Learning Online Trends for Interactive Query Auto-Completion

Yingfei Wang, Hua Ouyang, *Member, IEEE*, Hongbo Deng, and Yi Chang, *Senior Member, IEEE*

Abstract—Query auto-completion (QAC) is widely used by modern search engines to assist users by predicting their intended queries. Most QAC approaches rely on deterministic batch learning algorithms trained from past query log data. However, query popularities keep changing all the time and QAC operates in a real-time scenario where users interact with the search engine continually. So, ideally, QAC must be timely and adaptive enough to reflect time-sensitive changes in an online fashion. Second, due to the vertical position bias, a query suggestion with a higher rank tends to attract more clicks regardless of user's original intention. Hence, in the long run, it is important to place some lower ranked yet potentially more relevant queries to higher positions to collect more valuable user feedbacks. In order to tackle these issues, we propose to formulate QAC as a ranked Multi-Armed Bandits (MAB) problem which enjoys theoretical soundness. To utilize prior knowledge from query logs, we propose to use Bayesian inference and Thompson Sampling to solve this MAB problem. Extensive experiments on large scale datasets show that our QAC algorithm has the capacity to adaptively learn temporal trends, and outperforms existing QAC algorithms in ranking qualities.

Index Terms—Query auto-completion, multi-armed bandits, exploration v.s. exploitation, Thompson sampling, Bayesian learning model

1 Introduction

Query auto-completion (QAC) is one of the most important features in modern search engines. The main objective of a QAC system is to reduce users' efforts required in submitting a text query. These efforts include typing lengthy texts, correcting spelling mistakes and even finding the most relevant query words. These benefits are becoming increasingly prominent due to the prevalence of mobile devices, since typing on smaller portable devices using virtual keyboards takes more efforts than on PCs. Nowadays QAC is widely adopted by web search engines, news and e-commerce portals, social networks, major web browsers and operating systems. It has also drawn a considerable amount of interests in research, and existing work has addressed this problem from various perspectives, e.g., leveraging contextual information to estimate users' intentions [1], using time-series models to predict popularities based on long-term and short-term query logs [2], [3], [4], personalization based on learning to rank [5], click models of user behavior [6].

During a search session, QAC processes start as soon as a user is ready to type the first character into the search input box. Following each new character he or she entered or deleted in the input box, QAC provides a list of completion suggestions matching the current input (i.e., the *prefix*). The user might choose any item from the suggestion lists as a submission candidate, or totally ignore QAC's suggestions and submit his or her own query by typing the whole words.

Ideally a user's intended query should appear at the top of QAC's suggestion list. Hence the suggested query should be generated by filtering and ranking according to the likelihood of each candidate, given user's search intent and the prefix. Without considering the personalization issue, which is not the focus of this paper, the most popular QAC approach is to estimate this likelihood using the wisdom of crowds, i.e., the popularity of submitted candidates calculated from query logs. This simple idea is also known as the *MostPopularCompletion* (*MPC*) method. It is the basis of many existing work [1], [2], [3], [5], [6], [7], [8], [9].

Most previous MPC approaches were proposed under the batch learning framework, where the i.i.d. assumption is made over the data distribution, and the prediction hypothesis is obtained during the training phase while remaining the same during testing. However, QAC operates in a real-time scenario where users arrive and provide feedback on the displayed suggestion list continually. The most natural and interactive way to solve this problem is learning while doing in an online fashion. One obvious advantage of online learning over batch learning is that online learning has the ability to continually and instantly utilize each new user feedback and make the model better progressively. Most prior QAC work has not considered such an online decision-making setting.

As observed previously [3], [4], [10], [11], MPC is most effective for consistently popular queries such as "dictionary" and "pizza hut", but simple aggregation of the past cannot provide plausible prediction of short-term and unpredictable

- *Y. Wang is with the Foster School of Business, University of Washington, Seattle, WA 98105. E-mail: yingfei@uw.edu.*
- *H. Ouyang is with Apple Inc, Cupertino, CA 95014. E-mail: hua_ouyang@apple.com.*
- *H. Deng is with Google Inc, Mountain View, CA 94043. E-mail: hbdeng@google.com.*
- *Y. Chang is with Jilin University, Changchun, Jilin 130021, China, and Huawei Research, Santa Clara, CA 95050. E-mail: yichang@ieee.org.*

Manuscript received 20 May 2015; revised 12 July 2017; accepted 25 July 2017. Date of publication 11 Aug. 2017; date of current version 4 Oct. 2017. (Corresponding authors: Yi Chang and Yingfei Wang.) Recommended for acceptance by D. Jiang. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TKDE.2017.2738639

w	m
walmart	malaysia airlines
wells fargo	mh370
weather	maps
world cup	mapquest

Fig. 1. Suggestion lists of a major search engine. Left: Prefix *w*, screenshot taken on July 12, 2014. Right: Prefix *m*, screenshot taken on July 18, 2014.

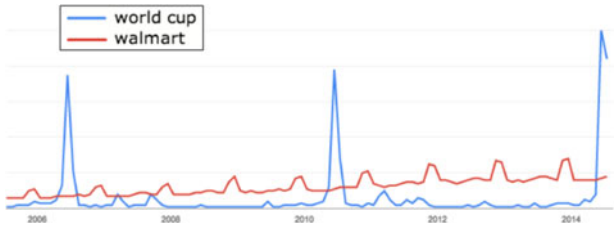


Fig. 2. Daily frequencies for *world cup* and *walmart* from May 2006 to June 2014 according to *Google Trends*.

events or time-sensitive queries. For example, Fig. 1 (Left) shows the suggestion list of a major search engine when a user types *w*. This screenshot was taken during the World Cup 2014 matches. Although *walmart* is generally a more popular query than *world cup*, interests for *world cup* dramatically surpass *walmart* every four years, as shown in Fig. 2. Ranking *world cup* higher during these seasons will apparently improve users' experience. Another example is given in Fig. 1 (Right). This screenshot was taken on July 18, 2014, one day after the tragic crash of the Malaysian Airlines Flight MH17. The suggestion list of the QAC was not able to reflect this breaking news timely.

The second major limitation of existing work is due to the vertical position bias. According to the recent study [6], a query suggestion ranked higher by QAC tends to attract more user clicks regardless of its relevance to the prefix. Therefore there are chances where some queries ranked lower due to the lack of users' feedback are actually more relevant than those ranked higher by QAC. It is important in the long run to place some lower ranked yet potentially highly relevant queries to higher positions, such that more user feedback can be collected. It is possible that this kind of exploration degrades the performance in the short run, since we are taking the risk of abandoning the (seemingly) most relevant queries. This phenomenon is the fundamental exploration-exploitation tradeoff in the setting of online decision making under uncertainty. This setting requires balancing reward maximization given current information (exploitation) with trying new actions to obtain more knowledge (exploration).

Exploration versus exploitation tradeoff is everywhere. In real life, one might always want to try new restaurants although he or she already has some favorite ones in mind; In online advertisements, no matter how relevant the current ads are, it is always important to display different ones that are potentially more relevant. Most off-line methods such as MPC and its variants predict users' intended query purely based on past popularities. This is a "pure exploitation" strategy that suffers from the danger of getting stuck in local maxima. We need to acknowledge that the "optimal" completion under current estimation based on historical data is not truly optimal since it ignores that the estimate-and-optimize cycle will be repeated in the future. The conventional approach does not put any emphasis on active learning and it has no intention on query experimentation in order to learn the actual behavior of putting each possible query in different ranks. Making what we think is currently the best decision may not be the best given the uncertainty in our environment, forcing us to recognize that we have to learn to make better decisions in the future.

To address the time-sensitive challenges and balance the exploration-exploitation tradeoff in QAC, for the first time we propose formulating QAC as an online decision making problem, where the Multi-Armed Bandit (MAB) concept [12], [13] applies naturally. MAB originates from slot machine gambling. The objective of the gambler is to maximize the cumulative rewards over time. We treat the QAC system as such a game, each keystroke as one try and each query as one arm. The rewards come from users' submissions matching the words in the suggestion lists. At each time step, rather than recommending only one query, QAC provides a list of ranked suggestions. This ranked bandit setting has been studied in the literature, from general algorithmic analysis [14], [15], [16] to domain-specific applications such as document retrieval [17], [18] and news recommendation [19]. However, the special problem structures in QAC have not been thoroughly explored by prior work. For example, it is not clear how to properly encode the prior knowledge aggregated from user interactions or editor's labels; existing work has problems in dealing with the huge amount of suggestion candidates associated with a prefix; how to leverage the performance of existing QAC approaches largely remains an open problem.

Although there are very few literatures [14], [20] borrow ideas from bandit community to solve the web document retrieval problem, major differences between the QAC process and the web document retrieval [6], as well as the intrinsic limitations of previous methods prevent them from yielding a good performance on QAC or even applicable to QAC. As studied in [6], most recent click models of web document retrieval are not applicable to QAC without significant modification. Thus the click model based bandit approach [20] is not suitable for QAC. The learning diverse ranking methods [14] uses the information gathered too locally and it does not incorporate any prior knowledge/past popularity about the data. It needs a huge amount of time learning in order to get a fairly well performance even with only a pool of 50 documents in their experiments. Without significant modification, it is not promising for QAC tasks with millions of queries.

In order to tackle these issues, we propose a novel ranked bandits algorithm for QAC, and employ the Bayesian Thompson sampling as an MAB policy. Extensive experiments on large scale datasets show that with modest exploration, our proposed approaches are able to learn the online trending queries dynamically, and significantly outperform existing QAC algorithms in ranking qualities in the long run.

Our main contributions of this work can be summarized as follows:

- We are the first to propose formulating the QAC process as an online decision making problem based on

multi-armed bandits which interact with users in real-time. As a consequence, our MAB model can quickly capture the searching trend while other algorithms based on batch learning suffer from delays in the prediction. In the meantime, the MAB formulation explicitly addresses query experimentation by managing the dilemma that in order to learn the CTR of a query, it needs to be displayed, leading to a potential loss of short-term performance.
- Prior knowledge from past statistics is essential for the relevance of each query. In this paper, we propose a Bayesian framework to fully utilize the prior knowledge derived from previous query logs. We propose to use Thompson Sampling and show that it outperforms other non-Bayesian bandit algorithms.
- We introduce a general framework where one can leverage the performance of any existing ranking model with our proposed bandit models. It learns the users' changing interests and balances exploitation versus exploration while retaining all the favorite features of the existing ranking model.

While in our work, the filtering of candidates is based on exact prefix matching via hash tables for efficient lookup, it is easily combined with any work on predictive auto-completion, e.g., fuzzy matching [21], [22], to expand the list of possible candidates that are eligible for our MAB model. Our work can be also combined with the hybrid framework [1] to improve the ranking of candidates with or without context.

2 Related Work

The novelty of our proposed QAC algorithm is inspired by the progresses in both QAC and the bandit community. In this section we briefly survey some of the most related work.

2.1 Query Auto-Completion

The main objective of QAC is to predict users' intention and assist them formulate a query while typing. The most popular QAC algorithm is to suggest completions according to their past popularity: a frequency score is assigned to each query based on the query log from which the query database was built. This simple QAC algorithm is called MostPopularCompletion, which can be regarded as an approximate maximum likelihood estimator [1].

Several QAC methods [1], [3], [4], [5], [6] were proposed to extend MPC from various aspects. Bar-Yossef and Kraus [1] introduced the context-sensitive QAC method by treating the user's recent queries as context and taking into account the similarity of QAC candidates with this context for ranking. Shokouhi [5] employed learning-based strategy to incorporate several global and personal features into the QAC model. Recently, Li et al. [6] created a two-dimensional click model to combine users' behaviors with the existing learning-based QAC model. However, these models focus on improving relevance ranking, while ignoring the trends and recency of the query log.

To address the time-sensitive problem, Shokouhi and Radinsky [3] proposed an improved MPC, where past frequencies are replaced with frequencies predicted by recursive time series models. Another extension of MPC was proposed by Whiting and Jose [4]. The authors aggregate previous N queries for a prefix (called LNQ), and use linear regression to predict future LNQs based on previous ones. Although these two methods focused on the time-sensitive aspect, their proposed algorithms are not adaptive enough to reflect time-sensitive changes in an online decision-reward fashion.

Our proposed ranked MAB algorithm directly addresses the exploration versus exploitation dilemma. Moreover, as discussed in Section 4.4, domain knowledge explored in the above work can be further leveraged by our proposed method.

2.2 Multi-Armed Bandits

The bandit problem [13] was originally studied under Bayesian assumptions [23]. The major categories include stochastic bandits and adversarial bandits. Many policies were proposed with provable regret bounds. For example, upper confidence bounding policies (UCB) [12], [24], [25], Thompson Sampling [26], [27], [28], and EXP3 [29]. The classic MAB setting was extended to contextual bandits where in each round the learner is provided with a context and the reward depends on both the context and the chosen arm. Different assumptions have been made on how the context and rewards are related [30], [31], [32], [33]. Various papers enriched the basic MAB setting to deal with large number of arms. [34] introduce Lipschitz MAB under the assumption that the arms form a metric space and the rewards satisfy the Lipschitz condition. Other work on Lipschitz MAB includes [35], [36], [37].

Even though the above mentioned methods cover a wide range of mathematical models, the decision of all methods at each time step is a single arm. Without significantly modifications, these bandits formulations can not be directly applied to QAC ranking.

Online learning-to-rank for web document retrieval has been formulated as bandit problems. Uchiya et al. [38] and Kale et al. [39] considered the problem of bandits with multi-plays. Finding better ranking by exploration has been considered in [15], [16], [17], [40], but their settings are different from ours. The results in [40] are obtained based on simulated user clicks rather than real ones. The dueling bandit and ϵ-Greedy based approaches [15], [16], [17] require a special functionality of the retrieval system to interleave two different ranking results. While ϵ-Greedy is a simple exploration strategy in IR, more effective algorithms have been developed in [13]. These work assumes that user can pull more than one arms in each round, which is not the case in QAC. Sloan and Wang propose a UCB-based bandit algorithm for multi-period information retrieval using click models [20]. Although click model has been fully studied for information retrieval, it remains immature for QAC.

Particularly relevant to our work is the ranked bandit formulation introduced in [14]. Independently, a more general sub-modular function maximization setting is studied in [41]. A key feature of ranked bandit is that the click through rate of each document depends on both its relevance and the document shown above. A user considers the results in order and clicks on up to one document. Ranked bandits start with no prior knowledge about the arms and might need to explore for a long time. These policies might work

for problems of 50 documents, but they are not practical for QAC, where millions of queries are presented. In order to deal with the large collection of documents, Slivkins et al. [42] extends the ranked bandits to contextual settings, under the assumption that the rewards satisfy the Lipschitz condition. Several other contextual bandit solutions have been developed for IR [18], [43]. In these works, rich context is shown to be very useful for learning to rank applications. However, the lack of query features in QAC makes it difficult to utilize the contextual information. Hence in this article we will focus on the context-free setting.

While it is clear that multi-armed bandit formulation has been considered in several areas in web search, to the best of our knowledge, this is the first time that exploration versus exploitation has been addressed in QAC ranking and a ranked bandit formulation has been proposed for QAC.

3 Problem Formulation

In this section we first present the standard multi-armed bandit setting. Then we show how the problem of learning an optimal QAC ranking can be formulated as an MAB online learning process.

3.1 Multi-Armed Bandit Setting

Multi-armed bandit is a sequential decision making setting defined over a set of K actions. At each time step t, the learner chooses an action I_t and observes some payoff X_{t,I_t} from the environment, where $X_{t,i}$ denotes the reward obtained on pulling arm i on trial t. In stochastic MAB, there is an unknown probability distribution for each action i and the reward $X_{t,i}$ for each action i is assumed to be drawn i.i.d. In adversarial bandits, no assumption is made about the reward sequence. The chosen action I_t in general depends on previous selections and observations $X_{1,I_1}, \ldots X_{t-1,I_{t-1}}$. The goal is to maximize the commutative payoff obtained in a sequence of T allocations over time, or equivalently minimizing the *regret* of an algorithm in expectation over the draw of the rewards

$$R_T \equiv \max_{i=1,\ldots,K} \mathbb{E}\left[\sum_{t=1}^{T} X_{t,i} - \sum_{t=1}^{T} X_{t,I_t}\right], \tag{1}$$

where $\max_{i=1,\ldots,K}\mathbb{E}\left[\sum_{t=1}^{T} X_{t,i}\right]$ is the best value we can get if we know the underlying probability distribution.

MAB is an instance of sequential decision making problems with partial feedback. It addresses the fundamental trade-off between exploration and exploitation. The learner must balance the exploitation of actions that performed well in the past and the exploration of less played actions that could potentially yield higher payoffs in the future.

3.2 Ranked Bandits for QAC

We use t to index successive users. In Query Auto Completion, a user u_t types in each keystroke. The time point right after the jth keystroke of the user u_t is denoted by (t, j). The first j characters are called a prefix $\mathcal{P}$. At time (t, j), relevant candidates obtained by exact prefix matching construct the possible action set $A_{t,j} \subseteq \mathcal{A}$, where $\mathcal{A}$ is the pool of all the submitted queries in the past and can be grown by adding unseen words. In this work, the list of candidates $A_{t,j}$ consists of all previous queries that start with prefix $\mathcal{P}$. Yet it can work with any advanced string completion techniques, for example, mid-string completions or fuzzy matching. QAC then presents a ranked suggestion list of m queries $Q_{t,j} = \left(q_1(t,j), \ldots, q_m(t,j)\right)$ with each query $q_i(t,j)$ comes from the query candidate set, i.e., $q_i(t,j) \in A_{t,j}$. The user will consider the suggestions in sequence and click up to one query. We assume that there is an unknown click-through rate $\mu_{t,i,k} \in [0,1]$ of any possible query $a_i \in \mathcal{A}$ displayed on the kth rank for the tth user u_t (conditional on the user u_t not clicking on a query displayed higher in the suggestion list). We do not require $\mu_{t,i,k}$ remain static over time. For each of the user's keystroke j, QAC gets a reward if u_t's final submission appears in the suggestion list at time (t, j).

A distinctive feature of our model is that $A_{t,j}$ is time and prefix varying, while most existing MAB algorithms assume a fixed $A_{t,j} = A$. Comparing with existing bandit work for web search [14], [20], our proposed method can directly model the CTR for each query-position pair, and gives us more flexibility to adopt most advanced bandit algorithms, to utilize prior knowledge and to incorporate existing QAC algorithms.

4 Proposed Approaches

In this section, we propose online MAB algorithms for QAC Ranked Bandits. To exploit prior knowledge from query logs, we employ a Bayesian model, and address the problem of prior distribution construction. We introduce a general framework to combine bandit algorithms with existing QAC ranking algorithms to further leverage their performance. Within this framework, one can easily learn users' changing preferences and reduce position bias while maintaining any favorable features of existing ranking models.

4.1 Multi-Armed Bandit Algorithms

We first present two classic MAB algorithms: Upper Confidence Bound policies (UCB) [12], [24], [25] and Thompson Sampling [26], [28]. These two algorithms enjoy provable regret bounds and good empirical performances [44]. They will be used as plugins in our proposed ranked bandit algorithm.

UCB1 [12]. At each time t, action I_t is selected according to the following policy:

$$I_t \in \arg\max_{i=1,\ldots,K} \hat{\mu}_{T_i(t-1),i} + \sqrt{\frac{2\log t}{T_i(t-1)}},$$

where $T_i(t-1)$ is the number of times arm i has been played before time t, $\hat{\mu}_{T_i(t-1),i}$ is the sample mean. The quantity $\hat{\mu}_{T_i(t-1),i}$ is initialized by measuring each alternative once.

Thompson Sampling [26]. This is a Bayesian algorithm which begins with a prior belief $\pi_{1,i}$ on the parameters of the reward distribution of each arm. At each time t the algorithm plays the arm that is the best among the samples from the posterior distribution: $I_t \in \arg\max_i \theta_{t,i}$ with $\theta_{t,i} \sim \pi_{t,i}$. After obtaining the payoff, it updates the posterior distribution $\pi_{t,i}$ by Bayes rule.

4.2 Exploitative Ranked Bandits Algorithm (ERBA) for QAC

Motivated by the Ranked Bandits Algorithm (RBA) developed for web document retrieval [14], we propose a general algorithm named *Exploitative Ranked Bandits Algorithm* for QAC. In contrast to RBA, our ERBA makes an intuitive improvement on how to select a query at a lower rank if the query recommended by the corresponding MAB has already been displayed higher.

ERBA runs an instance of MAB_k for each rank k of the ranked suggestion list. When a user u_t arrives at time t, after each keystroke j, the algorithm obtains a possible action set $A_{t,j}$ by StringMatching (this sub-algorithm is not the focus of this paper). Then it uses MAB_1 to select a query within $A_{t,j}$ and place this candidate in the suggestion list with rank 1. Next, it uses MAB_2 and suggests a query in $A_{t,j}$ to display at rank 2. If the recommended query has already been selected at rank 1, the second query is selected as the next best arm excluding the selected arms according to some criteria (e.g., index or distribution) of the particular MAB algorithm.

After a ranked suggestion list of length m is generated by QAC, the user might click on up to one of the queries, or type in a subsequent character, either because the current suggestion list does not contain any relevant queries, or because of the skipping behavior [6] regardless of the relevance of the queries. If the user types in a new character, ERBA will repeat the above steps to provide another ranked list of m queries. This process terminates when the user click on any query in the suggestion list or submit a query using the search button. Then the values for each MAB instance at each prefix length will be updated according to the following updating rule. For each previous prefix l, if the final submission appears in the kth position of suggestion list and the query is actually recommended by the corresponding MAB_k, the reward for the MAB instance is $r_k(t,l) = 1$; The reward is 0 for all other recommended arms. Pseudo-code of ERBA is shown in Algorithm 1.

The name *Exploitative* in ERBA comes from the conflicting strategy we imposed. ERBA chooses the next best arm upon conflicts, while RBA [14] chooses an arbitrary arm which is a pure exploration approach. The exploration approach could work well in the experimental setting of [14], where a total amount of 50 documents and one fixed query were used. However in QAC, the number of queries matching a short prefix is gigantic, with thousands of queries clicked for very few times and not relevant in general. In practice, a pure exploration approach will almost always degenerate QAC's performance, as we observe in Section 5.

The proposed ERBA is an exploitative counterpart of RBA [14]. In the following theorem we show that ERBA enjoys the same theoretical guarantee of RBA, that is, the bandit algorithm satisfies a sub-linear regret bound R_T, under the next set of assumptions similar to RBA:

(1) The bandit instances for each prefix length are independent, meaning that there is no information sharing across different prefix lengths.
(2) The query sets associated with any prefixes are fixed over time.
(3) The bandit instance specified in ERBA satisfies a sub-linear regret bound $R_T = o(T)$.

Algorithm 1. Exploitative Ranked Bandits Algorithm

input number of positions m
for $t = 1$ *to* T **do**
  j $\leftarrow 1$
  while *not submit* **do**
    $A_{t,j} \leftarrow$ StringMatching(prefix)
    $A_{\text{rest}} \leftarrow A_{t,j}$
    for $k = 1$ *to* m **do**
      $\hat{q}_k(t,j) \leftarrow$ Recommand($\text{MAB}_k(A_{t,j})$)
      if $\hat{q}_k(t,j) \in \{q_1(t,j), \ldots, q_{i-1}(t,j)\}$ **then**
        $q_k(t,j) \leftarrow$ Recommand($\text{MAB}_k(A_{\text{rest}})$)
      else
        $q_k(t,j) \leftarrow \hat{q}_k(t,j)$
      end
      $A_{\text{rest}} \leftarrow A_{\text{rest}} \setminus q_k(t,j)$
    end
    display ranked queries $\{q_1(t,j), \ldots, q_m(t,j)\}$
    $j \leftarrow j + 1$
  end
  for $l = 1$ *to* $j - 1$ **do**
    for $k = 1$ *to* m**do**
      if *submitted* $q_k(t,l)$ & $q_k(t,l) = \hat{q}_k(t,l)$ **then**
        $r_k(t,l) = 1$
      else
        $r_k(t,l) = 0$
      end
      Update(MAB_k, arm = $\hat{q}_k(t,l)$,reward = $r_k(t,l)$)
    end
  end
end

Theorem 1. *Suppose at each time step, a completion suggestion list of length m is provided. After T rounds,*

$$\left(1 - \frac{1}{e}\right)\mathbb{E}\Big[\#\text{ clicks of the optimal static ranking}\Big] \leq \mathbb{E}\Big[\#\text{ clicks of ERBA}\Big] + mR_T,$$

where the expectation is taken over the randomness of ERBA and the randomness of the users' behavior.

By Assumption 1 and the fact that the matching query sets are independent across different prefixes with the same length, we can analyze each prefix independently. From now on, we assume that the upcoming users u_t are querying upon the same prefix. Following the derivation in [14], we first consider deterministic users where each user u_t has a fixed relevant set U_t. The generalization to a standard user $u_t = (\mu_{t,i,k})$ is achieved by analyzing on each sampled deterministic user $\hat{u}_t$ and taking expectations thereafter.

For any set U and an ordered list $Q = (q_1, q_2, \ldots, q_m)$, let $G_k(U,Q) = 1$ if U intersects $\{q_1, \ldots, q_k\}$ and 0 otherwise. Define

$$g_k(U,Q) = G_k(U,Q) - G_{k-1}(U,Q), k = 1, 2, \ldots, m.$$

Let

$$Q^* = \arg\max_Q \sum_{t=1}^{T} G_m(U_t, Q),$$

$$\text{OPT} = \sum_{t=1}^{T} G_m(U_t, Q^*).$$

Lemma 1. *For all $k = 1, 2, \ldots, m$, let Q_t be the suggestion list at time t,*

$$\mathbb{E}\left[\sum_{t=1}^{T} g_k(U_t, Q_t)\right] \geq \frac{1}{m}\mathrm{OPT} - \frac{1}{m}\mathbb{E}\left[\sum_{t=1}^{T} G_{k-1}(U_t, Q_t)\right] - R_T.$$

Proof. Compared to RBA, when dealing with conflict, rather than selecting an unchosen query uniformly at random, ERBA selects next best arm excluding the selected arms according to some criteria (e.g., index or distribution) of the particular MAB algorithm.

Yet this modification does not affect the sequence of queries $(\hat{q}_1(t,) \ldots, \hat{q}_m(t))$ recommended by the algorithms MAB_1,..., MAB_m at time t. At the same time, this modification does not affect the reward provided for each MAB_k since MAB_k only gets a reward of 1 if user submits $q_k(t)$ and $q_k(t) = \hat{q}_k(t)$. Thus, the proof follows the original derivation in [14].

The first step is to show $g_k(U_t, Q_t) \geq f_{kt}(\hat{q}_k(t))$, where

$$f_{kt}(q) = \begin{cases} 1 & \text{if } G_{k-1}(U_t, Q_t) = 0 \text{ and } q \in U_t, \\ 0 & \text{otherwise.} \end{cases}$$

$f_{kt}(\hat{q}_k(t)) = 1$ implies $G_{k-1}(U_t, Q_t) = 0$ and $\hat{q}_k(t) \in U_t$. By construction of ERBA, we have $q_k(t) = \hat{q}_k(t)$, leading to $g_k(U_t, Q_t) = 1$.

By Assumption 3 (regret bound for MAB), we have

$$\sum_{t=1}^{T} \mathbb{E}[f_{kt}(\hat{q}_k(t))] \geq \frac{1}{m}\mathbb{E}\left[\sum_{q \in Q^*}\sum_{t=1}^{T} f_{kt}(q)\right] - R_T.$$

The last step is to prove that

$$\sum_{q \in Q^*} f_{kt}(q) \geq G_m(U_t, Q^*) - G_{k-1}(U_t, Q_t).$$

It is trivially true when the right side takes values in {-1,0}. For the case where $G_m(U_t, Q^*) = 1$ and $G_{k-1}(U_t, Q_t) = 0$, there exists one $q \in Q^* \cap U_t$ such that $f_{kt}(q) = 1$. □

Lemma 1 can be viewed as m constraints for the classic linear program encountered when bounding a greedy policy. Theorem 1 is thus readily obtained by either Lemma 4.1 of [45] or the argument in [14].

4.3 Using Thompson Sampling in ERBA

4.3.1 Motivation: Prior Knowledge

Existing work on ranked bandits [14], [20], [39] are all based on UCB and EXP3 policies. These algorithms start with no prior knowledge about the arms and need to explore for a long time. Lacking of prior information makes these algorithms perform poorly for QAC, since they need to put every arm in every rank for a reasonable amount of time, such that users' feedback can be gathered. These policies might work for toy problems of 50 documents, but they are not practical for QAC, where millions of queries are presented. The size of the candidate pool for a prefix is huge, where a large amount of queries were submitted with typos. In general these queries are less relevant to users. Including these queries for exploration will definitely lead to bad search experiences. In practice, no web search engine can ignore the prior information aggregated from past query logs. Based on these observations, we propose to adopt a Bayesian framework for our QAC tasks. We choose Thompson Sampling as our MAB algorithm, since this simple and classic Bayesian method was shown to have very good empirical performance [44].

4.3.2 Bayesian Learning Model

The Beta-Bernoulli distribution is good at modeling the probability of user u_t clicking a query a_i displayed on the kth rank $\mu_{t,i,k}$. Under the Bayesian setting, each observation of whether a user clicks a suggested query or not can be modeled as a Bernoulli random variable. It is 1 (success) with probability $\mu_{t,i,k}$ and 0 (failure) with probability $1 - \mu_{t,i,k}$.

It is standard to model the mean $\mu_{t,i,k}$ using a Beta distribution with parameters $\alpha_{i,k}$ and $\beta_{i,k}$ which are always integer and might be changing over time [44], [46], [47], [48]. Recall that the beta density is given by

$$f(x|\alpha, \beta) = \begin{cases} \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1}(1-x)^{\beta-1} & \text{if } 0 < x < 1 \\ 0 & \text{otherwise.} \end{cases}$$

Starting from a prior estimate $\alpha_{0,i,k}$ and $\beta_{0,i,k}$, after each Bernoulli trial, due to the conjugacy property, the parameters are updated according to following equations:

$$\alpha_{t+1,i,k} = \alpha_{t,i,k} + r_k \tag{2}$$

$$\beta_{t+1,i,k} = \beta_{t,i,k} + (1 - r_k). \tag{3}$$

Parameter α_t is roughly the number of clicks in t observations, and β_t is roughly the number of non-clicks upon a suggestion. Consequently, $\frac{\alpha_t}{\alpha_t+\beta_t}$ estimates the click-through-rate (CTR) for each query at a rank position, and $\alpha_{t,i,k} + \beta_{t,i,k}$ is roughly the number of times query i is shown at rank k up to time t. The prior is constructed as follows: given a query log of length t, $(\alpha_{t,i,k}, \beta_{t,i,k})$ are set to the number of clicks and non-clicks for each query i at each rank k. They will be set to $(1, 1)$ if the corresponding query never appeared in that rank, which mounts to the uniform distribution on $[0, 1]$. Since only two numbers are needed to represent the posterior, it is memory efficient in practice. Section 5.2 will include more details.

4.3.3 Thompson Sampling Based ERBA

Inserting Thompson Sampling into the general ERBA framework proposed in Algorithm 1, we obtain a concrete algorithm *TS-ERBA*, where prior is an additional input. For each prefix, each rank k and each query $a_i \in A_{t,j}$, we sample $\theta_{t,i,k}$ from the Beta($\alpha_{t,i,k}, \beta_{t,i,k}$) distribution. Only one sample is obtained, which takes O(1) time and is efficient in practice. The recommended query will be $\hat{q}_i(t) \in \arg\max_i \theta_{t,i,k}$. After user's submission, the algorithm will update its parameters according to Eqs. (2) and (3).

In the follows we explain the intuition why TS-ERBA can adaptively adjust QAC's suggestion list according to temporal changes and balance the exploration/exploitation tradeoff automatically.

Thompson Sampling relies on random sampling to balance exploration and exploitation. For breaking news that has never appeared in the history, the prior is $(1, 1)$ whose sampling distribution is the uniform distribution (equivalently

Beta(1,1)). This exploration behavior gives new queries a good chance to be selected. For queries that have been clicked previously, $\frac{\alpha}{\alpha+\beta}$ approximates the probability of being clicked. If this ratio becomes larger, the peak of the pdf will move towards the right, yielding a higher probability of a larger sampled value. Thus recent clicks of a suggested query will increase the probability of being displayed later, while recent non-clicks (may be due to users' decreasing interests) will decrease this probability.

In the same time, α and β can be viewed as a measure of our confidence in estimation. With the same ratio of α/β, larger values of α and β mean that we are very confident in our current estimation and thus the distribution is more centered around its expected value. Smaller values of α and β mean that we do not have much knowledge of this query. A wider bandwidth offers a higher chance of sampling a larger value and thus gives less displayed queries an opportunity to gather feedback from users. Moreover, the estimation of a query with large α and β is not likely to be changed a lot, while that with small α and β tends to be changed after only a few observations. This implies that consistent popular queries from past query logs will still remain popular, while those queries with potentially higher relevance will have a chance to prove themselves.

Thompson Sampling might recommend the same query for several rank positions, since the queries could be really competitive. However, our TS-ERBA can make sure that the the recommended query is displayed at the highest rank. According to the updating rule of TS-ERBA, if a user clicked this query, the highest rank will get reward 1, otherwise 0, while the lower ranks will always get 0. This will decrease the probability of recommending this arm at lower ranks, or equivalently increasing the probability of other queries being recommended at lower ranks.

A desirable property of any ranking strategy is to gradually boost the rank of a recent popular query according to the MRR metric. TS-ERBA achieves this property by punishing a non-clicked higher ranked query. In order to boosting this effect, we assume that due to position bias, if a query is clicked at rank k, it will also be clicked if it is displayed at higher ranks. Thus the positive feedback gathered at some position can be propagated to other positions. Specifically, if user submits a query on the kth rank, aside from the update of the kth rank's statistics, we also give a positive reward to any higher ranks $j < k$ by updating their $\alpha_{t+1,i,j}$ to $\alpha_{t+1,i,j} = \alpha_{t,i,j} + 1$. We do not propagate the reward if it is zero. The resulting algorithm is named *Boosted Thompson Sampling based ERBA (Boosted-TS-ERBA)*.

4.4 Leveraging Existing QAC Methods

Previous studies addressed the QAC problem from various perspectives, e.g., ranking based on static scores [1], [21], developing click models [6], combining personalized features [5] and using time-series models [3], [4]. Although our approaches are developed under a different setting (online decision making), previous progresses are definitely not ruled out. A better strategy is to incorporate existing ranking algorithms into our framework to further increase the performance. We propose a very simple method for this purpose: at each time t, an existing QAC method provides their top-n queries matching the current prefix of length j. This set is then used as the action set $A_{t,j}$ in our proposed ERBA and TS-ERBA. Here n can be treated as a tunable parameter that controls the exploration rate performed by ERBA since larger value leads to more seemingly less relevant queries to be considered.

Our method can be then regarded as a re-ranking mechanism. If an existing QAC algorithm A can achieve a better prediction than another algorithm B, the top-n queries returned by algorithm A will be more relevant to the user's intent. If we combine our methods with algorithm A, our methods are then based on a better base action set to perform real-time interaction with users.

5 Experimental Results

In this section, we conduct a series of experiments to validate the claims of our proposed algorithms using Yahoo Search's query log data.

5.1 Experiment Settings

Real-Time QAC Simulation. As QAC operates in a real-time scenario, and our model works in a online learning fashion, we simulate the real-time user search behavior based on the query log. The QAC processes start when a user types a first character into the search box and following each new character entered in the search engine, QAC provides suggestions matching the current prefix. Accordingly, the query log datasets are ordered and tested chronologically.

We use Yahoo Search's June and July 2014 QAC log data for evaluation. Two types of experiments are carried out. The first one assumes that a user would have selected their submission if it had been displayed to them. The second one does not make such assumption and considers the skipping user behavior. Specifically, for type-one experiments, given a submitted query e.g., 'abc', we run each algorithm for all prefixes 'a', 'ab', and 'abc'. Experiments for each prefix length were run independently as suggested in [3]. Type-two experiments assume that a user will submit the query after a fixed prefix length regardless of whether the query has been presented before or not.

Metrics. Mean Reciprocal Rank (MRR) is used as the main measurement metric in the QAC literature [1], [5]. Reciprocal rank of a query response is the inverse of the displayed rank. If there is no matching result for the query, the inverse is set to zero. Mean reciprocal rank is the average of the reciprocal ranks of all the queries encountered during some time horizon. We also report Click-Through-Rate and Clicked-MRR where only the final submitted queries are taken into account in calculation.

Baselines. The most common approach is to use the aggregated frequency of a query over past search logs as an approximation for its future. It then uses the aggregated values for ranking QAC suggestions. It is referred to as *MostPopularCompletion* and has been reported to be very competitive and widely used in the industry [1], [5]. It ranks the queries according to

$$\arg\max_{q \in \mathcal{A}(\mathcal{P})} w(q),\ w(q) = \frac{f(q)}{\sum_{q'} f(q')},$$

where $f(q)$ denotes the number of times the query q occurs in the precious search log. However, the query popularity

may change over time and thus we also consider refreshing the query statistic to account for recent changes. But it is not clear how often the refreshness can best approximate the future popularity. In our comparisons, we investigate the affect of different refreshing scheme to the MPC algorithm. In the experimental results, we use subscripts to indicate the refreshing frequency (in hours) of MPC. For example, MPC_{24} stands for the most popular clicks in the past 24 hours. MPC without subscript is used to indicate the popularity in May 2014 without refreshing.

We also compare with time-sensitive auto-completion which replaces the aggregated query frequencies with forecasted values competed by time-series modeling of query history [2], [3]. It can be viewed as the context-sensitive QAC ranking [1] by treating time as context. Since the time span of the data used in our experiments do not include seasonality, we use double exponential smoothing in our comparison to cover for the trends. The initial value of $\bar{y}_1$ is set to y_1 and $F_1 = y_2 - y_1$.

We also consider UCB1-based RBA [14] as a baseline. UCB-type policies are preferable for problems with large time horizon relative to the action space. They will actually behave like a pure exploration strategy when the time horizon is not large enough compared to the action space. However, in QAC tasks, the number of queries associated with each prefix is huge, and the number of times that each query could be displayed in the suggestion list is small. In order to improve the performance of UCB1-RBA, we impose a similar strategy as we proposed in Section 4.4: the top n suggestions of MPC is provided to UCB1-RBA as its action set. Here n is a tuning parameter.

5.2 Prior Construction

To obtain reliable prior information for beta distribution parameters α and β, we collected a high-resolution QAC query log from May 2014. This log records every keystroke along with the associated suggestion lists (top-10) returned by Yahoo Search. Users may type multiple keystrokes before they submit the final query. We denote the suggestion list associated with each keystroke as a single *view* of a QAC *session*. A QAC *session* may consists of multiple views starting from the first keystroke to the final view of a submitted query. Generally, there is a single clicked query corresponding to a QAC session. For every view, if the suggestion i at rank k equals to the final clicked query, we say that suggestion i is clicked at rank k; otherwise, it is not-clicked at rank k. With this QAC log, we count the number of clicks $\alpha_{0,i,k}$ and the number of observations $\alpha_{0,i,k} + \beta_{0,i,k}$ for every suggestion i at rank k, and our prior information is obtained.

Counting from past data is definitely not the only way to construct priors. Other approaches could be incorporating knowledge from domain experts or using annotations from human editors. These information might be qualitative in nature and are not easily utilized by frequentist approaches (such as UCBs). Prior construction and the impact of any inconsistencies in the prior distribution are common problems inherent in any Bayesian approaches [49]. However, a study on this topic is beyond the scope of this paper.

Similar to the idea of refreshing frequency in MPC or the choice of unit intervals in TimeSeries, our model can also be combined with frequent updates of query statistics to construct the prior distributions. In fact, without refreshing the prior distributions, after each view, either α or β will be increase by 1. As time goes by, high values of α and β indicate a high confident in our estimate, hence this estimation is not likely to be changed significantly to account for popularity changes. As a consequence, we propose to reconstruct the prior distribution after each time unit (e.g., day, or month) by aggregating actual values of the data within the last time unit. A longer time unit provides less biased estimation while it is harder to timely follow the temporary trend.

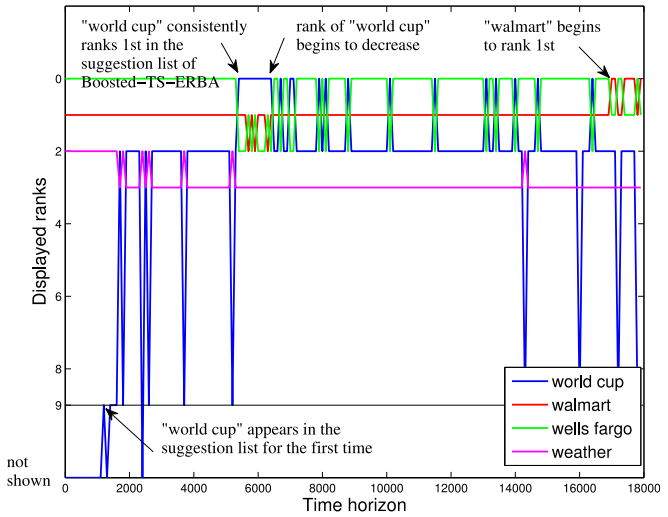


Fig. 3. Changing ranks of the four queries *walmart*, *weather*, *wells fargo*, and *world cup* experimented with a prefix length of 1.

5.3 Learning Temporal Trends

In this section, we demonstrate how our proposed algorithm is able to learning the online trends adaptively. In real-world query logs, users' interaction behaviors are complex, and it is hard to see how the algorithms react to users' feedback instantaneously. Hence we first construct a toy example in order to illustrate the interaction between our proposed algorithm and users' time-varying interests.

5.3.1 Toy Example

We use the *world cup* example mentioned in Section 1 to illustrate the sharp increase of users' interests. We mimic users' behavior by assuming that there are 6,000 successive search sessions for *world cup*, followed by 12,000 successive search sessions for *walmart*. MPC's top 4 suggestions for prefix *w* is {*walmart*, *white pages*, *weather*, *wells fargo*}, with *world cup* ranked 22.

Fig. 3 illustrates the changing ranks of the four queries *walmart*, *weather*, *wells fargo* and *world cup*. These ranks are generated by our proposed Boosted-TS-ERBA. We can see that *world cup* appears for the first time after only 785 steps. Its rank is boosted to top 1 after 5,291 steps and stay on the top of the list after 5,676 steps. When the temporal interests of *world cup* diminish, its rank also falls gradually over time. This validates our intuition explained in Section 4.3.3.

5.3.2 Learning Temporal Tends for w

We perform a series of experiments to test our approaches with the log data collected on 20140613. This date is within

TABLE 1
Relative Performances of Different Methods Compared to MPC

	CTR+	MRR+
UCB1-RBA	-26.8%	-62.05%
UCB1-RBA ($n = 15$)	-21.07%	-43.96%
Boosted-TS ($n = 30$)	$+69.22\%$	$+0.87\%$
MPC_4	$+16.63\%$	$+1.61\%$
MPC_8	$+15.35\%$	$+1.41\%$
TimeSeries_4	$+15.48\%$	$+1.47\%$
Boosted-TS_4	$+41.44\%$	$+1.52\%$

Used 655,132 w-initiated records from sampled query log data of 20140613. Experimented with prefix 1.

TABLE 2
Relative Performances of Different Methods Compared to MPC

	CTR+	MRR+
UCB1-RBA	$-51.99\ \%$	$-73.46\ \%$
UCB1-RBA ($n = 15$)	$-26.44\ \%$	$-59.15\ \%$
Boosted-TS ($n = 30$)	$+20.18\%$	$+8.65\%$
Boosted-TS ($n = 20$)	$+19.38\%$	$+10.67\%$
Boosted-TS ($n = 15$)	$+14.79\%$	$+5.24\%$
MPC_4	$+20.9\%$	$+7.24\%$
MPC_8	$+19.7\%$	$+6.48\%$
TimeSeries_4	$+32.5\%$	$+10.97\%$
Boosted-TS_4	$+37.6\%$	$+11.86\%$

Used 655,132 w-initiated records from query log data of 20140613. Experimented with prefix 2.

the events of World Cup 2014. We report QAC performances for 655,132 QAC sessions that begin with the letter *w*. All algorithms generate a suggestion list of length 10.

Since we adopt an exact string match when providing possible action set for each prefix, the updating of our Bayesian model and the suggestion list are independent across different initial letters, and thus provide the ability of paralleling chronologically testing records with different initial characters on large datasets.

We use subscripts to indicate the refreshing frequency (in hours) of MPC, TimeSeries and Boosted-TS. For a fair comparison, we use the same time unit (4 hour) on this dataset for different algorithms. TimeSeries uses smoothing scheme to account for historical data, so the choice of the time unit does not dramatically damage its performance other than possible delays in the prediction. As for our Boosted-TS, the balance between exploitation and exploration achieved by Thompson sampling can largely offset the potential bias introduced by different choices of time units. Yet it is not clear how often the logs need to be updated to best predict the future popularity in MPC. So in this section, we experiment with an additional instance of MPC with an 8-hour refresh rate.

Table 1 reports the CTR and MRR of different algorithms on *w*-initiated records experimented with a prefix length 1. The reported percentages are relative to the MPC (without refreshing frequencies) baseline. With a prefix *w*, *world cup* does not appear in the top-10 suggestions of MPC.

Since the number of queries associated with each prefix is huge, the number of times that each query could be provided in the suggestion list is small. Thus all bandit algorithms need a quite long time to learn the landscape before they reach a good performance, resulting in a short term performance degeneration. Only a few queries have been clicked for significant amount of times, and most others are not relevant. In order to obtain a good performance in both short and long term run, we used the methods proposed in Section 4.4 and feed Boosted-TS-ERBA's action set with a list of n queries generated by MPC. We experiment with $n = 15, 20, 25, 30, 35, 50$ for UCB1-RBA and Boosted-TS. Table 1 reports the best values we achieved.

Without utilizing past information, UCB1-RBA experiences a huge performance decrease in this one-day period. This is due to excessive exploration required by RBA's conflicting strategy. Its performance could be better after a long waiting of exploration. However, a bad CTR/MRR for several weeks or even months is not acceptable in practice. After combining with MPC's top-15 suggestions, UCB1-RBA's performance gets slightly better. With a properly constructed prior, Boosted-TS significantly outperforms both MPC and UCB1-RBA in terms of CTR, showing the power of a proper exploration-exploitation tradeoff. The increment of MRR, compared to that of CTR, is relatively smaller. One possible explanation is that when achieving the overall goal of optimal ranking, ranked bandit algorithms are exploring potential queries at different ranks to allow training data about these queries to be collected.

MPC_4 and MPC_8 achieve better perform than MPC due to their recency of the query log to reflect the trend. TimeSeries_4 has a similar performance on this dataset. Even though Boosted-TS_4 with prior reconstruction yields a higher CTR than that of MPC_4, MPC_8 and TimeSeries_4, the price to pay for exploration is a smaller increase in the MRR.

Compared to methods with frequent refresh that are based on more recent estimations of the queries, more data provides less biased estimation. For example, even though the trending words are not ranked as the highest in Boosted-TS without refreshing, the suggestion list contains other popular words such as walmart. While for the methods with frequent refresh, the suggestion lists are more biased towards the trending words. This explains the fact that Boosted-TS achieves the highest CTR, but a relative low MRR.

The same sets of experiments are conducted for a prefix length of 2 and reported relative performances in Table 2. $n = 15$ yields the best performance for UCB1-RBA. Performance of different choices of n for Boosted-TS-ERBA is provided to demonstrate the effects of n. We can observe that that no matter which n we chose, the performance of Boosted-TS-ERBA is consistently better than previous methods without refreshing frequencies. Larger n implies more exploration and less bias since more queries are considered. If actual relevant queries lie in between the MPC ranking of a smaller n and a larger n, a larger n provides a chance to discover the truth. Of course price will be paid for exploring more queries. Same as the exploration/exploitation tradeoff, a good balance is hard to find in practice. A good strategy is to use larger n at the beginning and gradually decreasing the exploration rate as time goes by.

For prefix *wo*, *world cup* statically ranks the 6th in the suggestion list of MPC. In comparison, *world cup* appears at rank 1 for 34,093 times for Boosted-TS-EBRA ($n = 30$),

TABLE 3
Relative Performances of Different Methods Compared to MPC

	CTR+	MRR+
UCB1-RBA	−80.3%	−89.4%
UCB1-RBA ($n = 20$)	−39.5%	−67.2 %
Boosted-TS ($n = 30$)	+11.53%	+3.76%
Boosted-TS_4	+24.29%	+4.96%

Used 655,132 w-initiated records from query log data of 20140613. Experimented with up to 2-character prefixes.

TABLE 4
Number of Times *mh17* Appears at Each Rank on the *mh*-Data from 20140716 to 20140720 Experimented with a Prefix Length of 2

Rank	TS-ERBA	Boosted-TS-ERBA	Boosted-TS_{24}
0	554	6,062	10,104
1	2,110	2,674	2,820
2	5,158	1,558	328
3	486	957	143
4	162	576	79
5	75	415	45
6	1,322	309	38
7	223	203	55
8	200	173	29
9	272	168	42

48,602 times for Boosted-TS-EBRA ($n = 20$) and 48,168 times for Boosted-TS-EBRA ($n = 15$). The boost in ranks clearly shows that the temporal trends of World Cup are captured by our algorithms.

We also compare with MPC_4, MPC_8, $TimeSeries_4$ and Boosted-TS_4. It can be seen that MPC_8 yields a worse performance than MPC_4. $TimesSeries_4$ and Boosted-TS_4 achieve the best performance.

We did more experiments with prefix lengths greater than 2 and have not observed much improvement. The reason is that QAC is considerably more effective with a longer prefix due to the huge reduction of the space of possible matching queries for each extra character [1], [4]. Since MPC already has a good chance of completion suggestion match and the intent of the user is more predictable, exploration does not bring much benefits to it.

We then run the type-two experiments, assuming that the user will submit the query after a fixed prefix length of 2 regardless of whether the query has been presented before or not. To be specific, for each record in the query log, we will fist provide a top-10 suggestion list for the first character and assume that the user will not stop at this point due to skipping behavior and types in a second character. After the suggestion list is provided for the prefix of length 2, the user will submit the intended query which provides feedback for both our suggestion for the prefix of length 1 and of length 2. The skipping user behavior does not affect the performance of QAC algorithms, e.g., MPC and TimeSeries, that predicts though total frequencies (rather than individual statistics at each rank) of each candidate. We omit the results here since they are the same as in the previous type-one experiments. While in our models, since the same candidate can occur in different positions in the suggestion list of different prefix lengths, the statistics at different ranks will be updated. Relative performance improvements are reported in Table 3. Again our Boosted-TS, either using the 4-hour refresh or not, outperforms MPC and UCB1-RBA.

TABLE 5
Performances of Different Methods on the *mh*-Data, Collected from 20140716 to 20140720

	CTR	MRR
MPC	3.59%	0.0141
TS-ERBA ($n = 20$)	12.87%	0.0418
Boosted-TS ($n = 20$)	15.82%	0.0986
MPC_{24}	20.63%	0.1301
$TimeSeries_{24}$	20.89 %	0.1308
Boosted-TS_{24}	21.43%	0.1282

Experimented with a prefix length 2.

5.4 Learning Breaking News

In this section, we examine the reaction speed of our proposed algorithms to an unpredictable query. We use the breaking new of the tragic crash of the Malaysian Airlines Flight MH17 as an example. This accident took place on July 17th, 2014. The refreshing time unit for MPC, TimeSeries and Boosted-TS is set to be daily. We experiment with *mh*-initiated QAC sessions from 20140716 to 20140720 and report the performance of each method for a prefix length 2. Table 4 reports the number of times *mh17* appears at each rank position.

Since MPC predicts based on past popularity, it never explores and displays a newly occurred event. In comparison, after the first occurrences of *mh17*, it is selected by TS-ERBA with high probability to display at any rank, due to high uncertainty on this query. If such exploration receives positive feedback (which is mostly the case for breaking news), our algorithm will gradually boost the query to a higher position. Boosted-TS-ERBA not only updates the belief on the displayed rank but also propagates the feedback to high ranks, hence the boosting effect is more significant, as we can observe in Table 4. Since the most recent query log is moving towards the *mh17* related contents, with a daily refresh, Boosted-TS_{24} is able to place more emphasis on the query of interest by displaying it more often in the ranked list, especially on the first two positions. Aside from this exploitative behavior, we can observe a decent number of exploration on lower ranks. In contrast, with a daily refresh, MPC_{24} displayed *mh17* for 11,220 times, only on the first position with no demonstration on exploration. $TimeSeries_{24}$ exhibits a similar behavior.

The value of exploration is shown in Table 5 when we report CTR, MRR and Clicked-MRR achieved by different algorithms. Thanks to exploration, all the recorded TS-ERBA methods manage to find current popular queries and all the measures are significantly better than MPC. Since boosted version makes more use of each user's feedback, its performance is even better.

Even without refreshing, Boosted-TS is comparable with MPC_{24}, $TimeSeries_{24}$ and Boosted-TS_{24}. MPC_{24} and $TimeSeries_{24}$ have similar performance, which will be further demonstrated in Fig. 4. Boosted-TS_{24} yields the highest CTR at the end of time period. In fact, the peek of CTR within

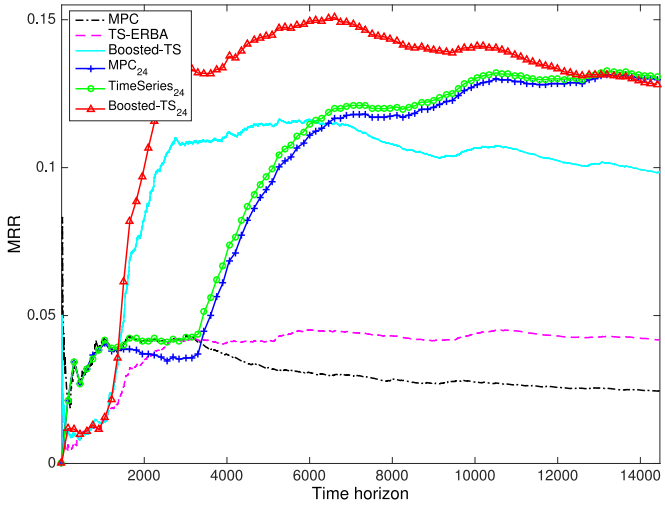


Fig. 4. MRR of different methods on the *mh*-data from 20140716 to 20140720 experimented with a prefix length 2.

these 5 days is achieved by Boosted-TS$_{24}$ at 23.85 percent with a MRR maximum at 0.1512.

Fig. 4 illustrates the changing MRR of each algorithm, over a total of 14,412 sessions plotted chronologically. We see that in early stage, especially in the first 1,000 iterations, all ERBA algorithms suffer from a severe deterioration of their short-term performance. This is a common phenomenon for any multi-armed bandit algorithm, when comparing with pure exploitation strategies. Yet after a modest exploration and learning period, it gradually finds the satisfactory rankings under continuously refined CTR estimates as information accrues. While pure exploitation strategy has the danger of getting stuck in local maximum, leading to linear regret, bandit algorithms balance exploration and exploitation carefully, and can achieve an ideal sub-linear regret. The boost in performance shortly after the trough illustrates the value of more user feedback collected, the balance between exploitation versus exploration, and the capture of the recent query popularity. The gap between the Boosted-TS and the non-boosted TS-ERBA demonstrates the effectiveness of the information propagation.

The other message we can get from this chronologically plot is that the prediction of TimeSeries and MPC with refreshing suffers from lags. To be more specific, the trend predicted by TimeSeries and MPC are similarly delayed by one time unit after the breaking news. This is easily explainable since MPC and TimeSeries are essentially batch learning algorithms which cannot perform model update within each time interval. While in contrast, our MAB model works in real time and it can quickly capture the searching trend of the breaking news. This property is best illustrated in Fig. 4 with the sharp increase of TE-ERBA, Boosted-TS and Boosted-TS$_{24}$ shortly after the first occurrences of mh17 related queries.

5.5 Larger Scale Experiments

In this section we use a larger dataset sampled from query logs for date 20140613. This dataset contains 9,519,634 QAC sessions, significantly larger than previous experiments. All experiments were carried out with a prefix length 2.

In the following results, both TS-ERBA and Boosted-TS are combined with a top-20 ($n = 20$) suggestion list provided by MPC as the possible action set. Since queries with different initial letters have no effect on each other, the testing can be performed in parallel for efficient computation. The statistics (CTR and MRR) can be obtained for each letter initiated records. For MPC, totally 52 character-initiated data subsets have at least one matching result, while the number is 65 for UCB1-RBA and Boosted-TS, and 66 for TS-ERBA.

Boosted-TS achieves a CTR of 18.5 percent and a MRR of 6.67 percent for $>$-initiated sessions which appears 27 times in one-day period. In comparison, no matching is found in the MPC suggestion list; Boosted-TS achieves a CTR of 0.15 percent and a MRR of 0.08 percent for ″-initiated sessions which appear 26,730 times in one-day period with no matching results found in MPC. The reason is that since the number of searching for such queries are far smaller than others (e.g., compared with 796,777 *c*-initiated sessions), MPC is more biased due to the shortage of past data. Exploration then plays an important role when the current information is of high uncertainty and tries to discover the true users' intents.

Table 6 summarizes the proportions of initial characters on which other methods achieve higher values of CTR and MRR compared to MPC, respectively. Since UCB1-RBA does not utilize past information and uses an explorative conflicting strategy, the performance is degenerated at least in a one-day period. The beating portion comes from the exploration for less displayed queries. TS-ERBA and Boosted-TS enjoy better performance aided by prior information and a better exploration/exploitation balance.

Table 7 reports the CTR and MRR on *all* the 9,519,634 QAC sessions. Table 8 extracts the results for all initial characters (excepting *w*) with more than 500,000 QAC sessions. Due to a higher certainty resulting from more past data, MPC is less biased and thus harder to beat at least in short term without notable temporal trends. In general both TS-ERBA and Boosted-TS manage to achieve a higher CTR (which is actually the objective for bandit algorithms) by balancing exploitation and exploration and yet pay the price of discarding seemingly promising queries at top ranks. This explains why MRR of ERBA is smaller than MPC while CTR is higher as seen in *m*-initiated and *t*-initiated sessions. Meanwhile the

TABLE 6
The Proportions of Initial Characters on which Other Methods Beat MPC

	CTR	MRR
UCB1-RBA	36.92%	29.2%
TS-ERBA	89.23%	57.69%
Boosted-TS	87.69%	59.23%

TABLE 7
Performances of QAC Rankings of Different Methods on all the 9,519,634 QAC Sessions

	CTR	MRR
MPC	6.68%	2.57%
UCB1-RBA	1.11%	0.25%
TS-ERBA	6.71%	2.84%
Boosted-TS	6.84%	2.85%

TABLE 8
Performances of Different Methods on the Selected QAC Sessions from the Query Log Data of 20140613

	CTR	MRR	CTR	MRR
	a-initiated		*b*-initiated	
MPC	7.00%	2.26%	3.11%	1.33%
UCB1-RBA	0.93%	0.22%	0.88%	0.2%
TS-ERBA	7.12%	2.69%	3.99%	1.23%
Boosted-TS	7.14%	2.7%	4.1%	1.24%
	c-initiated		*h*-initiated	
MPC	2.09%	0.84%	3.74%	1.07%
UCB1-RBA	0.53%	0.11%	0.68%	0.15%
TS-ERBA	4.75%	1.21%	3.54%	1.24%
Boosted-TS	4.77%	1.22%	3.85%	1.27%
	m-initiated		*s*-initiated	
MPC	3.52%	1.57%	2.08%	0.75%
UCB1-RBA	1.08%	0.24%	0.81%	0.18%
TS-ERBA	3.33%	1.01%	2.35%	0.79%
Boosted-TS	3.41%	1.03%	2.46%	0.81%
	t-initiated		*p*-initiated	
MPC	4.58%	2.14%	4.44%	2.13%
UCB1-RBA	1.58%	0.33%	1.14%	0.23%
TS-ERBA	4.71%	1.84%	5.33%	2.17%
Boosted-TS	4.82%	1.87%	5.44%	2.18%

amount of increase of MRR is usually smaller than that of CTR. By propagating the feedback to higher ranks and boosting the ranking effect, Boosted-TS always achieves a higher MRR than TS-ERBA in the experiments.

6 Conclusion

We propose a novel online decision making formulation for Query Auto-Completion. While most existing work focuses on batch learning algorithms, we show that our online setting is able to learn the trending queries dynamically from users' feedback. We develop a general Exploitative Ranked Bandits Algorithm for QAC. To the best of our knowledge, this is the first time that the exploration and exploitation tradeoff is explicitly addressed in QAC to balance immediate earnings on the one hand and learning the actual CTR of potential rankings on the other. To exploit important prior knowledge derived from query history, we propose using Thompson sampling as a Bayesian multi-armed bandit instance for ERBA, which randomly selects a query according to its probability of being optimal in the exploration/exploitation setting. We also leverage existing QAC algorithms by combining them with the proposed ERBA framework. We demonstrate that the proposed ERBA significantly outperforms existing QAC algorithms. The Bayesian Thompson sampling algorithm effectively utilizes prior knowledge and significantly outperforms non-Bayesian UCB policies.

High values of α and β indicate a high confident in our prior estimate, hence this estimation is not likely to be changed significantly. To this end, we propose prior reconstruction/refresh after a unit time interval and compare with TimeSeries and MPC with frequent updates. We come to the conclusion that our MAB model works in real time and it can quickly capture the searching trend while MPC and TimeSeries suffer from delays in the prediction.

As on-going work, we are using online bucket test to validate the effectiveness of our algorithms. As future work, we plan to investigate better methods and information sources to generate prior knowledge for Thompson sampling. We also plan to improve Thompson sampling's rewarding procedure for QAC: a rank-dependent reward is desired.

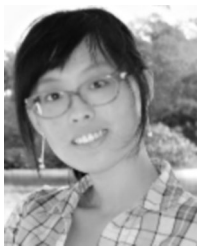
Acknowledgments

This work was done while Hua Ouyang, Hongbo Deng, and Yi Chang were at Yahoo! Labs.

References

[1] Z. Bar-Yossef and N. Kraus, "Context-sensitive query auto-completion," in *Proc. 20th Int. Conf. World Wide Web*, 2011, pp. 107–116.
[2] M. Shokouhi, "Detecting seasonal queries by time-series analysis," in *Proc. 34th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2011, pp. 1171–1172.
[3] M. Shokouhi and K. Radinsky, "Time-sensitive query auto-completion," in *Proc. 35th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2012, pp. 601–610.
[4] S. Whiting and J. M. Jose, "Recent and robust query auto-completion," in *Proc. 23rd Int. Conf. World Wide Web*, 2014, pp. 971–982.
[5] M. Shokouhi, "Learning to personalize query auto-completion," in *Proc. 36th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2013, pp. 103–112.
[6] Y. Li, A. Dong, H. Wang, H. Deng, Y. Chang, and C. Zhai, "A two-dimensional click model for query auto-completion," in *Proc. 37th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2014, pp. 455–464. [Online]. Available: http://doi.acm.org/10.1145/2600428.2609571
[7] H. Bast, D. Majumdar, and I. Weber, "Efficient interactive query expansion with complete search," in *Proc. 16th ACM Conf. Conf. Inf. Knowl. Manage.*, 2007, pp. 857–860.
[8] R. W. White and G. Marchionini, "Examining the effectiveness of real-time query expansion," *Inf. Process. Manage.*, vol. 43, no. 3, pp. 685–704, 2007.
[9] H. Bast and I. Weber, "Type less, find more: Fast autocompletion search with a succinct index," in *Proc. 29th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2006, pp. 364–371.
[10] 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. 27th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2004, pp. 321–328.
[11] A. Kulkarni, J. Teevan, K. M. Svore, and S. T. Dumais, "Understanding temporal query dynamics," in *Proc. 4th ACM Int. Conf. Web Search Data Mining*, 2011, pp. 167–176.
[12] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Mach. Learn.*, vol. 47, no. 2/3, pp. 235–256, 2002.
[13] S. Bubeck and N. Cesa-Bianchi, "Regret analysis of stochastic and nonstochastic multi-armed bandit problems," *Found. Trends Mach. Learn.*, vol. 5, no. 1, pp. 1–122, 2012. [Online]. Available: http://dx.doi.org/10.1561/2200000024
[14] F. Radlinski, R. Kleinberg, and T. Joachims, "Learning diverse rankings with multi-armed bandits," in *Proc. 25th Int. Conf. Mach. Learn.*, 2008, pp. 784–791.
[15] Y. Yue and T. Joachims, "Interactively optimizing information retrieval systems as a dueling bandits problem," in *Proc. 26th Annu. Int. Conf. Mach. Learn.*, 2009, pp. 1201–1208.
[16] Y. Yue, J. Broder, R. Kleinberg, and T. Joachims, "The k-armed dueling bandits problem," *J. Comput. Syst. Sci.*, vol. 78, no. 5, pp. 1538–1556, 2012.
[17] K. Hofmann, S. Whiteson, and M. de Rijke, "Balancing exploration and exploitation in listwise and pairwise online learning to rank for information retrieval," *Inf. Retrieval*, vol. 16, no. 1, pp. 63–90, 2013.
[18] T. Moon, W. Chu, L. Li, Z. Zheng, and Y. Chang, "An online learning framework for refining recency search results with user click feedback," *ACM Trans. Inf. Syst.*, vol. 30, no. 4, 2012, Art. no. 20.
[19] L. Li, W. Chu, J. Langford, and X. Wang, "Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms," in *Proc. ACM Int. Conf. Web Search Data Mining*, 2011, pp. 297–306.

[20] M. Sloan and J. Wang, "Iterative expectation for multi period information retrieval," in *Proc. ACM Int. Conf. Web Search Data Mining Workshop Web Search Click Data*, 2013.
[21] S. Chaudhuri and R. Kaushik, "Extending autocompletion to tolerate errors," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2009, pp. 707–718.
[22] S. Ji, G. Li, C. Li, and J. Feng, "Efficient interactive fuzzy keyword search," in *Proc. 18th Int. Conf. World Wide Web*, 2009, pp. 371–380.
[23] J. C. Gittins, "Bandit processes and dynamic allocation indices," *J. Roy. Statist. Soc. Series B (Methodoogical)*, vol. 41, pp. 148–177, 1979.
[24] T. L. Lai and H. Robbins, "Asymptotically efficient adaptive allocation rules," *Advances Appl. Math.*, vol. 6, no. 1, pp. 4–22, 1985.
[25] R. Agrawal, "Sample mean based index policies with $O(\log n)$ regret for the multi-armed bandit problem," *Advances Appl. Probability*, vol. 27, pp. 1054–1078, 1995.
[26] W. R. Thompson, "On the likelihood that one unknown probability exceeds another in view of the evidence of two samples," *Biometrika*, vol. 25, pp. 285–294, 1933.
[27] D. Russo and B. Van Roy, "Learning to optimize via posterior sampling," *Math. Operations Res.*, vol. 39, pp. 1221–1243, 2014.
[28] S. Agrawal and N. Goyal, "Analysis of thompson sampling for the multi-armed bandit problem," in *Proc. 25th Annu. Conf. Learning Theory*, 2012 pp. 39.1–39.26.
[29] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire, "The nonstochastic multiarmed bandit problem," *SIAM J. Comput.*, vol. 32, no. 1, pp. 48–77, 2002.
[30] L. Li, W. Chu, J. Langford, and R. E. Schapire, "A contextual-bandit approach to personalized news article recommendation," in *Proc. 19th Int. Conf. World Wide Web*, 2010, pp. 661–670.
[31] W. Chu, L. Li, L. Reyzin, and R. E. Schapire, "Contextual bandits with linear payoff functions," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2011, pp. 208–214.
[32] P. Rigollet and A. Zeevi, "Nonparametric bandits with covariates," in *Proc. Conf. Learn. Theory*, 2010, pp. 54–66.
[33] A. Slivkins, "Contextual bandits with similarity information," in *Proc. Conf. Learn. Theory*, 2011, pp. 679–702.
[34] R. Kleinberg, A. Slivkins, and E. Upfal, "Multi-armed bandits in metric spaces," in *Proc. 40th Annu. ACM Symp. Theory Comput.*, 2008, pp. 681–690.
[35] R. Kleinberg and A. Slivkins, "Sharp dichotomies for regret minimization in metric spaces," in *Proc. 21st Annu. ACM-SIAM Symp. Discrete Algorithms*, 2010, pp. 827–846.
[36] P. Auer, R. Ortner, and C. Szepesvári, "Improved rates for the stochastic continuum-armed bandit problem," in *Learning Theory*. Berlin, Germany: Springer, 2007, pp. 454–468.
[37] S. Bubeck, R. Munos, G. Stoltz, and C. Szepesvari, "X-armed bandits," *J. Mach. Learn. Res.*, vol. 12, pp. 1655–1695, 2011.
[38] T. Uchiya, A. Nakamura, and M. Kudo, "Algorithms for adversarial bandit problems with multiple plays," in *Algorithmic Learning Theory*. Berlin, Germany: Springer, 2010, pp. 375–389.
[39] S. Kale, L. Reyzin, and R. E. Schapire, "Non-stochastic bandit slate problems," in *Proc. Advances Neural Inf. Process. Syst.*, 2010, pp. 1054–1062.
[40] F. Radlinski and T. Joachims, "Active exploration for learning rankings from clickthrough data," in *Proc. 13th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2007, pp. 570–579.
[41] M. Streeter and D. Golovin, "An online algorithm for maximizing submodular functions," in *Proc. Advances Neural Inf. Process. Syst.*, 2009, pp. 1577–1584.
[42] A. Slivkins, F. Radlinski, and S. Gollapudi, "Ranked bandits in metric spaces: Learning diverse rankings over large document collections," *J. Mach. Learn. Res.*, vol. 14, no. 1, pp. 399–436, 2013.
[43] K. Hofmann, A. Schuth, S. Whiteson, and M. de Rijke, "Reusing historical interaction data for faster online learning to rank for IR," in *Proc. 6th ACM Int. Conf. Web Search Data Mining*, 2013, pp. 183–192.
[44] O. Chapelle and L. Li, "An empirical evaluation of thompson sampling," in *Proc. Advances Neural Inf. Process. Syst.*, 2011, pp. 2249–2257.
[45] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions," *Math. Program.*, vol. 14, no. 1, pp. 265–294, 1978.
[46] Y. Song, D. Zhou, and L.-W. He, "Query suggestion by constructing term-transition graphs," in *Proc. 5th ACM Int. Conf. Web Search Data Mining*, 2012, pp. 353–362.
[47] D. Agarwal, B.-C. Chen, and P. Elango, "Spatio-temporal models for estimating click-through rate," in *Proc. 18th Int. Conf. World Wide Web*, 2009, pp. 21–30.
[48] O. Chapelle and Y. Zhang, "A dynamic Bayesian network click model for web search ranking," in *Proc. 18th Int. Conf. World Wide Web*, 2009, pp. 1–10.
[49] A. Gelman, J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin, *Bayesian Data Analysis*. Boca Raton, FL, USA: CRC Press, 2013.



Yingfei Wang received the bachelor's degree from the Department of Computer Science, Peking University, China, and the PhD degree in computer science from Princeton University, in 2017. She is an assistant professor in the Foster School of Business, University of Washington. Her research interests include machine learning, stochastic optimization, dynamic programming with broad application in Web search, e-commerce, social networks, health IT, revenue management, and natural language processing.

Hua Ouyang received the MPhil degree from the Chinese University of Hong Kong, in 2007, where he worked on large scale multi-modal classification for human speech and computer vision and the PhD degree in computer science from the School of Computational Science and Engineering, College of Computing, Georgia Tech, in 2013, where he focused on designing and analyzing large scale machine learning, stochastic optimization, computational geometry, information retrieval, and recommender systems. He is a senior research engineer at Apple, leading Siri Search Relevance. He was a research scientist at Yahoo Labs. His research is focused on the theory and algorithms for scalable machine learning, and applications in search and recommendation. He worked as an intern at IBM T. J. Watson Research Center, Hawthorne, New York, in 2010, where he worked on large-scale image retrieval and object categorization. He was the recipient of the 2010 best student paper award from the Statistical Computing Section, American Statistical Association. He is a member of the IEEE.

Hongbo Deng received the PhD degree from the Department of Computer Science and Engineering, Chinese University of Hong Kong, in 2009. He is a research scientist and engineer with rich experience in information retrieval, data mining, machine learning, and natural language processing. He is a software engineer at Google. Before that, he was a senior research scientist at Yahoo Labs. He also worked as a research scientist in the Department of Computer Science, University of Illinois at Urbana-Champaign.

Yi Chang is a professor with the Jilin University in China. He is also a technical vice president of Huawei Research, where he is leading Huawei's Poisson Lab. He has broad research interests on information retrieval, data mining, machine learning, and natural language processing. He has published more than 100 research papers in premium conferences or journals, and he is an associate editor of the *IEEE Transactions on Knowledge and Data Engineering*. He is a senior member of the IEEE.