Time-Sensitive Query Auto-Completion

Milad Shokouhi
Microsoft Research
Cambridge, UK
milads@microsoft.com

Kira Radinsky
Technion, Israel Institute of Technology
Haifa, Israel
kirar@cs.technion.ac.il

ABSTRACT

Query auto-completion (QAC) is a common feature in modern search engines. High quality QAC candidates enhance search experience by saving users time that otherwise would be spent on typing each character or word sequentially.

Current QAC methods rank suggestions according to their past popularity. However, query popularity changes over time, and the ranking of candidates must be adjusted accordingly. For instance, while *halloween* might be the right suggestion after typing *ha* in October, *harry potter* might be better any other time. Surprisingly, despite the importance of QAC as a key feature in most online search engines, its temporal dynamics have been under-studied.

In this paper, we propose a time-sensitive approach for query auto-completion. Instead of ranking candidates according to their past popularity, we apply time-series and rank candidates according their *forecasted frequencies*. Our experiments on 846K queries and their daily frequencies sampled over a period of 4.5 years show that predicting the popularity of queries solely based on their past frequency can be misleading, and the forecasts obtained by time-series modeling are substantially more reliable. Our results also suggest that modeling the temporal trends of queries can significantly improve the ranking of QAC candidates.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Information Search and Retrieval—*Search Process, selection process*

General Terms

Algorithms, Measurement, Experimentation

Keywords

Query auto-completion, temporal ranking, time-series

1. INTRODUCTION

Query auto completion (QAS) is a feature incorporated in essentially every search engine, where the goal is to save user time by predicting user's intent and suggesting possible other queries matching the first few keystrokes typed. The filtering of related candidates (query suggestions) is typically based on string matching, and the ranking is ideally based on the likelihood of the filtered candidate being the query which the user has in mind. The latter challenge (ranking) is the focus of this paper, for which the goal is to sort queries according to their expected popularity. The common practice is to use past frequencies as proxy for future popularity [2, 8]. However, those approaches assume that user intent is static and does not change over time. Simple aggregation can overshadow the temporal trends that could potentially provide valuable signals for better ordering of QAC candidates.

Consider the example in Figure 1 where a user has typed *di* in Google query box on *Sunday*, February 12th, 2012. At the first glance, knowing that *dictionary* is generally a more frequent query than *disney*, it might be difficult to notice how the ranking might be improved. However, looking at the daily trends for these queries in Figure 2 reveals that *disney* is more popular on weekends. Hence, given that the first snapshot was taken on a Sunday, swapping *disney* and *dictionary* could lead to a better ranking at the time of this query. Similar observations can be made for different granularity of time-spans. For instance, previous work [4] has shown that users are more likely to search for entertainment-related queries at night, while queries related to personal finance are more common in the morning. Therefore, the ranking of QAC candidates can vary dynamically even according to the time of the day.

In this paper, we add time-sensitivity to ranking auto-completion candidates. We consider the temporal variations of query popularity in ranking QAC suggestions, that are normally shadowed by aggregation. Rather than summarizing the entire query history in one aggregated number as expected popularity, we rely on shorter but more frequent aggregation of data, and model the overall query trends by time-series. We show that the expected popularity values produced by our time-sensitive approaches are closer approximations to what will be observed later in the logs, and are more effective for ranking QAC suggestions.

The contributions of this work are three-fold: First, we study the shadowing problem that is caused by ignoring the temporal variations of query frequency in data aggregation. We investigate several aggregation options and show that query popularity predictions based on shorter but more recent data are more accurate than those produced based on aggregation over longer periods. This simple and counter-

SIGIR'12, August 12–16, 2012, Portland, Oregon, USA.

Figure 1: Google auto-completion candidates after typing *di* on Sunday, February 13th, 2012.

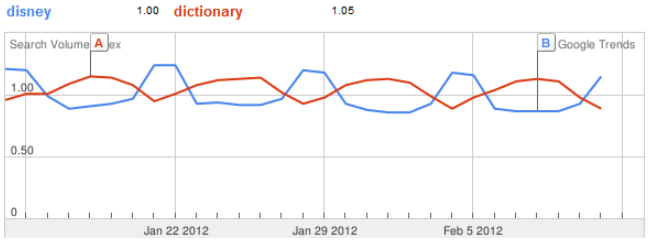

Figure 2: Daily frequencies for queries *dictionary* (red) and *disney* (blue) during January 2012 according to Google Trends (the snapshot was taken on Monday, 13-Feb-2012). Among the two queries, *disney* is more popular on weekends, while *dictionary* is issued more commonly by users on weekdays.

intuitive finding, can be useful in scenarios where applying more sophisticated time-series models is feasible. Second, we introduce a time-sensitive query auto-completion model in which the expected popularity of each query suggestion is forecasted by time-series, and dynamically varies depending on the time of query. Finally, we perform a large scale analysis over 4.5 years of sampled log data, and evaluate our predictions on daily and monthly intervals. Our experimental results indicate that accurate query popularity forecasts are essential in generating high quality QAC rankings.

2. RELATED WORK

Query auto-completion. Auto-completion during typing is a familiar feature and has been utilized in several applications ranging from early UNIX Shells to modern text editors and web browsers.

Previous work on auto-completion can be grouped into two main categories; the first group (also referred to as *predictive* auto-completion [8]) deploys information retrieval and NLP techniques to generate and rank candidates on the fly as the user enters new words and characters [13, 18, 27]. For instance, Grabski et al. [18], and Bickel et al. [6] studied sentence completion based on lexicon statistics of text collections. Fan et al. [17] ranked AC candidates according to a generative model learned by Latent Dirichlet Allocation (LDA) [7]. White and Marchionini [33] developed a real-time query expansion system that produces an updated list of candidates based on the top-ranked documents as the user types new words in the search box.

In the second group of AC techniques — including this work — candidates are pre-generated and stored in tries and hash tables for efficient *lookup*. The list of candidates is updated by new lookups with each new input from the user. The filtering of candidates in this group is typically on the basis of exact prefix matching.[1]

The ranking in *lookup* techniques is based on static scores that are assigned to candidates according to their importance. For example, in a product engine such as `amazon.com` with products names as candidates, static scores can be assigned according to popularity, price or the review scores of products [8]. In the context of web search, the most conventional approach is to rank candidates (query suggestions) according to their past popularity. In the most related study to our work, Bar-Yossef and Kraus [2] referred to this baseline as *MostPopularCompletion* (MPC) and suggested that it can be regarded as an approximate maximum likelihood estimator. The authors also proposed a context-aware technique in which the default static scores for the candidates are combined with contextual-scores based on recent session history to compute the final ranking.

We also take MPC [2] as our QAC ranking baseline and show that it can be improved significantly by considering the temporal characteristics of queries. Our work is orthogonal to all aforementioned techniques; for instance, using error-tolerant data structures that support fuzzy matching [8, 20] can expand the list of candidates that are eligible for our temporal modeling. Our work can be also combined with the *hybrid* framework of Bar-Yossef and Kraus [2] to improve the ranking of candidates with no or little context.

Time-sensitive search. The world is constantly changing and the dynamics of daily life are reflected on the web in the forms of fresh content and new information needs. Different aspects of freshness in web search have been explored; Jones and Diaz [21] and Li and Croft [24] extended the language modeling framework to incorporate the time extracted from document time-stamps. Berberich et al. [5] matched explicit temporal expressions in queries with the time-stamps mentioned in documents in a query likelihood approach for ranking. Their work was extended by Kanhabua and Nørvåg [22] that used the content of top-ranked documents for approximating the most related time-interval for the query.

Elsas and Dumais [16] showed that there is a strong connection between the rate of content changes in a document and its relevance. They proposed a probabilistic ranking model in which the term weights vary according to their temporal characteristics. Relatedly, Efron [15] investigated using linear time-series models for term weighting. Dai and Davison [11] introduced a time-sensitive version of PageRank based on multiple crawls and showed that it can be used as an effective static score for document ranking. Dong et al. [14] and Dai and Davison [12] integrated freshness in learning to rank by introducing new ways of optimizing for both freshness and relevance.

Kulkarni et al. [23] classified queries into different categories based on their change of popularity over time. The authors showed that monitoring the query popularity and content updates can reveal useful signals for detecting the change in query intent. Metzler et al. [26] classified queries with *implicit* temporal intent (e.g. halloween) according to their previous occurrences in the logs along with explicit temporal expressions (e.g. halloween 2010). Shokouhi [31]

[1]Recently, Chaudhuri and Kaushik [8] and Ji et al. [20] proposed flexible fuzzy matching models that are tolerant to small edit-distance differences between the query (prefix) and candidates.

leveraged time-series decomposition techniques for classifying seasonal queries.

Chien and Immorlica [9] demonstrated that queries with similar temporal patterns (e.g. halloween and pumpkin) can be semantically related despite no lexical overlap. Liu et al. [25] introduced a unified model for forecasting query frequency in which the forecast for each query, is influenced by the frequencies predicted for similar and correlated queries. Vlachos et al. [32] developed a compressed representation for time-series and proposed a model for detecting significant bursts in query frequencies. Baraglia et al. [3] investigated the impact of aging on *query flow graphs* and in the presence of evolving query trends. They argued that generating the graphs from scratch may not be feasible and described an incremental approach for more efficient updates. Alfonseca et al. [1] clustered queries according to their frequency time-series. They suggested that their approach can be used for query suggestion and query categorization. Zhang et al. [34] reranked documents based on their time-stamps in snippets and suggested that their approach can improve search effectiveness for temporal queries.

Radinsky et al. [29] have performed an analysis of predictability of different user behaviors (such as query clicks, URL clicks and query-URL clicks) using time-series analysis and a learning model. Similarly, Cho and Varian [10] and Shimshoni et al. [30] study the general applicability of time-series analysis for modeling query trends. However, those studies have been performed on small sets of queries (≈10,000 queries) and of short periods of time (6 months), disallowing identification of long term yearly seasonality. In our work, we perform rigorous analysis of a large scale corpus of 846K queries over a period of 4.5 years for the purpose of better query suggestions.

While it is clear that freshness has been considered in several areas of web search, to the best of our knowledge, this is the first time that the temporal trends of queries are modeled and used for QAC ranking.

3. TIME-SENSITIVE AUTO-COMPLETION

In a typical QAC scenario, the user is presented with a list of query suggestions that their prefix matches the text entered in the search box. In this paper, we refer to the latest text in the search box as *prefix* (e.g. "di" in Figure 1). The prefix is updated as the user enters new characters and will eventually match the final query once the query is submitted. For each prefix $\mathcal{P}$, the list of candidates $\mathcal{C}(\mathcal{P})$ consists of all previous queries that start with $\mathcal{P}$.[2] The list of candidates is dynamically updated at run-time with each new character typed by the user.

Ideally, the candidates should be ranked based their likelihood of matching the query that the user has in mind. In the absence of any contextual information, this likelihood can be set according to the wisdom of crowds by using the general frequency in the past as a proxy for the *expected* popularity in future. The most common approach is to *aggregate* the query frequencies over a query log, and use these *aggregated* values for ranking QAC suggestions. Bar-Yossef and Kraus [2] referred to this general form of QAC ranking as *MostPopularCompletion* (MPC) where,

$$MCP(\mathcal{P}) = \arg\max_{q \in \mathcal{C}(\mathcal{P})} w(q), \quad w(q) = \frac{f(q)}{\sum_{i \in \mathcal{Q}} f(i)} \qquad (1)$$

here, $f(q)$ denotes the number of times the query q occurs in a previous search log $\mathcal{Q}$. Under the MPC model, the candidate scores do not change as long as the same query log $\mathcal{Q}$ is used. Each query is represented by a single value as its popularity which is computed by counting the number of times that it has occurred in the past. As long as $\mathcal{Q}$ is not replaced by a fresher log, the assumption is that the aggregated frequencies can be regarded as reasonable approximations for future popularity. However, as mentioned earlier, the query popularity may change over time and the candidate lists must be adjusted consequently to account for recent changes.

While frequent updates of query statistics may address the freshness problem to some extent, it is not clear how often the logs must be updated to provide the best approximations for future popularity. Without the right update policy, trend effects will be disregarded. For example, a query such as *Sarah Burke* that gained high popularity in January 2012, but was not queried as often in the past, might get lower ranking if compared to the query *Sarah Palin*, which has high volume, since it was queried for many years, despite being relatively less popular in January 2012. Furthermore, aggregation methods ignore seasonal effects. For instance, at the time of this writing on 13th February 2012 – a day before the *Valentine's day* – Google suggests *verizon wireless* as top candidate for *v* and does not suggest any Valentine's related candidate in the QAC ranking (the top plot in Figure 3). The query frequency trends in the bottom plot of Figure 3 however clearly show that *valentines day* is temporally more relevant for suggestion. As another example, weekly patterns such as those shown in Figure 2 for *disney* and *dictionary* disappear in monthly aggregations. In summary, the past is not always a good proxy for future particularly for trendy and seasonal queries. Today's frequency for query *dictionary* is not necessarily the best estimate for its frequency tomorrow.

We propose TS, a time-sensitive QAC ranking model in which the default *aggregated* candidate scores in Equation 1 are replaced with *forecasted* values computed by time-series modeling of query history. Here, the score of each candidate at time t is determined according to its predicted value calculated using time-series models that capture trend, seasonality. Hence, our time-sensitive QAC ranking model can be formalized as,

$$TS(\mathcal{P}, t) = \arg\max_{q \in \mathcal{C}(\mathcal{P})} w(q|t), \quad w(q|t) = \frac{\hat{y}_t(q)}{\sum_{i \in \mathcal{Q}} \hat{y}_t(i)} \qquad (2)$$

where $\mathcal{P}$ is the entered prefix, and $\mathcal{C}(\mathcal{P})$ represents its list of QAC candidates, and $\hat{y}_t(q)$ denotes the estimated frequency of query q at time t. As a result, the quality of a candidate list generated by our TS model is directly influenced by the accuracy of predictions obtained by time-series modeling. Note that, our time-sensitive model becomes similar to the context-sensitive QAC ranking of Bar-Yossef and Kraus [2] if one considers time as "context". The focus of our work is on the time-sensitivity of query suggestions, and on how we can enhance QAC to handle this aspect. We leverage techniques from time-series analysis to model temporal query behavior

[2]Without the loss of generality we ignore more advanced fuzzy matching techniques [8, 20] in our work.

Figure 3: (Top) The auto-completion candidates ranked by Google on 13th February 2012, a day before Valentine's. (Bottom) The query frequencies for *valentines day* vs. *verizon wireless* since 2004.

(Section 3.1), and demonstrate how these models can be learnt from query logs to do better QAC ranking (Section 3.2).

3.1 Time-Series Analysis & Forecast

A *time-series* consists of a sequence of data points in successive time order and with uniform intervals. Hence, annual salaries, the daily rate of CO2 emissions and any other sequence of numbers collected at uniform intervals can be represented by a time-series. In practice, time-series analysis is often used to model the temporal changes in data and to forecast future trends.

At any given time, the *Exponential smoothing* methods [19] apply a weighted average over the frequencies of previous data points to forecast the upcoming trends. In the simplest form — also known as *Single exponential smoothing* — the time-series is smoothed as follows,

$$\bar{y}_t = \lambda y_t + (1-\lambda)\bar{y}_{t-1} \tag{3}$$

where y_t and $\bar{y}_t$ respectively denote the actual and smoothed values for the data at time t, and λ is the *smoothing parameter* ranging between 0 and 1. The recursive equation above can be also used to provide the forecast for the next interval ($\hat{y}_{t+1}$). That is,

$$\hat{y}_{t+1} = \bar{y}_t \tag{4}$$

While single exponential smoothing may produce reasonable forecasts for stationary time-series, it performs poorly in the presence of a *trend* in time-series. To remedy this problem, *Double exponential smoothing* methods extend the previous model by introducing a trend variable F_t,

$$\bar{y}_t = \lambda_1 y_t + (1-\lambda_1)(\bar{y}_{t-1} + F_{t-1}) \tag{5}$$
$$F_t = \lambda_2(\bar{y}_t - \bar{y}_{t-1}) + (1-\lambda_2)F_{t-1} \tag{6}$$

Here, parameter F_t models the linear trend of time-series at time t, while λ_1 and λ_2 are smoothing parameters. As in previous equations, y_t and $\bar{y}_t$ represent the actual and smoothed values at time t. The forecast for the next interval based on double exponential smoothing can be obtained by:

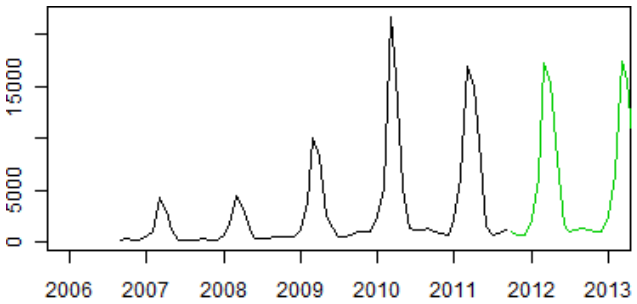

Figure 4: The black line shows monthly frequency values for query *spring flowers* between Sep'06–Jun'11. The green curve depicts the predicted values for the next 24 months (Jul'11–Jun'13) based on triple exponential smoothing. Data from `bing.com`.

$$\hat{y}_{t+1} = \bar{y}_t + F_t \tag{7}$$

That is, the value of the next data point at time $t+1$ is predicted by linear combination of the smoothed value at time t and the latest trend parameter F_t. Hence, a negative F_t sets a smaller predicted value for the next time interval $(t+1)$, compared to the most recently observed value y_t.

Double exponential smoothing does not capture the frequency variations due to potential seasonality (periodicity) in the data. *Triple exponential smoothing* generalizes the previous techniques by explicitly modeling the seasonality. The updated model — also known as HoltWinters smoothing — is often expressed as:

$$\bar{y}_t = \lambda_1(y_t - S_{t-\tau}) + (1-\lambda_1)(\bar{y}_{t-1} + F_{t-1}) \tag{8}$$
$$F_t = \lambda_2(\bar{y}_t - \bar{y}_{t-1}) + (1-\lambda_2)F_{t-1} \tag{9}$$
$$S_t = \lambda_3(y_t - \bar{y}_t) + (1-\lambda_3)S_{t-\tau} \tag{10}$$
$$\lambda_1 + \lambda_2 + \lambda_3 = 1 \tag{11}$$

where S_t captures the seasonality of data at time t, and τ specifies the length of periodic cycle (e.g. 12 for annual cycles when the data points represent monthly frequencies). The smoothing parameters λ_1, λ_2, and λ_3 vary between 0 and 1 and can be optimized using standard techniques such as maximum likelihood. The forecast for the value of the time-series at the next interval is given by:

$$\hat{y}_{t+1} = (\bar{y}_t + F_t)S_{t+1-\tau} \tag{12}$$

Figure 4 illustrates an example of applying triple exponential smoothing for forecasting future trends. The underlying data comprises of monthly frequency values for query *spring flowers* sampled from logs of `bing.com` between September 2006 and June 2011. The green curve shows the forecast for July 2011 – June 2013 ($\tau = 12$).

3.2 Parameter Estimation and Forecasting

In this work we utilize a hill-climbing optimization technique called limited-memory BFGS (L-BFGS), which is a limited memory variation of the Broyden-Fletcher-Goldfarb-Shanno (BFGS). The technique estimated the direction by

solving the following Newton equation:

$$B_k \mathbf{p}_k = -\nabla f(\mathbf{x}_k) \tag{13}$$

where B_k is the approximation of the Hessian matrix at step k, and $f = \sum \epsilon^2$ is the sum of the prediction error rates. The initial values B_0 and x_0 are guessed, and a line search in the direction at step k is then used to find the next point x_{k+1}, and a new value of the matrix B_{k+1} is estimated [28].

4. EVALUATION

Our first set of experiments focuses on forecast accuracy while in the second part we evaluate the effectiveness of QAC rankings. For each task, we borrow a couple of metrics from information retrieval and statistics for measurement.

Forecast accuracy. *Mean absolute error* (MAE) is widely used in statistics to measure the accuracy of forecasts and is defined as follows,

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |\hat{y}_i - y_i| \tag{14}$$

where y is the true value and $\hat{y}$ is the prediction. MAE is an unbounded measure and is not strongly resilient to outliers. Therefore, its is often used along with other metrics such as *Symmetric mean absolute percentage error* (SMAPE) to diagnose the forecast variation. SMAPE is defined as,

$$SMAPE = \frac{1}{n}\sum_{i=1}^{n} \frac{|\hat{y}_i - y_i|}{\hat{y}_i + y_i} \tag{15}$$

In contrast to mean absolute error, SMAPE is bounded between 0 and 1, although it penalizes under- and over-estimations unequally.

QAC quality. In the absence of any contextual and personalized information, the ground-truth QAC ranking for a prefix $\mathcal{P}$ is the list of all queries that start with $\mathcal{P}$ (or are somehow filtered for the prefix) ordered according to their *true* popularity. At any given time t, the true popularity values are set according to the observed query frequency values at that time. Obviously, this information is not available to QAC ranking models at runtime, and they have to rely on historical data at time $t-1$ and before to rank candidates as close as possible to this unobserved ground-truth ($\mathcal{G}$).

We measure the quality of QAC lists by computing their *Spearman correlation* (ρ) against this gold standard $\mathcal{G}$. We also employ MRR to specifically measure the mean reciprocal rank of the *best* ground-truth candidate (top-ranked in $\mathcal{G}$) in QAC rankings.

Small differences in query frequency can cause significant changes in the ordering of QAC candidates. Both MRR and Spearman correlation penalize swapping two queries with similar frequencies the same way they penalize swapping a pair with a substantial gap in popularity. Consider a scenario where the user has typed *hall* in the search box in October, and the QAC model has to rank the following candidates: *halloween*, *halloween party games*, and *halloween party ideas*. The first candidate is orders of magnitude more popular than the other two which more or less have similar frequencies. Clearly, pairwise swappings that involve the first candidate should be rewarded or penalized more heavily than the pairwise swapping of the other pair.

To make our evaluation more robust against the potential noise caused by insignificant differences, we first take a log of — true/predicted — query frequencies and then round them to the closest integer across all our QAC ranking evaluations.

5. DATA

Our experimental data comprises of all queries that were submitted to `bing.com` by users – with a US-based IP address – between January 1st, 2007 and June 30th, 2011. We dropped queries that never appeared more than 28 times in at least one of the months during this period. In total 846,432 queries and their daily frequencies were extracted.

The quality of QAC candidates can vary depending on the amount of historical data available for each query. The effectiveness of time-series techniques for modeling the query trends is also sensitive to the amount of training data and the length of seasonal cycle (τ in Equation 12). Therefore, we use our sampled query statistics to construct two testbeds with distinctive characteristics.

Daily buckets, weekly cycles (D-W). Our first testbed is generated from daily query frequency values between January 2nd, 2011, and June 30th, 2011 (180 days). For each query at time (day) t, we use the frequencies at all previous intervals (previous daily frequencies) to fit the time-series models. We consider the last 30 days (June 2011) as our testing period for reporting the results. We set the τ value in time-series models to 7 which would allow them to capture potential weekly cycles in query frequency.

Monthly buckets, annual cycles (M-A). We aggregated the daily frequency values between January 2007 and June 2011 in monthly buckets . Hence, the history of each query is represented by 54 data points one for each month covered during our sampling time-frame. We report our evaluation results on the first 6 months of 2011. Here, we set the τ parameter in our time-series models to 12, to allow modeling of potential annual cycles.

Compared to the previous testbed, there are fewer data points for time-series modeling of query trends, and there is less variance and sparsity in query frequencies.

6. PREDICTING QUERY POPULARITY

In an oracle list of QAC suggestions, candidates are sorted in descending order of their *true* popularity. Since this ground-truth information is unavailable at runtime, QAC ranking models order candidates according to their *expected* (predicted) popularity inferred from previous logs.

Our TS auto-completion method models the entire query history by time-series and forecasts the future popularity accordingly. The MCP method [2] that we use as our baseline assumes that the aggregated frequency of a query over past search logs is a reasonable approximation for its future popularity. Surprisingly, little is known about the validity of this assumption and its impact on the ranking of QAC candidates. Furthermore, the trade-off between the recency (and size) of the query log used for aggregation, and the accuracy of outcome predictions has not been investigated.

In this section, we evaluate the accuracy of various methods for predicting the future query popularity, and in the

Table 1: The forecast error rates obtained by different methods on the D-W testbed. The query frequencies are predicted once for each of the 30 days in the testing period (June 2011). The best performing method in each row is specified by an underline. All (TS vs. P_*) and (TMS vs. P_*) pairwise differences are detected as statistically significant by the t-test ($p < 0.01$).

	P_1	P_3	P_6	P_{12}	P_h	TS	TMS
MAE	14.02	23.37	23.62	24.05	23.12	10.57	10.50
SMAPE	0.277	0.272	0.273	0.275	0.270	0.228	0.251

Table 2: The forecast error rates obtained by different methods on the M-A testbed. The query frequencies are forecasted once for each of the 6 months in the testing period (Jan–Jun 2011). The best performing method in each row is specified by an underline. Except for (MAE, $k = 1$), all (TS vs. P_*) and (TMS vs. P_*) pairwise differences are detected as statistically significant by the t-test ($p < 0.01$).

	P_1	P_3	P_6	P_{12}	P_h	TS	TMS
MAE	338	1065	1108	1189	1012	343	323
SMAPE	0.168	0.330	0.346	0.381	0.313	0.179	0.164

next section we measure the impact of these predictions on the quality of QAC rankings.

Forecast quality. MCP considers the past query frequency as expected popularity for future. For this, the query frequencies are aggregated over a past query log. We explore several aggregation options by averaging the query frequencies over the last k intervals ($k \in \{1, 3, 6, 12\}$), and also over the entire query history. We refer to latter form of aggregation as P_h and denote the others by P_k where k is the number of previous intervals used for averaging. We study the accuracy of these aggregated baselines in predicting future query popularity, and compare them against the forecasts produced by our time-series modeling of query trends (TS).

Table 1 includes the forecast error rates of different methods on the D-W testbed. The frequency of each query is forecasted once for each of the 30 days in our testing period (June 2011). At each time t, the P_k baselines use the average frequency of the past k days as their prediction. P_h computes the average over the entire query history before t. The TS model also uses the entire previous history but produces its forecasts by triple exponential smoothing ($\tau = 7$). For now, ignore the TMS column as we will get to it later. The numbers show that our TS predictions are better than all aggregated baselines on both metrics; all differences are detected as statistically significant by the t-test ($p < 0.01$). Among the aggregated baselines, MAE favors P_1 while SMAPE picks P_h as the best model. The discrepancy suggests that averaging over the entire history provides more robust predictions overall but may fail more noticeably on outliers.

We take a closer look at the daily error rates produced by the best three methods (P_1, P_h, and TS) in Figure 5. The results are consistent with the overall numbers; the MAE rates have higher variance and P_h is more robust than P_1 although it may produce greater absolute errors. P_h performs worst over the weekends and P_1 errors are highest on Saturdays and Mondays. This can be explained by the fact that search traffic is generally lower on the weekends, and grows again on Mondays. P_1 is constantly one step behind to react to these changes and P_h is too stale in general.

We repeated our analysis on the M-A testbed and summarized the results in Table 2. All methods produce 6 forecasts for each query, one for every month in our testing period Jan–June 2011. The P_k baselines use the average frequency of the past k months as their forecast while P_h computes the average over the entire query history since January 2007. The TS model also uses the entire previous history but produces its forecasts by triple exponential smoothing ($\tau = 12$). Once again, ignore the values under the TMS column as

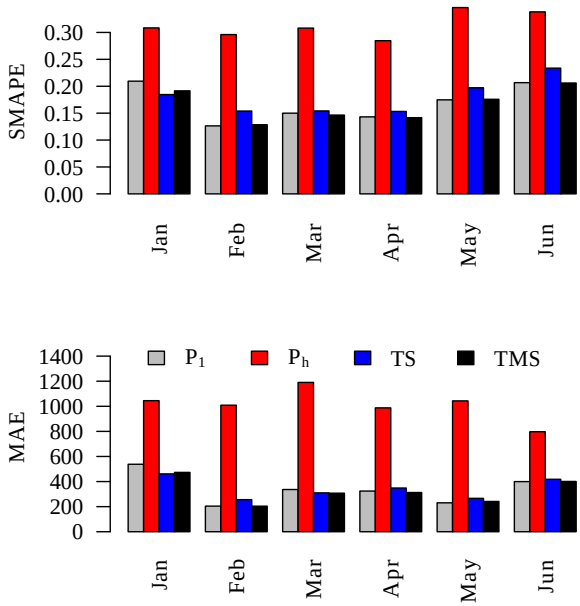

Figure 6: Daily SMAPE (top) and MAE (bottom) rates for P_1, P_h, TS and TMS predictions on the M-A testbed (January–June 2011).

they will be described later. The numbers show that with the exception of P_1, our TS predictions are better than all aggregated baselines on both metrics; all differences are detected as statistically significant by the t-test ($p < 0.01$). The difference between P_1 and TS is statistically significant on SMAPE but not so according to MAE. The competitive performance of P_1 on this testbed can be explained by several reasons; compared to the daily frequency values used the in D-W testbed, the data here is aggregated in monthly partitions. The monthly query frequencies are less sparse and have lower variance. Furthermore, the TS model has about 6 times more data points in the D-W testbed which allows it to fit the query trends better.

Figure 6 illustrates the monthly error rates for different methods. While the trends are consistent with the overall numbers, it is interesting to note the higher error rate of P_1 for January which is due to the change of year and end of holiday season.

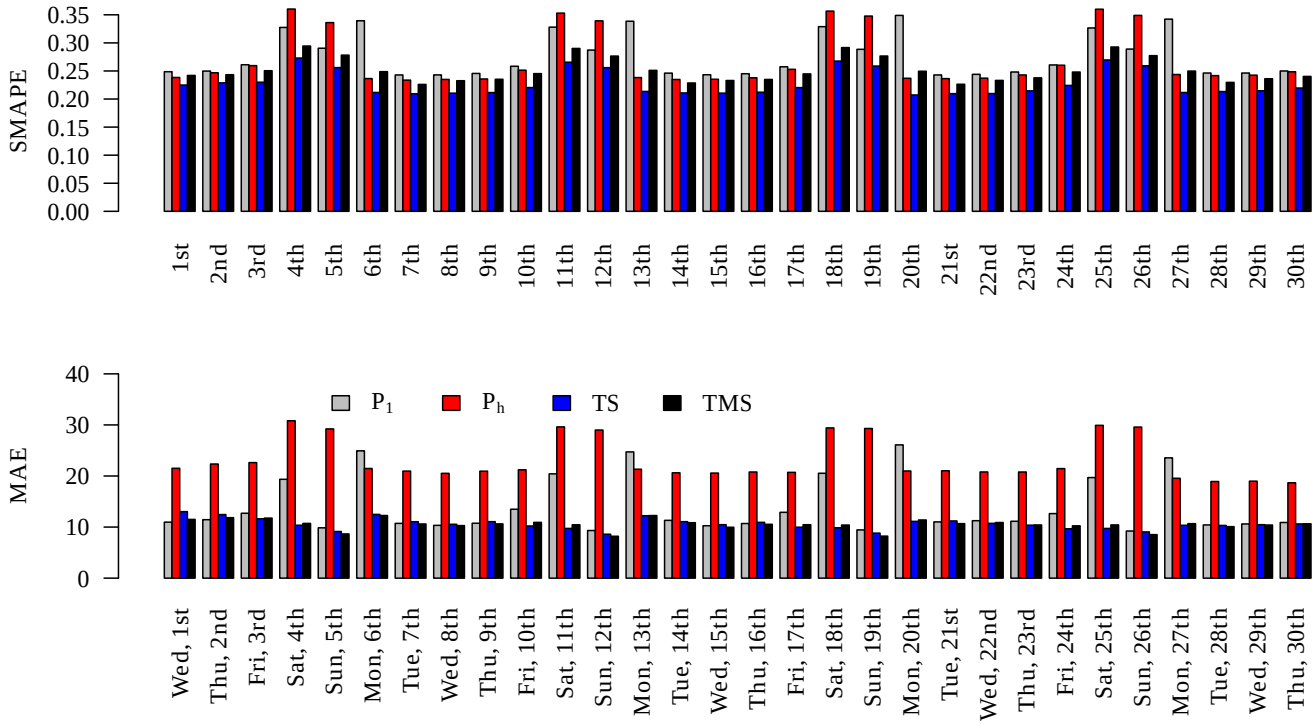

Figure 5: Daily SMAPE (top) and MAE (bottom) rates for forecast models on the D-W testbed (June 2011).

Success & Failure Analysis. We picked P_1 and TS as the two most effective methods according to the overall metrics and inspected their performance on per-query basis.

On the D-W testbed, lower error rates for TS versus P_1 are most apparent in queries with highly periodic frequency trends. Figure 7 depicts the actual and predicted frequencies for queries *nascar* and *irish lottery* as two examples from this category. The P_1 gains over TS often come from queries that were either periodic and highly popular but suddenly lost their demand (e.g. recently ended tv shows), or those that face an unexpected spike in popularity (e.g. celebrity names). Figure 8 shows two of these examples for queries *american idol* and *ginger lee*.

We also grouped the TS *weekday winners* by extracting the queries that their TS predictions are better than P_1 on those weekdays, consistently over the entire testing period (June 2011). The top winners for each day are listed in Table 3.[3] As expected all these queries have periodic trends and are related to intents that have weekly cycles. The list is dominated by queries related to lottery (e.g. euromillion), tv shows (e.g. zero punctuation) and those targeting weekly deals (e.g. dominos) or updates (e.g. week in pictures). There were no *weekday losers* apart from queries that were related to *americal idol* and *dancing with the stars*, the two tv shows that their latest series ended in May 2011, just before the start of our testing time period.

Similar patterns can be observed on the M-A testbed. TS is more effective in predicting the frequency of queries with seasonal trends, while it fails to adapt quickly when there is a sudden change in the query popularity. Figures 9 and 10 contain four examples of significant wins and losses for TS versus P_1.

[3]For presentation purposes, we excluded queries that have high overlap with those already included in the list. For example, *abc family* is dropped when *abc* is already included.

Table 3: A sample of queries where TS performs substantially better than P_1 consistently on certain weekdays during the testing period (June 2011).

Mon	dear prudence; fedex tracking; gmail; ···
Tue	euromillion; abc; daily show; dominos ···
Wed	mega millions; publix; zero punctuation ···
Thu	louisiana lottery; footy tips; olg; ···
Fri	week in pictures; nascar; cinemex ···
Sat	irish lottery; wclc; toto; ···
Sun	wisconsin unemployment; mail on sunday; ncesc; ···

Temporal model selector (TMS). Our results so far have shown that when it comes to query frequency forecast models, there is no one size fits all method that always wins. For queries with an unexpected spike in particular, the time-series models tend to overfit when trying to optimize their *smoothing* parameters. Motivated by this observation, we propose a *temporal model selector* that dynamically decides which forecast model to choose at each time-interval for the query. For this purpose, we consider a segment of query history as the *validation period* and decide between different models according to their performance on this segment. The validation process can be designed in different ways. For instance, one can pick the model that has the lowest error rate on the entire validation set. Alternatively, it is possible to train a classifier that takes in monthly error rates and other features for supervised model selection. We decided to take a simple approach and left more sophisticated

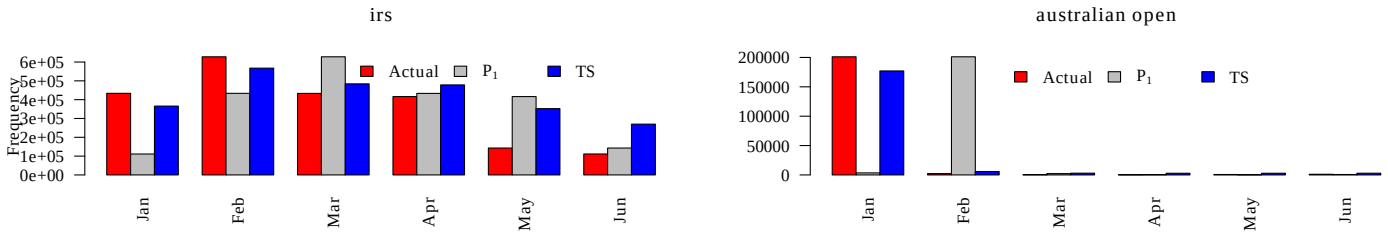

Figure 9: (left) The actual daily frequencies and the predicted values by P_1 (SMAPE ≈ 0.26) and TS (SMAPE ≈ 0.18) for query *irs* between Jan–Jun 2011. (right) Same data for query *australian open* by P_1 (MAE ≈ 0.52), and TS (SMAPE $\approx$ 0.49). TS is more successful in predicting the frequency of queries with seasonal trends.

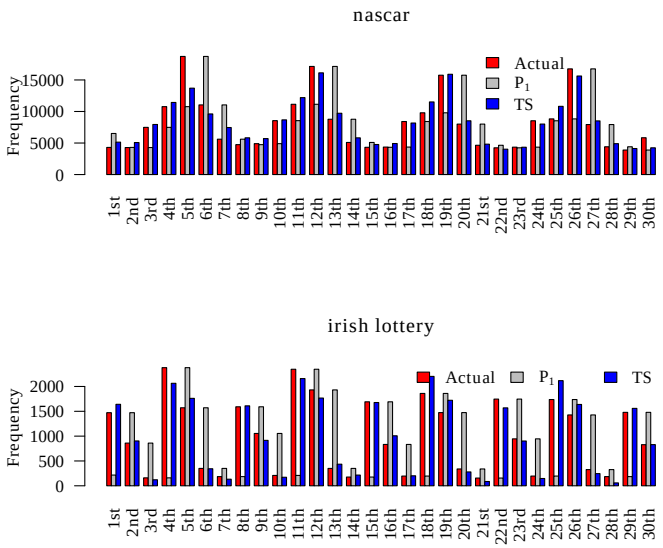

Figure 7: (Top) The actual daily frequencies and the predicted values by P_1 (SMAPE ≈ 0.19) and TS (SMAPE $\approx$ 0.06) for query *nascar* in June 2011. (Bottom) Same data for query *irish lottery* by P_1 (SMAPE ≈ 0.52), and TS (MAE $\approx$ 0.09). TS is more successful in modeling the periodic trends.

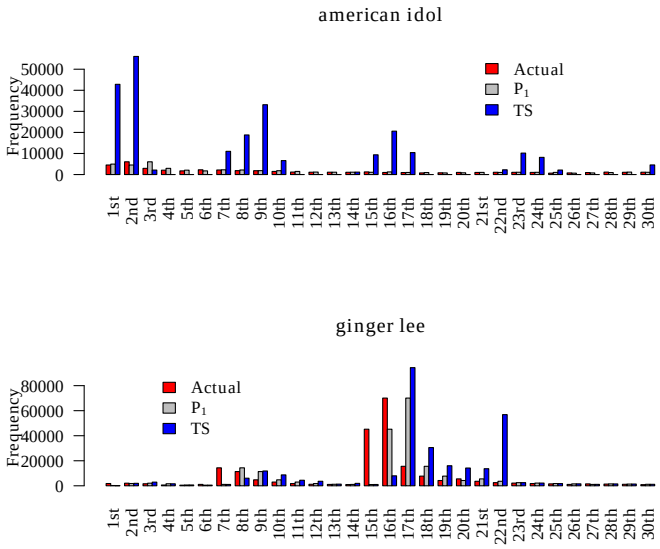

Figure 8: (Top) The actual daily frequencies and the predicted values by P_1 (MAE ≈ 0.08) and TS (MAE ≈ 0.81) for query *american idol* in June 2011. (Bottom) Same data for query *ginger lee* by P_1 (SMAPE ≈ 0.26), and TS (SMAPE $\approx$ 0.40). P_1 reacts to quick changes of popularity more effectively.

models for future work. At each time interval t, we test the performance of — P_1 and MS — forecast models at times $t-\tau, t-2\tau, ..., t-k\tau$ as long as they are included in the validation segment. We then pick the model that wins more often on the selected months. We break any tie by backing off to the model that has lower SMAPE on the entire validation set. In nutshell, the validation step reduces the risk of using time-series forecast models that overfit their smoothing parameters to fit an unexpected recent burst. Note that the corrections achieved by model selection still cannot cover the opposite cases such as *american idol* where the popularity suddenly drops.

On the D-W testbed, we used the daily frequencies in May 2011, and on the M-A testbed we used the monthly values between 2009–2010 as our validation sets. The overall error rates obtained by TMS on these testbeds can be respectively found in Tables 1 and 2. On the M-A testbed, temporal selection boosts both metrics. On the D-W testbed TMS leads to further improvements in MAE but it negatively affects SMAPE. The daily and monthly error rates in Figures 5 and 6 are consistent with the overall numbers.

In summary, our experiments suggest that despite the common assumption, aggregated query frequencies may not provide a good proxy for future popularity. Although aggregation based on fresher data is generally more effective, it still fails to predict the popularity of periodic queries. Our forecasts based on time-series analysis, and temporal model selection consistently outperform the aggregated baselines on both testbeds. In the next section, we show how improvements in forecast accuracy leads to better QAC ranking.

7. TIME-SENSITIVE QAC RANKING

We pick P_1 as the best aggregated model to form the best case for *MostPopularCompletion* (MCP) [2] baseline in our QAC experiments. At each time t (day/month), we rank the QAC candidates according to their frequency at time $t-1$ and compare them against the oracle ranking generated by the true frequency values at time t. We also use the forecasts produced by the TS and TMS models to generate two time-sensitive QAC rankings for each query.

QAC Candidates. We extract the QAC prefixes and candidates from the same query log described in Section 4. For each prefix $\mathcal{P}$ at time t, we match all the queries that start with $\mathcal{P}$ and rank them according to their true frequency at time t to generate the ground-truth ranking. We filter out all prefixes that have fewer than 3 characters, and those that have less than 5 candidates. For each forecast method at time t, we rank the top-20 ground-truth candidates based on the data available at time $t-1$ and before. Those ground-truth candidates that are observed for the first time in the logs at time t and did not exist before are omitted.

Results. Table 4 contains the evaluation results for ranking QAC candidates based on different prediction models. On the D-W testbed 30 rankings are generated for each prefix, one for each day in June 2011. Similarly, on the M-A testbed, each prefix is used to generate 6 QAC rankings, one for each month during January–June 2011. The numbers in the table are averaged across all queries and over the entire testing period. All pairwise differences are detected as statistically significant for both evaluation metrics.

The MRR and Spearman (ρ) results in Table 4 closely follow patterns to observed for SMAPE error rates in our previous experiments. On the D-W testbed in Table 5 we saw that TS had lower error rates than TMS and they both produced significantly more accurate predictions compared to P_1. The same can be observed when comparing the QAC lists generated based on their forecast values. TS has the edge over TMS while P_1 performs significantly poorer than both. Similarly, the SMAPE numbers in Table 6 suggested that TMS produces the most accurate forecasts on the M-A testbed. Here the results in Table 4 confirm that indeed these more accurate forecasts lead to higher quality QAC rankings.

We also computed the Pearson coefficient (r) between the average forecast error rates of the top five QAC suggestions and the final ρ and MRR values computed for those rankings. As expected, the Pearson coefficient suggests a negative correlation between the quality of QAC rankings and the average forecast errors of the top five candidates ($r \approx -0.17$ for SMAPE-Spearman and $r \approx -0.21$ for SMAPE-MRR).

Once again, time-series modeling is a key factor for better QAC ranking. Time-sensitive models such as TS and TMS are able to choose between *big east conference* and *big east tournament*, or between *when to plant tulips* and *when to plant tomatoes* depending on the time of query, while methods based on aggregated data constantly fall behind.

Table 4: The effectiveness of QAC rankings produced according to different forecast models on the D-W (left) and M-A (right) testbeds. The best performing method in each experiment is specified by an underline. All pairwise differences are detected as statistically significant by the t-test ($p < 0.01$).

	D-W Testbed		M-A Testbed	
	Spearman (ρ)	MRR	Spearman (ρ)	MRR
MCP(P_1)	0.569	0.763	0.607	0.867
MCP(TS)	<u>0.623</u>	<u>0.803</u>	0.597	0.868
MCP(TMS)	0.611	0.796	<u>0.618</u>	<u>0.873</u>

8. CONCLUSIONS

We proposed a new time-sensitive query auto-completion model in which the ranking of query suggestions varies according to their predicted popularity at the time of query.

We leveraged time-series techniques to predict the query popularity and showed that such methods consistently outperform different baselines that use aggregated data. We also demonstrated that predictions based on aggregated frequency over a long period are usually worse than those made on smaller but more recent data. Finally, we showed that using more accurate query popularity predictions produced by time-series modeling leads to higher quality auto-completion query suggestions.

As feature work, a mixture model produced by fitting several time-series models simultaneously (with different values of τ) may improve the quality of predictions. Last but not least, similar analysis can be done for different levels of granularity such as hourly intervals.

9. REFERENCES

[1] E. Alfonseca, M. Ciaramita, and K. Hall. Gazpacho and summer rash: lexical relationships from temporal patterns of web search queries. In *Proc. Conf. Empirical Methods in NLP*, pages 1046–1055, Singapore, 2009.

[2] Z. Bar-Yossef and N. Kraus. Context-sensitive query auto-completion. In *Proc. WWW*, pages 107–116, Hyderabad, India, 2011.

[3] R. Baraglia, F. M. Nardini, C. Castillo, R. Perego, D. Donato, and F. Silvestri. The effects of time on query flow graph-based models for query suggestion. In *Proc. RIAO*, pages 182–189, Paris, France, 2010.

[4] S. M. Beitzel, E. C. Jensen, A. Chowdhury, D. Grossman, and O. Frieder. Hourly analysis of a very large topically categorized web query log. In *Proc. SIGIR*, pages 321–328, Sheffield, United Kingdom, 2004.

[5] K. Berberich, S. Bedathur, O. Alonso, and G. Weikum. A language modeling approach for temporal information needs. In *Proc. ECIR*, pages 13–25, Milton Keynes, UK, 2010.

[6] S. Bickel, P. Haider, and T. Scheffer. Learning to complete sentences. In *Proc. ECML*, pages 497–504. 2005.

[7] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, Mar. 2003.

[8] S. Chaudhuri and R. Kaushik. Extending autocompletion to tolerate errors. In *Proc. SIGMOD*, pages 707–718, Providence, Rhode Island, USA, 2009.

[9] S. Chien and N. Immorlica. Semantic similarity between search engine queries using temporal correlation. In *Proc. WWW*, pages 2–11, Chiba, Japan, 2005.

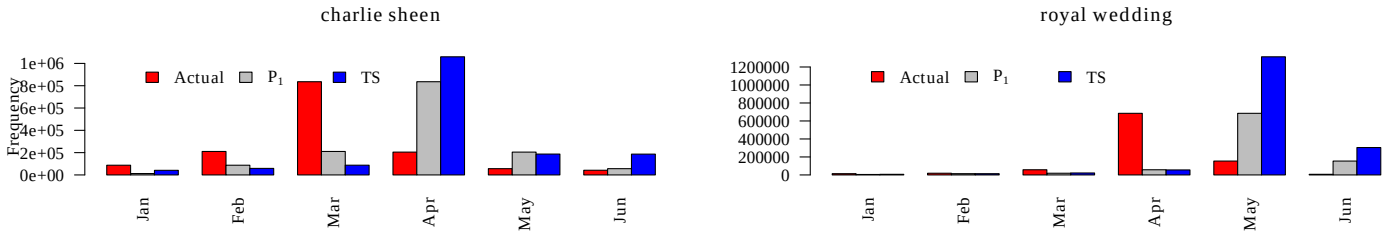

Figure 10: (left) The actual daily frequencies and the predicted values by P_1 (SMAPE ≈ 0.51) and TS (SMAPE ≈ 0.60) for query *charlie sheen* between Jan–Jun 2011. (right) Same data for query *royal wedding* by P_1 (MAE ≈ 0.58), and TS (SMAPE $\approx$ 0.59). P_1 reacts to quick changes of popularity more effectively.

[10] H. Cho and H. Varian. Predicting the present with google trends. Technical report, Google Inc, April 2009.

[11] N. Dai and B. D. Davison. Freshness matters: in flowers, food, and web authority. In *Proc. SIGIR*, pages 114–121, Geneva, Switzerland, 2010.

[12] N. Dai, M. Shokouhi, and B. D. Davison. Learning to rank for freshness and relevance. In *Proc. SIGIR*, pages 95–104, Beijing, China, 2011.

[13] J. J. Darragh, I. H. Witten, and M. L. James. The reactive keyboard: A predictive typing aid. *Computer*, 23:41–49, November 1990.

[14] A. Dong, R. Zhang, P. Kolari, J. Bai, F. Diaz, Y. Chang, Z. Zheng, and H. Zha. Time is of the essence: improving recency ranking using twitter data. In *Proc. WWW*, pages 331–340, Raleigh, North Carolina, USA, 2010.

[15] M. Efron. Linear time series models for term weighting in information retrieval. *J. Am. Soc. Inf. Sci. Technol.*, 61, July 2010.

[16] J. L. Elsas and S. T. Dumais. Leveraging temporal dynamics of document content in relevance ranking. In *Proc. WSDM*, 2010.

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

[18] K. Grabski and T. Scheffer. Sentence completion. In *Proc. SIGIR*, pages 433–439, Sheffield, United Kingdom, 2004.

[19] C. C. Holt. Forecasting seasonals and trends by exponentially weighted moving averages. *International Journal of Forecasting*, 20(1):5–10, 2004.

[20] S. Ji, G. Li, C. Li, and J. Feng. Efficient interactive fuzzy keyword search. In *Proc. WWW*, pages 371–380, Madrid, Spain, 2009.

[21] R. Jones and F. Diaz. Temporal profiles of queries. *ACM Trans. Inf. Syst.*, 25, July 2007.

[22] N. Kanhabua and K. Nørvåg. Determining time of queries for re-ranking search results. In *Proc. ECDL*, 2010.

[23] A. Kulkarni, J. Teevan, K. M. Svore, and S. T. Dumais. Understanding temporal query dynamics. In *Proc. WSDM*, pages 167–176, Hong Kong, China, 2011.

[24] X. Li and W. B. Croft. Time-based language models. In *cikm*, pages 469–475, 2003.

[25] N. Liu, J. Yan, S. Yan, W. Fan, and Z. Chen. Web query prediction by unifying model. In *Proc. IEEE Int. Conf. Data Mining Workshops*, pages 436–441, Washington, DC, USA, 2008.

[26] D. Metzler, R. Jones, F. Peng, and R. Zhang. Improving search relevance for implicitly temporal queries. In *Proc. SIGIR*, pages 700–701, Boston, MA, USA, 2009.

[27] A. Nandi and H. V. Jagadish. Effective phrase prediction. In *Proc. VLDB*, pages 219–230, Vienna, Austria, 2007.

[28] P. L. R. Byrd and J. Nocedal. *SIAM Journal on Scientific and Statistical Computing*, 16(5):1190–1208, 1995.

[29] K. Radinsky, K. M. Svore, J. Teevan, S. T. Dumais, A. Bocharov, and E. Horvitz. Learning and predicting behavioral dynamics on the web. In *Proc. of WWW*, pages to–appear, 2011.

[30] Y. Shimshoni, N. Efron, and Y. Matias. On the predictability of search trends. Technical report, Google Inc, August 2009.

[31] M. Shokouhi. Detecting seasonal queries by time-series analysis. In *Proc. SIGIR*, pages 1171–1172, Beijing, China, 2011.

[32] M. Vlachos, C. Meek, Z. Vagena, and D. Gunopulos. Identifying similarities, periodicities and bursts for online search queries. pages 131–142, Paris, France, 2004.

[33] R. W. White and G. Marchionini. Examining the effectiveness of real-time query expansion. *Inf. Process. Manage.*, 43:685–704, May 2007.

[34] R. Zhang, Y. Chang, Z. Zheng, D. Metzler, and J.-y. Nie. Search result re-ranking by feedback control adjustment for time-sensitive query. In *Proc. NAACL-Short*, pages 165–168, Boulder, Colorado, 2009. Association for Computational Linguistics.