Top N completions of each prefix of all queries;

for each $q \in Q$ **do**
  $T_q \leftarrow autocor(Count(q))$;
  for $i = 1, \cdots, L_t$ **do**
    for $j = 1, \cdots, L_v$ **do**
      $\hat{y}_{t_0+1}(q)_{trend}[j] \leftarrow Regression(Count(q)[1:i])$;
      $AbsoluteError[j] \leftarrow |\hat{y}_{t_0+1}(q)_{trend}[j] - y_{t_0+1}(q)|$;
    end for
    $MAE(i) \leftarrow mean(AbsoluteError)$;
  end for
  $N_{days} \leftarrow \arg\min_{1\leq i\leq L_t} MAE(i)$;
  Update $\hat{y}_{t_0+1}(q)_{trend}$ with optimal N_{days} and Compute $\bar{y}_{t_0+1}(q)_{peri}$;
end for
Find an optimal λ^* by (8);
$\lambda \leftarrow \lambda^*$;
for each $q \in Q$ **do**
  $\tilde{y}_{t_0+1}(q,\lambda) \leftarrow \lambda \times \hat{y}_{t_0+1}(q)_{trend} + (1-\lambda) \times \bar{y}_{t_0+1}(q)_{peri}$;
end for
for each $q \in Q$ **do**
  for each prefix p of q **do**
    Return top N completions of p ranked by $\tilde{y}_{t_0+1}(q,\lambda)$;
  end for
end for

Additionally, we choose an optimal free parameter λ^* by minimizing a forecast accuracy metric *Mean Absolute Error* (MAE) (described in §4.3) as:

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

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

Algorithm 1 details the major steps of time-sensitive QAC. For our time-sensitive QAC method with a fixed λ we write λ-TS-QAC; we write λ^*-TS-QAC when we use an optimal λ^*.

3.2 Personalized QAC

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

Our personalized QAC works here by scoring the candidates $q_c \in \mathcal{S}(p)$ using a combination of 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:

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

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

To compute the 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. From the statistics of our datasets (see Table 2 and Fig. 3), we find that users often request the same query or reformulate the query by extending or simplifying previous ones within the same session. We treat a user's preceding queries Q_s in the current session and Q_u in the historical log as context to personalize QAC where we measure similarity at the character level.

We represent each 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 $\{w_{c1}, w_{c1}, \ldots, w_{cn}\}$. We let $N(w_*, q_*)$ denote the frequency of term w_* appearing in q_*. We score the similarity between q_c and Q_s as a conditional probability:

$$Score(Q_s, q_c) = p(q_c|Q_s) = \sum_{q_s \in Q_s} norm(\omega_s) \cdot p(q_c|q_s), \quad (10)$$

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 [9] as:

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

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

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

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 (9) because in this setting it is desirable to consider both query count and time interval. We output $Score(Q_u, q_c)$ as:

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

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 (TS-QAC) with personalized QAC. First, TS-QAC produces a list of QAC candidates $S(p)$ of prefix p. We assign $TSscore(q_c)$ to each candidate $q_c \in S(p)$ 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:

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

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:

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

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

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

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; (13) provides the overall ranking score (see line 15).

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

Algorithm 2 Hybrid QAC model

Input: Predictions: $\bar{Q}$; user: u; prefix p; N; α;
Output: Ranking list of top N QAC candidates of p;
1: Produce $S(p)$ consisting of top N QAC candidates 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 (10);
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 $S(p)$;

3.4 Combined QAC models

For comparison, we also introduce further combined QAC models that combine popularity and personalization. G-QAC and λ^*-H$_G$-QAC rank QAC candidates according to a combined score:

$$Cscore(q_c) = \gamma \cdot MPCscore(q_c) + (1-\gamma) \cdot Gscore(q_c), \quad (16)$$

where G-QAC $MPCscore(q_c)$ is obtained using MPC-ALL and $Gscore(q_c)$ by measuring similarity between q_c and previous queries using an n-gram representation; λ^*-H$_G$-QAC obtains the score $Cscore(q_c)$ by combining λ^*-TS-QAC score (see §3.1) as $MPCscore(q_c)$ with the n-gram similarity score as $Gscore(q_c)$.

4. EXPERIMENTAL SETUP

Below, §4.1 lists the research questions to guide our experiments; §4.2 describes the datasets; §4.3 gives details about our evaluation metrics and baselines; we detail our settings and parameters in §4.4.

4.1 Research questions

The research questions guiding the remainder of the paper are:

RQ1 As a sanity check, what is the relative performance of our time-sensitive QAC models λ-TS-QAC and λ^*-TS-QAC in terms of query popularity prediction? (See §5.1.)

RQ2 How does the trade-off between recent trend and periodicity (as encoded in λ) impact the accuracy of query popularity prediction? (See §5.2.)

RQ3 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 §5.3.)

RQ4 Does our λ^*-H-QAC significantly outperform time-sensitive QAC methods (O-MPC-R and λ^*-TS-QAC)? (See §5.4.)

RQ5 How does λ^*-H-QAC compare against personalized QAC method using n-gram based query similarity (G-QAC)? (See §5.5.)

RQ6 Which part contributes more to a better QAC ranking, the predicted popularity or the query similarity? (See §5.6.)

RQ7 How do λ^*-H$_G$-QAC and λ^*-H-QAC compare? (See §5.7.)

4.2 Dataset

We use two query log datasets[2] in our experiments: AOL [27] and one made available by The Netherlands Institute for Sound and Vision,[3] a large audiovisual archive [20], which we will refer to as "SnV." AOL 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 of 75% of the query log, and a test set consisting of the remaining 25%. Traditional k-fold cross-validation is not applicable to streaming sequence since it would disorder the temporal data [15]. 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 the 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 two partitions were kept. In total, 95,043 unique queries (21%) in the processed AOL and 6,023 (7%) 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 [20]. Table 2 details the statistics of the datasets.

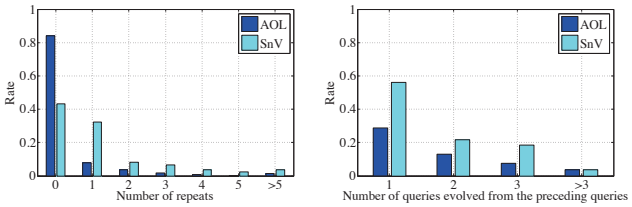
We display the overlaps of queries with various ways of binning in Fig. 3. Fig. 3a shows the rates of unique <*user*, *query*> pairs posted at different number of repeats. A considerable number of queries are posted more than once by the same user within the training period (15.9% for AOL and 56.9% for SnV). The discrepancy between the rates can be explained by considering the type of users the search engine serves. Fig. 3b 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 common 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 from its previous queries. The difference between the sum of all rates (0.531 for AOL and 1 for SnV) is a consequence of different session segmentation methods.

[2]We did not use the MSN and Sogou query logs as the former lacks users IDs and the latter is too small.

[3]http://www.beeldengeluid.nl

Table 2: Statistics of processed AOL and SnV Dataset. Queries: Qs, Sessions: Ss, Users: Us, URLs: Clicked Ds.

	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/Ss	1.36	1.63	1.65	1.51
#Qs/Us	14.81	11.49	277.25	192.50



(a) Distribution of unique pair <*user, query*> at various number of repeats.

(b) Distribution of sessions containing various number of evolved queries of the preceding queries.

Figure 3: Query repeat rates (left) and variation rates (right) for AOL and SnV.

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 QAC rankings. For each task, we use metrics from statistics and information retrieval for measurement, which are widely used in the literature on QAC task [3, 31, 32, 36].

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

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i|, \tag{17}$$

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, its is often used along with another metric as Symmetric Mean Absolute Percentage Error (SMAPE) to diagnose the forecast variation. SMAPE is defined as:

$$SMAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|\hat{y}_i - y_i|}{\hat{y}_i + y_i}, \tag{18}$$

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

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 completed query q', Reciprocal Rank (RR) is computed as:

$$RR = \begin{cases} \frac{1}{\text{rank of } q' \text{ in } S(p)}, & \text{if } q' \in S(p) \\ 0, & \text{else.} \end{cases} \tag{19}$$

Table 3: The forecast metrics produced by different methods on AOL and SnV dataset. The best performer in each column is highlighted. Statistical significance of pairwise differences (λ-TS-QAC vs. the best baseline P_* and λ^*-TS-QAC vs. the best baseline P_*) are detected by the t-test ($^{\blacktriangle}/^{\blacktriangledown}$ for α = .01, or $^{\triangle}/^{\triangledown}$ for α = .05).

	AOL		SnV	
Method	MAE	SMAPE	MAE	SMAPE
P_1	0.2906	0.2278	1.2287	0.3104
P_3	0.2944	0.2363	1.3739	0.3265
P_6	0.2893	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**$^{\blacktriangle}$

Then MRR is computed as the mean of RR for all queries in Q.

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 $^{\triangle}/^{\triangledown}$ for $\alpha = .05$.

We consider several QAC baselines: (1) the most popular completion (MPC) 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 [36]; (3) a recent QAC method with an optimized time window referred as O-MPC-R, which learns the optimal time window for each prefix and performs best on the AOL dataset in [36].

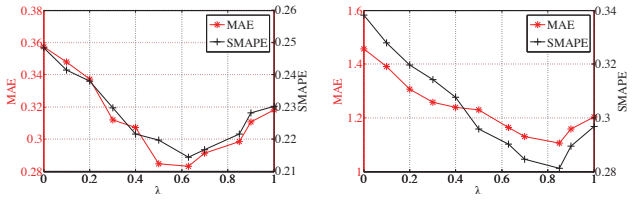
4.4 Settings and parameters

Following [6], we set the factor $f = 0.95$ in decay function in §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 $y_{t_0+1-m\times T_q}$, while for AOL, we firstly generate predictions per hour and then accumulate them to produce $y_{t_0+1-m\times T_q}$. 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 QAC candidates; 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 at most ten queries issued before while collecting all preceding queries in the current session to form Q_s (see Table 2). For personalized QAC comparisons, we set the size of n-grams to be $n = 4$, which has been recommended in string search [23] to represent queries. For hybrid models, we set $\gamma = 0.5$ in (13).

5. RESULTS AND DISCUSSIONS

In §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 §5.2. We examine the performance of various TS-QAC approaches in §5.3. §5.4 details the effectiveness of our hybrid QAC model; §5.5 provides an analysis of the hybrid QAC model with various personalized QAC scenarios; §5.6 zooms in on the effect on QAC ranking via varying the contribution weight in hybrid QAC model. §5.7 compares the performance of combined QAC models.



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

Figure 4: Impact of the trade-off λ in TS-QAC on the accuracy of query popularity prediction for AOL (left) and SnV (right).

5.1 Query popularity prediction evaluation

Since the true popularity of QAC candidates is unavailable at runtime, QAC ranking models order candidates according to their previously observed popularity [3] or predicted popularity inferred from previous logs [32]. In this section, we evaluate the prediction accuracy on query popularity, and measure the impact of these predictions on the quality of QAC rankings in section §5.3.

Our time-sensitive prediction method considers both the recent and long-term query frequency as predicted popularity for future. To compare, the predicted query frequencies are aggregated over a past query log (used in [32]) or only contributed over recent trend as described in (4). 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 baseline because of deficiency of periodic queries (21% in AOL and 7% in SnV, see §4.2). Table 3 includes the forecast error rates of different methods on datasets. The numbers show that our λ^*-TS-QAC performs better in terms of MAE and SMAPE than all aggregation- and trend-based baselines, as well as λ-TS-QAC.

We take a closer look at the error rates produced by various methods. The MAE achieved on the AOL dataset is much smaller than 1 owing 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 the competition 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 is 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 have lower variance.

5.2 Impact of the trade-off parameter λ

To answer **RQ2**, we manually vary the parameter λ in (2) to achieve the best prediction accuracy. We show the results in Fig. 4

For AOL in Fig. 4a, λ^*-TS-QAC performs best in terms of prediction accuracy under the setting $\lambda^* = 0.62$ achieved by optimization as (2), suggesting the predictions emphasize a bit more on recent variations. We repeat our analysis on SnV and summarized the results in Fig. 4b. 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 less periodic queries than AOL and hence it favors the predictions from the trend more.

5.3 Performance of TS-QAC rankings

Next, we turn to **RQ3** and use MPC-based models to generate QAC rankings for each query to compare with our results produced by time-sensitive QAC models, namely, λ-TS-QAC and λ^*-TS-QAC. Table 4 contains the evaluation results of different QAC models in terms of MRR. On both two datasets, each prefix is used to

Table 5: 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 length of 1-5 characters on AOL and SnV query logs. The symbol "-" before MRR changes means λ^*-H-QAC outperforms the corresponding method. Statistical significance of pairwise differences (O-MPC-R vs. λ^*-H-QAC and λ^*-TS-QAC vs. λ^*-H-QAC) is detected by the t-test ($^{\blacktriangle}/^{\blacktriangledown}$ for α = .01, or $^{\triangle}/^{\triangledown}$ for α = .05).

	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%$^{\blacktriangledown}$

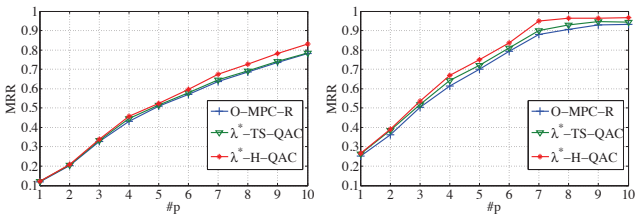
generate 10 QAC rankings, one for each day in the test period. For now, ignore the λ^*-H-QAC column as we will get to it later. The numbers in the table are averaged across all queries with different length of typed prefix p. All pairwise differences are detected and marked if statistically significant.

We find that λ^*-TS-QAC outperforms all MPC-based baselines as well as λ-TS-QAC in terms of MRR, whereas λ-TS-QAC loses two competitions against O-MPC-R under $\#p$ =1 and 2 on AOL. Detailedly, λ^*-TS-QAC offers a maximal MRR increase against O-MPC-R by up to 3.2% significantly at $\#p = 4$, and λ-TS-QAC brings an increase by up to 1.7% over O-MPC-R at $\#p = 4$ on AOL. Specifically on SnV, we see the best performance improvement over O-MPC-R almost 7.1% brought by λ^*-TS-QAC and 3.3% by λ-TS-QAC both when typing a 2-character prefix. The limited improvement of λ-TS-QAC is probably due to predictions on occasional queries as news search, whereas λ^*-TS-QAC smoothes it with cyclic phenomena for QAC ranking tasks.

5.4 Hybrid QAC ranking performance

Our research question **RQ4** aims at examining whether a user's personal query similarity helps generate better QAC rankings. We first give the absolute MRR scores of our λ^*-H-QAC in Table 4. 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 5. 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. However, despite the additional overhead of scoring similarity between queries, λ^*-H-QAC presents relatively small ($\approx$ 2%) improvements over λ^*-TS-QAC on AOL. This is due to the fact that no strongly differential features are explored yet for users.

In contrast with AOL, λ^*-H-QAC on SnV is more sensitive to user's search log with longer prefix, although AOL on most cases does have much lower QAC effectiveness than SnV, see Table 4. In part, this may be caused by following factors. Firstly, 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 completion candidates since it has a larger sample of similar behavior. Secondly, AOL is a more general search log across topics while SnV focuses on multimedia search. Thirdly, 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. Additionally, the higher performance of SnV as compared to



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

Figure 5: QAC performance in terms of MRR observed for each approach, with a query prefix p length of 1-10 characters for AOL (left) and SnV (right) query logs.

Table 6: MRR scores of G-QAC and Personalized QAC, as well as MRR changes in bracket produced by comparing G-QAC against λ^*-H-QAC (in Table 4), 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. Statistical significance of pairwise differences are detected by the t-test ($^{\blacktriangle}/^{\blacktriangledown}$ for$^{\blacktriangledown}$ α = .01, or $^{\triangle}/^{\triangledown}$ for α = .05).

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

AOL could be a consequence of the difference in user activity as $\#Qs/Us$ in Table 2 indicates SnV users submit $\backsim$20 times more queries than AOL ones.

Clearly, for both two query logs, λ^*-H-QAC is considerably more effective with a longer prefix, see Table 4 and 5. To verify this, we examine the MRR metric with a longer prefix of up to 10 characters in Fig. 5. 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 more possible completion candidates in any case on SnV.

5.5 Personalized QAC performance analysis

To help us answer **RQ5**, we compare the performance of λ^*-H-QAC with two personalized QAC scenarios (G-QAC and Personalized QAC listed in Table 1) and record the MRR scores of these two methods in Table 6. 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 6.

We find that λ^*-H-QAC significantly outperforms G-QAC and Personalized QAC on both AOL and SnV in terms of MRR scores at all cases, which again confirms the above observations in Table 4. For AOL, Personalized QAC does not work well and its MRR scores are always substantially lower than those of G-QAC, suggesting that ranking QAC candidates only according to query similarity on bigger dataset is not reliable because the number of QAC candidates is beyond control and users often issue new queries. On the contrary, Personalized QAC wins all competition against

Table 4: The effectiveness of QAC rankings produced by different forecast models on AOL and SnV in terms of MRR, with a query prefix p length of 1–5 characters. The best performer in each experiment is highlighted. The statistical significance of pairwise differences (λ-TS-QAC, λ^*-TS-QAC and λ^*-H-QAC vs. the best baseline O-MPC-R, respectively) are detected by the t-test ($^{\blacktriangle}$/$^{\blacktriangledown}$ for α = .01, or $^{\triangle}$/$^{\triangledown}$ for α = .05) and marked in the upper right hand corner of the corresponding scores; a statistically significant difference between λ^*-H-QAC and λ^*-TS-QAC is also marked in the upper left hand corner of λ^*-H-QAC scores.

		MPC-R-TW								
#p	MPC-ALL	2 Days	4 Days	7 Days	14 Days	28 Days	O-MPC-R	λ-TS-QAC	λ^*-TS-QAC	λ^*-H-QAC
					AOL					
1	0.1090	0.1093	0.1082	0.1120	0.1140	0.1147	0.1175	0.1169	0.1208	**0.1224**$^{\blacktriangle}$
2	0.1903	0.1866	0.1814	0.1938	0.1994	0.2009	0.2027	0.1982$^{\triangledown}$	0.2056$^{\triangle}$	**0.2091**$^{\blacktriangle}$
3	0.3018	0.2989	0.2902	0.3107	0.3217	0.3233	0.3267	0.3270	0.3317$^{\triangle}$	$^{\triangle}$**0.3387**$^{\blacktriangle}$
4	0.3996	0.3970	0.3875	0.4113	0.4254	0.4276	0.4318	0.4390$^{\triangle}$	0.4455$^{\blacktriangle}$	$^{\triangle}$**0.4562**$^{\blacktriangle}$
5	0.4813	0.4681	0.4593	0.4830	0.4985	0.5076	0.5087	0.5115$^{\triangle}$	0.5143$^{\blacktriangle}$	$^{\triangle}$**0.5236**$^{\blacktriangle}$
					SnV					
1	0.1573	0.2437	0.2281	0.2209	0.1953	0.1731	0.2519	0.2536	0.2637$^{\blacktriangle}$	**0.2662**$^{\blacktriangle}$
2	0.2497	0.3526	0.3349	0.3158	0.2946	0.2690	0.3607	0.3726$^{\blacktriangle}$	0.3864$^{\blacktriangle}$	**0.3907**$^{\blacktriangle}$
3	0.3281	0.4917	0.4751	0.4519	0.4318	0.3873	0.5034	0.5117$^{\triangle}$	0.5193$^{\blacktriangle}$	$^{\blacktriangle}$**0.5355**$^{\blacktriangle}$
4	0.4762	0.6096	0.5794	0.5528	0.5317	0.5167	0.6133	0.6296$^{\blacktriangle}$	0.6439$^{\blacktriangle}$	$^{\blacktriangle}$**0.6690**$^{\blacktriangle}$
5	0.5438	0.6913	0.6681	0.6327	0.6108	0.5731	0.6992	0.7103$^{\blacktriangle}$	0.7203$^{\blacktriangle}$	$^{\blacktriangle}$**0.7491**$^{\blacktriangle}$

G-QAC on SnV. This is due to: (i) users of SnV frequently issue similar queries; (ii) users of SnV submit considerable queries; (iii) queries within the same session in SnV must be similar.

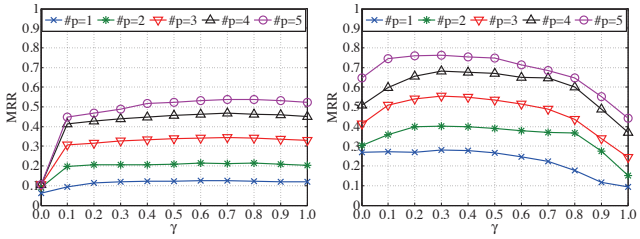
Additionally, the MRR improvement produced by λ^*-H-QAC against G-QAC are still very high, indicating that MPC-ALL in G-QAC may often eliminate useful QAC candidates. This drawback can be further exaggerated owing to the low volume of queries as the relative changes on SnV (around 15%) are larger than those on AOL (around 7%). We conclude that a small dataset suffers more from uncertainty on query popularity for ranking QAC candidates.

5.6 Effect of contribution weight γ

To answer **RQ6**, we examine the effect on the overall QAC performance by varying the contribution weight γ in (13) in our hybrid QAC model, λ^*-H-QAC, from 0 to 1 gradually tested on AOL and SnV. See Fig. 6.

For AOL, see Fig. 6a, when 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 ($0.4 < \gamma \leq 1$). When we rank QAC candidates 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 demonstrates the fact that our λ^*-H-QAC ranking model favors time-sensitive popularity than user's query similarity on AOL. This finding can be further confirmed by averaging MRR values produced under different settings: $0 \leq \gamma \leq 0.5$ and $0.5 \leq \gamma \leq 1$ for each length of prefix. Obviously, the average MRR of the latter ($0.5 \leq \gamma \leq 1$) is higher for all cases.

In contrast to AOL, the optimal γ on SnV, see Fig. 6b, makes a substantial move to around 0.3, which indicates that QAC ranking on SnV favors user's query similarity a bit more. The discrepancy between the optimal γ on SnV and the optimal γ on AOL can perhaps be explained by considering the number of issued queries of each user. Sufficient personal queries result 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 under setting $0.5 \leq \gamma \leq 1$. The overall MRR result 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.



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

Figure 6: Performance of λ^*-H-QAC when varying the combination weight γ with a query prefix p length of 1-5 characters for AOL (left) and SnV (right) query logs.

5.7 Performance of combined QAC models

To answer **RQ7**, we compare λ^*-H$_G$-QAC (in Table 1) with our λ^*-H-QAC (MRR scores reported in Table 4). The MRR scores of λ^*-H$_G$-QAC and the corresponding changes against λ^*-H-QAC tested on AOL and SnV are recorded in Table 7. 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 still wins all competitions against 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 $\approx 1\%$). This appears to be due to the fact that (i) λ^*-H$_G$-QAC completes personalized QAC on the similar character level but confronts the sparseness problem, and (ii) the number of grams n is artificially fixed, resulting in failure to rank QAC candidates properly.

6. CONCLUSION

Most previous work on query auto-completion (QAC) 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. We proposed to use time-series analysis to identify the periodicity and trend of query popularity for predicting its future frequency. We assigned an opti-

Table 7: 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 4), with a query prefix p length of 1–5 characters tested on AOL and SnV query logs. Statistical significance of pairwise differences (λ^*-H_G-QAC vs. λ^*-H-QAC) determined using the t-test ($^{\blacktriangle}/^{\blacktriangledown}$ for α = .01, or $^{\triangle}/^{\triangledown}$ for α = .05).

	AOL		SnV	
#p	MRR	MRR changes	MRR	MRR changes
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%$^{\triangledown}$

mal time window after learning to each query to find its popularity trend, which led to better prediction. To understand a user's personal search task, we extended our time-sensitive QAC method with personalized QAC, which infers the similarity between current requests and preceding queries in his current search session and previous search tasks at a char-level. We verified the effectiveness of our best performer λ^*-H-QAC on two datasets, showing significant improvements over various time-sensitive QAC baselines.

As to future work, parallel processing may enhance the efficiency of our method and other metrics can be used to evaluate the QAC rankings. Meanwhile, we aim to transfer our approach to other datasets with long-term query logs, which helps us to benefit from queries with longer periodicity than we had access to in our current work. Finally, 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 QAC ranking.

Acknowledgments. This research was supported by the Innovation Foundation of NUDT for Postgraduate under No. B130503, the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement nrs 288024 and 312827, the Netherlands Organisation for Scientific Research (NWO) under project nrs 727.011.005, 612.001.116, HOR-11-10, 640.006.013, the Center for Creation, Content and Technology (CCCT), the Dutch national program COMMIT, the ESF Research Network Program ELIAS, the Elite Network Shifts project funded by the Royal Dutch Academy of Sciences (KNAW), the Netherlands eScience Center under project number 027.012.105, the Yahoo! Faculty Research and Engagement Program, the Microsoft Research PhD program, and the HPC Fund.

REFERENCES

[1] E. Alfonseca, M. Ciaramita, and K. Hall. Gazpacho and summer rash: Lexical relationships from temporal patterns of web search queries. In *EMNLP '09*, pages 1046–1055, 2009.

[2] M. Arias, J. M. Cantera, and J. Vegas. Context-based personalization for mobile web search. In *VLDB '08*, pages 33–39, 2008.

[3] Z. Bar-Yossef and N. Kraus. Context-sensitive query auto-completion. In *WWW '11*, pages 107–116, 2011.

[4] H. Bast and I. Weber. Type less, find more: Fast autocompletion search with a succinct index. In *SIGIR '06*, pages 364–371, 2006.

[5] H. Bast, D. Majumdar, and I. Weber. Efficient interactive query expansion with complete search. In *CIKM '07*, pages 857–860, 2007.

[6] P. N. Bennett, R. W. White, W. Chu, S. T. Dumais, P. Bailey, F. Borisyuk, and X. Cui. Modeling the impact of short- and long-term behavior on search personalization. In *SIGIR '12*, pages 185–194, 2012.

[7] S. Bhatia, D. Majumdar, and P. Mitra. Query suggestions in the absence of query logs. In *SIGIR '11*, pages 795–804, 2011.

[8] S. Bickel, P. Haider, and T. Scheffer. Learning to complete sentences. In *ECML '05*, pages 497–504, 2005.

[9] F. Cai, S. Liang, and M. de Rijke. Personalized document re-ranking based on bayesian probabilistic matrix factorization. In *SIGIR '14*, 2014.

[10] H. Cao, D. Jiang, J. Pei, Q. He, Z. Liao, E. Chen, and H. Li. Context-aware query suggestion by mining click-through and session data. In *KDD '08*, pages 875–883, 2008.

[11] C. Chatfield. *The Analysis of Time Series: An Introduction*. Chapman and Hall, New York, 2004.

[12] S. Chaudhuri and R. Kaushik. Extending autocompletion to tolerate errors. In *SIGMOD '09*, pages 707–718, 2009.

[13] S. Chien and N. Immorlica. Semantic similarity between search engine queries using temporal correlation. In *WWW '05*, pages 2–11, 2005.

[14] J. Fan, H. Wu, G. Li, and L. Zhou. Suggesting topic-based query terms as you type. In *APWEB '10*, pages 61–67, 2010.

[15] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia. A survey on concept drift adaptation. *ACM Comput. Surv.*, 46(4): 44:1–44:37, March 2014.

[16] N. G. Golbandi, L. K. Katzir, Y. K. Koren, and R. L. Lempel. Expediting search trend detection via prediction of query counts. In *WSDM '13*, pages 295–304, 2013.

[17] K. Grabski and T. Scheffer. Sentence completion. In *SIGIR '04*, pages 433–439, 2004.

[18] J. Guo, X. Cheng, G. Xu, and X. Zhu. Intent-aware query similarity. In *CIKM '11*, pages 259–268, 2011.

[19] Q. He, D. Jiang, Z. Liao, S. C. H. Hoi, K. Chang, E.-P. Lim, and H. Li. Web query recommendation via sequential query prediction. In *ICDE '09*, pages 1443–1454, 2009.

[20] 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. Am. Soc. Inf. Sci. Techn.*, 61(6): 1180–1197, June 2010.

[21] A. Kulkarni, J. Teevan, K. M. Svore, and S. T. Dumais. Understanding temporal query dynamics. In *WSDM '11*, pages 167–176, 2011.

[22] 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.*, 3(1):Article 17, 2011.

[23] W. Litwin, R. Mokadem, P. Rigaux, and T. Schwarz. Fast ngram-based string search over data encoded using algebraic signatures. In *VLDB '07*, pages 207–218, 2007.

[24] N. Liu, J. Yan, S. Yan, W. Fan, and Z. Chen. Web query prediction by unifying model. In *ICDM '08*, pages 437–441, 2008.

[25] V. Michail, M. Christopher, V. Zografoula, and G. Dimitrios. Identifying similarities periodicities and bursts for online search queries. In *SIGMOD '04*, pages 131–142, 2004.

[26] T. Miyanishi and T. Sakai. Time-aware structured query suggestion. In *SIGIR '13*, pages 809–812, 2013.

[27] G. Pass, A. Chowdhury, and C. Torgeson. A picture of search. In *InfoScale '06*, 2006.

[28] R. L. T. Santos, C. Macdonald, and I. Ounis. Learning to rank query suggestions for adhoc and diversity search. *Inf. Retr.*, 16:429–451, 2013.

[29] C. Sengstock and M. Gertz. Conquer: A system for efficient context-aware query suggestions. In *WWW '11*, pages 265–268, 2011.

[30] M. Shokouhi. Detecting seasonal queries by time-series analysis. In *SIGIR '11*, pages 1171–1172, 2011.

[31] M. Shokouhi. Learning to personalize query auto-completion. In *SIGIR '13*, pages 103–112, 2013.

[32] M. Shokouhi and K. Radinsky. Time-sensitive query auto-completion. In *SIGIR '12*, pages 601–610, 2012.

[33] A. Strizhevskaya, A. Baytin, I. Galinskaya, and P. Serdyukov. Actualization of query suggestions using query logs. In *WWW '12*, pages 611–612, 2012.

[34] I. Weber and C. Castillo. The demographics of web search. In *SIGIR '10*, pages 523–530, 2010.

[35] R. W. White and G. Marchionini. Examining the effectiveness of real-time query expansion. *Inf. Proc. Man.*, 43(3):685–704, 2007.

[36] S. Whiting and J. M. Jose. Recent and robust query auto-completion. In *WWW '14*, pages 971–982, 2014.