Classifying User Search Intents for Query Auto-Completion

Jyun-Yu Jiang
Department of Computer Science
University of California, Los Angeles
California 90095, USA
jyunyu.jiang@gmail.com

Pu-Jen Cheng
Department of Computer Science
National Taiwan University
Taipei 10617, Taiwan
pjcheng@csie.ntu.edu.tw

ABSTRACT

The function of query auto-completion[1] in modern search engines is to help users formulate queries fast and precisely. Conventional context-aware methods primarily rank candidate queries[2] according to term- and query- relationships to the context. However, most sessions are extremely short. How to capture search intents with such relationships becomes difficult when the context generally contains only few queries. In this paper, we investigate the feasibility of discovering search intents within short context for query auto-completion. The class distribution of the search session (i.e., issued queries and click behavior) is derived as search intents. Several distribution-based features are proposed to estimate the proximity between candidates and search intents. Finally, we apply learning-to-rank to predict the user's intended query according to these features. Moreover, we also design an ensemble model to combine the benefits of our proposed features and term-based conventional approaches. Extensive experiments have been conducted on the publicly available AOL search engine log. The experimental results demonstrate that our approach significantly outperforms six competitive baselines. The performance of keystrokes is also evaluated in experiments. Furthermore, an in-depth analysis is made to justify the usability of search intent classification for query auto-completion.

Keywords

query auto-completion; short context; search intent classification

1. INTRODUCTION

Query auto-completion is one of the most popular features implemented by modern search engines. The goal of query completion is to accurately predict the user's intended query with only few keystrokes (or even without any keystroke), thereby helping the user formulate a satisfactory query with less effort and less time. For instance, users can avoid misspelling and ambiguous queries with query completion services. Hence, how to capture user's search intents for precisely predicting the query one is typing is very important.

The context information, such as previous queries and click-through data in the same session[3], is usually utilized to capture user's search intents. It also has been extensively studied in both query completion [1, 21, 35] and suggestion [18, 25, 37]. Previous work primarily focused on calculating the relationship between candidates[2] and the context. Nevertheless, most of real query sessions are too short to identify such relations. In the AOL search engine log, more than 67% of multi-query sessions consist of only two queries; and only less than 13% of sessions have been constructed with more than three queries. The context consists of only single query if we would like to predict the last query of a 2-query session. Some works rank the candidates based on query similarity [1,35] or user reformulation behavior [21] such as adding and reusing terms in the context. However, the number of terms in a query is very limited (average 2.3 terms in web search [10]), so the single-query context might be less likely to contain the identical terms to candidates. Moreover, queries might belong to similar topics, even though they utilize distinct and inconsistent terms. Other works [18, 25, 37] model query or term dependencies along whole search sessions, but such approaches will deteriorate to the naïve adjacent frequency one when there is only single query in the context. Consequently, short context might cause sparseness problems and ambiguous dependencies.

We then take a query session obtained from the AOL search engine logs to explain in more details: *"reverse address"* → *"white pages."* The query *"reverse address"* represents that the user would like to find a person's profile by an address, and it is also the purpose of the website *"white pages."* Suppose the last query, i.e., *"white pages,"* is the intended query, and the prefix is w. Its preceding one query serves as the context. The generated queries from conventional context-aware methods, such as *"www.google.com"* (from [1,35]) and *"walmart"* (from [18,21]), are not satisfactory in our experiments for the following reasons: (1) the actual query, i.e., "white pages," does not repeat any terms that appear in previous queries [1,21]; (2) the query dependencies between "reverse address" and "white pages" do not

[1]For simplicity, *query auto-completion* is hereinafter referred to *query completion* in this paper.
[2]Candidate queries are those queries that start with the prefix entered by a user in a search box.
[3]A query session is a sequence of queries submitted in a short time period with the same information need.

ICTIR '16, September 12–16, 2016, Newark, Delaware, USA.

DOI: http://dx.doi.org/10.1145/2970398.2970400

appear in the training data [18, 35]. That is, terms in the context might not provide enough information to predict accurately. One possible way to overcome the limitation and boost the performance is to discover the implication of the user's search intents in the query session. If the system can understand the high-level concepts of search intents, it would be better capable of recognizing the intended query.

Based on click-through data and external knowledge, classifying the whole search session, including queries and user click behavior, into a predefined set is a direct and effective way to discover the implication of user's search intents. It benefits not only query classification [24, 34] but also various applications such as relevance ranking [4]. If the search session and candidates are well classified and represented as search intents, the candidates can be ranked by their similarity to representative intents of the search session. For example, the search intents of *"white pages"* and *"reverse address"* are close to each other because they are all classified into the ODP[4] [14] category "Reference." It is reasonable because the website *"white pages"* is actually utilized to finding people, which is the purpose of *"reverse address."* However, mapping the search sessions into categories has been less investigated in query completion. Although some works on query suggestion [25,37] have attempted to cluster or classify queries into several concepts with offline training click-through data, we further consider user's online click behavior so that the classification results can effectively reflect current search intents.

Moreover, search intent classification can be also effective in longer sessions because the user might change her mind. In other words, queries submitted earlier might be less likely to share similar search intents to latter queries. Take a session with 13 queries obtained from AOL logs as an example: *"shoes"* → *"timberland"* → *"adidas"* → ⋯ → *"cheetos"* → *"doritos."* The user originally searched for shoes and related manufacturers; however, she finally sought for snacks. If the system exploits earlier queries, it might obtain incorrect messages. To put it differently, if the latter queries can provide enough information and search intent classification can capture user's search intents more accurately based on the latter queries, the prediction may be more effective. Although some works [26,39] attempt to discover the significant changes of search intents, none of them can directly predict the query most likely to be typed for completion.

In this paper, we focus on discovering users' search intents from search sessions by query classification. Given a prefix and its context together with a set of candidate queries that start with the prefix, the goal of this paper is learning to rank the candidate queries[5], so that the top-ranked queries are more likely to be the query the user is typing. The context includes previous queries and their click-through data in the same session. We first classify sessions and queries into a set of categories to form class distributions. Two kinds of class distributions including URL-class and query-class are considered here. Based on class distributions, we further derive the session-class distribution for the whole search session from three different aspects. For each candidate, eight distribution-based features are proposed to estimate its closeness to the context. Our method finally ranks the candidate queries based on a total of 22 distribution-based features (1 query feature and 7 session/query-session features for three aspects of the session-class distribution). Furthermore, we propose to combine term-based conventional approaches and our proposed features into an ensemble model to obtain knowledge from different aspects and achieve better performance.

Extensive experiments have been conducted on the publicly available AOL search engine log. The experimental results reveal that prediction performance can be significantly improved based on our method, compared to six competitive baselines. Moreover, since the main purpose of query completion is to save users' keystrokes, we further examine whether our approach does effectively reduce the number of keystrokes, which is less verified in most of the previous work [1, 21, 35]. Finally, an in-depth analysis of search intent classification is also provided as evidence, which explains why our distribution-based features are effective.

In the rest of this paper, we make a brief review on related work in Section 2 and present our problem and framework for query completion in Section 3. The experimental results and analysis of search intent classification are presented in Sections 4 and 5, respectively. Finally, in Section 6, we give our discussions and conclusions.

2. RELATED WORK

In this section, we first give the background of query completion, and indicate the difference between our approach and previous work. Query suggestion, which is a task close to query completion, is then introduced and compared to our approach. Last but not least, we discuss query classification, which is relevant to search intent classification in this paper.

2.1 Query Auto-Completion

Before ranking queries for query completion, the sufficient candidate queries starting from the given prefix should be generated in advance. Most works on query completion extract candidates from query logs [21,35] and document corpus [6]. Once candidates are generated, they can be ranked for various applications such as web search [1] and product search [12]. In this paper, the candidate queries are generated from query logs.

In the case of query completion in web search, the most common and intuitive approach is to rank candidates according to their past popularity. However, less popular query might be difficult to predict; on the other hand, the general popularity might not be favorable for every situation [36]. To achieve a better performance in prediction, the context information is usually utilized to rank candidates differently. Bar-Yossef and Kraus [1] and Shokouhi [35] measured the similarity to queries in the context. Jiang *et al.* [21] models users' reformulation behavior along the whole search session. Mitra [29] further automatically learns the distributed representation of query reformations by deep neural network models. Some other factors are also taken into account for improving performance. Shokouhi *et al.* [36] and Cai *et al.* [9] ranked candidates differently according to the temporal information. Whiting and Jose [38] predicted the query trends and distributions by a sliding window from the past few days. The personalization is also a possible solution considering users' personal information, such as demographics and search history [9,35].

However, all of previous approaches in query completion

[4]Open Directory Project (ODP) uses a hierarchical ontology scheme for organizing site listings.

[5]In this paper, we focus on the ranking problem only.

utilized only query terms, but not class information of user's search intents along the whole search session. Our solution differs from these works in that we try to derive class distributions of sessions as users' search intents, thereby ranking candidate queries more effectively.

2.2 Query Suggestion

Query suggestion, which has been extensively studied in the field of IR, is closely related to query completion, but the goals of the two tasks are very distinct. Different from saving users' keystrokes by precise prediction in query completion, query suggestion concentrates on discovering relevant queries in relation to the context. However, the intended query that the user is typing might be in the list of suggested queries, so it could be applied in query completion and regarded as the related work.

The context information including query session and click-through data is also usually applied in query suggestion. Huang *et al.* [20] and Fonseca *et al.* [16] suggested queries with high frequency of co-occurrence with queries in the context. Moreover, Boldi *et al.* [7] and Song *et al.* [37] constructed transition graphs of queries and terms for mining the dependencies. Furthermore, some works [18,25] adopted sequential probabilistic models to model the transition probabilities. In addition to query terms, the click-through data is helpful to reveal queries with similar topics. Mei *et al.* [27] applied the random walk on click-through bipartite graphs to discover relevant queries. Liao *et al.* [25] also utilized such graphs to cluster queries into several concepts. To deal with the problems of data sparseness, some works [18,25] allowed the partial matching to the context. Liao *et al.* [25] clustered queries for flexibility. Furthermore, some works [31, 33] applied machine learning frameworks with statistical features.

Compared to prior context-aware query suggestion methods, our approach can well exploit the context information and cope with the sparseness problems. Besides, none of the prevalent approaches to query suggestion attempts to classify user's search intents with the whole search session. Although some works [25,37] classified or clustered each query into a class or a group of relevant queries, they did not pay attention to the whole search sessions including preceding submitted queries and user click behavior, which represent users' actual search intents.

2.3 Query Classification

Query classification aims to classify a query into a predefined class or category. Gabrilovich *et al.* [17] identify queries and ad classes to improve the relevance of advertising. To classify queries, they make a weighted aggregation of the classes of URLs retrieved by search engines. Kang and Kim [22] utilized some IR and NLP measures as the features to classify queries and decide which algorithm for retrieval. With query logs from search engines, Beitzel *et al.* [3] made use of URLs information to pre-process queries into corresponding categories. The task of the 2005 KDD Cup [24] is also akin to query classification. The challenge of task is to classify queries with few training data. Most successful approaches [34] made use of external information such as ODP [14] or WordNet [28]. To enhance retrieval results, Bennett *et al.* [4] adopt ODP information and derive class distributions to improve ranked results. However, none of the previous work predicts users' intended queries with query classification. In our work, we attempt to derive class distributions with clicked URLs and ODP information for extracting distribution-based features of submitted queries and users' search intents.

3. QUERY AUTO-COMPLETION WITH SEARCH INTENT CLASSIFICATION

We first formally define the objective and the notations in this paper. A search context $\langle q_1, q_2, \cdots, q_{T-1} \rangle$ is defined as a sequence of $T-1$ queries issued by a user within a short period (i.e., the search session). Each query has the corresponding click information $\boldsymbol{u_i}$, which is the set of clicked URLs in the search result page of q_i. Suppose a user intends to issue q_T and has already inputs p, i.e., the prefix of q_T in the search box, after the search context. We would like to predict q_T for query completion. Given a search context with $T-1$ preceding queries $\langle q_1, q_2, \cdots, q_{T-1} \rangle$ and a candidate set Q_T matched the prefix x, the goal is to give a ranking of candidates $q'_j \in Q_T$ so that queries in higher positions are more likely to be q_T.

In our work, we propose a supervised approach consisting of two components and leverage class information of users' search intents to query completion. First, we adopt click-through information and the predefined knowledge to derive URL-, query- and session-class distributions. Then several features are introduced to measure some important properties between class distributions of the session and candidates. With the training data collected from search engine logs, almost any learning-to-rank algorithm can be applied to obtain our ranking model. We finally choose *LambdaMART* [8], which is one of the state-of-the-art ranking models.

3.1 Query and Session Classification

During a search session, the user may have her own search intents. When the user determines her search intents, the class of information need may also be implicitly chosen by the user. Although some previous work [25] tried to cluster each query into a "concept" in advance, a query may belong to more than one concept or topic. This is also the reason why we calculate class distributions but classify a query or a session into a single class. In addition, they ignore that some predefined external information such as ODP categories [14] and reading levels [23] might be helpful for predicting user's intended query.

Different from previous work [4], we focus on deriving class distributions of not only queries but search sessions. Based on the URL-class and query-class distributions, we attempt to model session-class distributions from three views for discovering users' search intents. With the class distributions of the candidate query and the session, we can find relations between the candidate query and previous behaviors, including preceding submitted queries and click-through URLs in the session.

Classification Space. With some predefined knowledge, the queries and sessions can be classified into several categories such as topical categories, semantic categories and reading levels [23]. In this paper, ODP categories [14], which is a topical hierarchy structure, are adopted as the classification target. For conducting experiments, we used 16 topical categories from the top level of the ODP structure, which were crawled in December 2014. In our experiments, more than 53% of user clicks are covered by URLs in ODP. The

remaining clicks are dealt by the smoothing techniques for computing a general distribution.

3.1.1 Query-class Distribution

Following the assumption mentioned in [4], we suppose that the query-class distribution is an aggregation over all relevant URLs. Moreover, here we assume that class distribution is only dependent to relevant URLs. Then we can calculate the query-class distributions with only query-URL relevance and URL-class distributions of all relevant URLs. More formally, the probability $P(c \mid q)$ is the query-class distribution for a query q and a class c. The probabilities $P(u \mid q)$ and $P(c \mid u)$ represents the query-URL relevance and the URL-class distribution. By marginalizing over all the relevant URL u of a query q and the above assumption, $P(c \mid q)$ can be re-written as follows:

$$\begin{aligned} P(c \mid q) &= \sum_u P(c \mid u, q) \cdot P(u \mid q) \quad \text{(marginalization)} \\ &= \sum_u P(c \mid u) \cdot P(u \mid q) \quad \text{(by assumption)}, \end{aligned}$$

then we can compute $P(u \mid q)$ and $P(c \mid u)$ separately for calculating the query-class distribution.

URL-class Distribution. With different targets of classification, there are various approaches for classifying URLs. Bennett *et al.* [4] and Kim *et al.* [23] built a classifier to estimate the class distribution $P(c \mid u)$ for each category. Baykan *et al.* [2] analyze characters of the URL to identify the category. However, building numerous and complex models for every category and all URLs is too inefficient for query completion, which aims to predict rapidly. In our approach, we assume that URLs which belong to the same host may have similar class distributions. Then we can precompute class distributions of plentiful URL hosts in advance and calculate a prior distribution $P(c)$ for URLs whose hosts are not in the pre-computed list with the smoothing technique.

The ODP data has been crawled and treated as the "gold-standard" classification results. Then we can derive class distributions for URLs by the smoothing computation as follows:

$$P(c \mid u) = \frac{Occurs(h(u), c) + m \cdot P(c)}{m + \sum_{c_i} Occurs(h(u), c_i)}.$$

Here m is a parameter for smoothing. The function $h(u)$ extracts the host of URL u. $Occurs(h(u), c)$ calculates how many websites with host $h(u)$ belong to category c in the contents of ODP. The prior distribution $P(c)$ can be computed by normalizing the number of websites in ODP for each category.

Query-URL Relevance. In order to estimate the relevance between queries and URLs, the click-through data is usually treated as a useful information and the relevance signal [11]. With such relevance signal, we attempt to derive the query-URL relevance from the large-scale search engine log. When a URL appearing in the search results of a query was clicked more times in search engine log, we trust that the URL is more relevant to the query than other URLs. However, some URLs are much rare in the log such that the estimated results may be dubious. To handle the sparseness problem, we keep following the assumption that URLs have similar distributions to their hosts. For remaining URLs whose hosts still do not appear in training data with the query, we calculate a prior distribution for them. Then we can calculate query-URL relevance with smoothing technique as follows:

$$P(u \mid q) = \frac{C(h(u), q) + m \cdot P(h(u))}{m + \sum_{h(u)} C(h(u), q)}.$$

$C(h(u), q)$ counts the clicked times of the host $h(u)$ with query q in the log. The prior distribution $P(h(u))$ is calculated by normalizing the number of times URLs which belong to $h(u)$ are clicked in the log.

3.1.2 Session-class Distribution

The session-class distribution can be treated as the distribution of user's information need from the context, including preceding queries and clicked URLs. If we capture the user's search intents precisely, it can be matched to the class distributions of candidate queries. Formally, we extract the session-class distribution $P(c \mid \langle q_1, q_2, \cdots, q_{T-1} \rangle)$ to simulate user's search intents, given the context $\langle q_1, q_2, \cdots, q_{T-1} \rangle$. In our approach, we derive session-class distributions in three different views, including *All Preceding Queries*, *Last Query* and *Local-clicked URLs*.

All Preceding Queries (All). The user's search intents may be shown in the preference of submitted queries. Although queries are able to have various distributions, we try to obtain information from all preceding queries submitted by users. As we know how to estimate the query-class distributions in the above paragraph, we can perform the class distribution of the session as a linear weighted combination as follows:

$$P_{all}(c \mid \langle q_1, q_2, \cdots, q_{T-1} \rangle) = \frac{1}{\sum w_i} \sum w_i P(c \mid q_i).$$

Here w_i are the weight parameters. We assume that the more recent submitted query is more likely relevant to user's search intents, and the weights are monotonically decreasing. In our approach, we make weight function as a linear decay function ($w_i = 1/(T - i)$).

Last Query (Last). In a long session with many queries, the search intents of latter queries may be changed and different from the original ones. The last query (i.e., the most recent query in the context) could be treated as a sign of user's recent search intents. In this view, we assign the distribution of the last query in the context as the session-class distribution as follows:

$$P_{last}(c \mid \langle q_1, q_2, \cdots, q_{T-1} \rangle) = P(c \mid q_{T-1}).$$

Local-clicked URLs (Local). In addition to queries in the context, the clicked-through information is also a important information. As a relevance signal [11], a URL clicked in the session is more authentic than click-through data in the log. Also, a query may belong to more than one class or concept, the local click-through information helps us to identify the correct distribution of current session more precisely. To derive session-class distributions constructed by local-clicked URLs, we first define the local query-URL relevance. We treat whole context as a new "query" and a local log to calculate the relevance probability as follows:

$$P_{local}(u \mid \langle q_1, q_2, \cdots, q_{T-1} \rangle) = \frac{C_{local}(h(u)) + m \cdot P(h(u))}{m + \sum_{h(u)} C_{local}(h(u))},$$

where the function $C_{local}(h(u))$ counts the clicked times of the host $h(u)$ in the context. To avoid biasing to rare

situations, we adopt the prior distribution to smooth the relevance score. Then we collect click-through URLs u_i in the session as a set of URLs $\boldsymbol{u}$, and calculate the session-class distribution as follows:

$$P_{local}(c \mid \langle q_1, q_2, \cdots, q_{T-1} \rangle) = \sum_{u_i \in \boldsymbol{u}} P(c \mid u_i) P_{local}(u_i \mid \langle q_1, q_2, \cdots, q_{T-1} \rangle)$$

Three kinds of session-class distributions show different views to observe a search session and capture the user's search intents. We then use them to extract distribution-features with query-class distributions of candidate queries.

3.2 Distribution-based Features

Based on the class distributions of session and candidate queries, we then define various features and apply machine learning algorithms to discover and model interactions with other features. Features for learning to rank can be categorized into three classes, including query features, session features and query-session features. The query features and session features are extracted from only class distributions of candidate queries and the session. The query-session features consider both candidate queries and the session.

Query/Session Class Entropy (QCE/SCE). In our approach, class entropy measuring the entropy of the class distribution is the only feature of query feature and session feature. The class entropy is helpful to identify the ambiguity of a candidate query or the session information. Queries and sessions with lower entropy (i.e., belonging to fewer topics) might be submitted by users trying to narrow the search results to such topics.

Next, we model the query-session features aiming to measure how well a candidate query matches to the search session with the class distribution.

Class Match (CM). Among the class distribution, we can find the most likely class of a candidate query and the session. Users may submit queries in the identical class of the session and previous context. So we treat whether the most likely classes of the query and the session are matched as the *ClassMatch* feature.

ArgMaxOdds (AMO) & MaxOdds (MO). We then try to measure how well the session-class distribution matches the most likely class of the candidate query. Define the most likely class of the candidate query q as $c_q^* = \underset{c}{\text{argmax}} P(c \mid q)$, we calculate the *ArgMaxOdds* feature as follows:

$$P(c_q^* \mid q) \log \frac{P(c_u^* \mid \langle q_1, q_2, \cdots, q_{T-1} \rangle)}{P(c_q^*)}.$$

Here the prior $P(c_q^*)$ is adopted to weight the class distribution as the probability of observing that class. By relaxing the *ArgMaxOdds* feature, we calculate the *MaxOdds* over all classes and find the maximum measure as follows:

$$\max_c P(c \mid q) \log \frac{P(c \mid \langle q_1, q_2, \cdots, q_{T-1} \rangle)}{P(c)}.$$

KL Divergence (KL) & Cross Entropy (CE). The previous two features encode information of only one class, which is the most likely class or the class with maximum measure. We adopt *KL Divergence* to estimate how a candidate query matches the session upon the entire class distribution, that is:

$$\sum_c P(c \mid q) \log \frac{P(c \mid q)}{P(c \mid \langle q_1, q_2, \cdots, q_{T-1} \rangle)},$$

and we handle the special cases $0 \log 0$ and zero denominator as 0. *Cross Entropy* is the other feature similar to *KL Divergence*. It measures how topics represented in a candidate query are covered by the session. The formula of *Cross Entropy* is shown as follows:

$$-\sum_c P(c \mid q) \log P(c \mid \langle q_1, q_2, \cdots, q_{T-1} \rangle).$$

Distribution Similarity (DS). Each distribution has probabilities in all classes. We treat such distribution of a query or a session as a n-dimensional vector, which n is the number of classes. With the vector representation, we calculate cosine similarity between two vectors and measure the relation of the candidate query and the session.

Table 1: Eight distribution-based features and used information. Note that features related to sessions will be extracted three times for three aspects of session-class distribution (See Section 3.1.2), so there are totally 22 features in the model.

Feature	Query	Session	# in Model
Query Class Entropy (QCE)	√		1
Session Class Entropy (SCE)		√	3
Class Match (CM)	√	√	3
ArgMaxOdds (AMO)	√	√	3
MaxOdds (MO)	√	√	3
KL Divergence (KL)	√	√	3
Cross Entropy (CE)	√	√	3
Distribution Similarity (DS)	√	√	3

Table 1 gives the summary of proposed features and their used information. Note that we apply three aspects of session-class distribution as described in Section 3.1.2 to extract session-related features. Hence, there are finally 22 features in the model (one query feature and seven session/query-session features for each aspect).

3.3 Ensemble with Conventional Approaches

Our proposed approach mainly focus on utilizing click-though data and external knowledge to classify queries into certain categories. In other words, the similarity based on query terms, which is the principal idea of conventional approaches, does not be considered in our model. Hence, the performance may be further improved if there is an ensemble model to combine our approach and other conventional approaches. Moreover, traditional term-based methods may favor longer search sessions so that more information can be found in the context. Conversely, class distribution extracted from short context in our approach may be able to more precisely represent users' search intents. That is, an ensemble model of two solutions may simultaneously benefit short and longer search sessions.

As the first study, we simply adopt all 43 features proposed in [21] as the other side of the ensemble model because this is one of the state-of-the-art approaches based on query terms in the context. Here *LambdaMART* [8] is utilized again to train an ensemble model to incorporate features

proposed in this paper and [21]. Note that the length of the search session is one of the utilized features, so theoretically the ranking model will learn the strength of each model and perform well with both long and short search sessions.

4. EXPERIMENTS

In this section, we conduct extensive experiments on the large-scale datasets to verify the performance of our approach predicting user's intended query in the session.

4.1 Datasets and Experimental Settings

We use the public AOL log [32] in our experiments. The data comprises sampled queries submitted to the AOL search engine from 1 March, 2006 to 31 May, 2006. The query log is first segmented into sessions with a 30-minute threshold as the session boundary. To remove misspelled and rare queries, queries appearing less than 10 times in the whole log are dropped. Sessions with only single query are discarded because there must be at least one preceding query as the context for context-aware methods. After removing rare queries and single-query sessions, there are 1,359,760 sessions and 141,975 unique queries in total. Here we notice that the number of unique queries is inconsistent with [35]. It might be caused by a different procedure, which is not mentioned in [35], to remove short sessions. For each session with T queries, we treat each query q_i from the second position (i.e., $i \geq 2$) as the ground truth, which is the intended query we want to predict. The context is composed of $i-1$ preceding queries $\langle q_1, q_2, \cdots q_{i-1} \rangle$ and their clickthrough information. Note that the cleaning process in this paper is consistent with previous work [35, 36].

After cleaning, the data is then partitioned into two sets. The first 2-month data is used for training. The remaining 1-month data is for testing. Finally, we collect 891,145 sessions as our training set. We use query frequency to generate candidate set. After filtering out those queries not starting from the prefix, the top-10 queries ranked by frequency form our candidate queries. The prefix is set to be the first character of q_T. Note that most of our settings are consistent with previous work, including the removal of rare queries [35], the way to generate candidates [21, 35].

To evaluate performance on tasks with different lengths of context, the testing sessions are divided into three subsets, including "Short Context" (1 query), "Medium Context" (2 to 3 queries) and "Long Context" (4 or more queries). Besides, we tune our LambdaMART model with parameters of 1,000 decision trees across all experiments; and we set the smoothing parameter m as 0.04 after tuning.

4.2 Competitive Baselines

To compare our approach with others, the following conventional methods are adopted as the baselines:

Most Popular Completion (or MPC). MPC is a *Maximum Likelihood Estimation* approach. It simply ranks candidates by their frequencies.

Hybrid Completion (or Hyb.C). Hyb.C proposed in [1] is a context-sensitive query completion method, which considers both context information and the popularity. The ranking of a candidate is determined by linear combination of its popularity (i.e., MPC) and similarity to recent queries.

Personalized Completion (or Per.C). Per.C proposed in [35] is a personalized query completion method, which considers users' personal information including submitted queries and demographics information. In this paper, we implement this method with short-term history (i.e., queries in the session), long-term history (i.e., queries in the user's search history) and query frequency.

Query-based VMM (or QVMM). QVMM proposed in [18] is a context-aware query suggestion method. It learns the probability of query transition along the sessions with the variable memory Markov model, which is an extension of Markov chain model.

Concept-based VMM (or CACB). CACB proposed in [25] is a concept-based context-aware query suggestion method. It clusters queries into several concepts to overcome the sparseness problem. After mapping each query to its corresponding concept, CACB tries to model the concept transition, instead of query transition, with variable memory Markov model.

Reformulation-based Completion (or RC). RC proposed in [21] models users' reformulation behavior during search sessions. It considers three kinds of reformulation-related features, including term-level, query-level and session-level features. The features carefully capture how users change preceding queries along the query sessions. We use *LambdaMART* [8] to train the model with 43 reformulation-related features.

Note that we do not compare our approach with [38] and [9] because [38] does not apply context information, and the temporal information [9] is not the focus of this paper.

4.3 Evaluation Metrics

In our experiments, *mean reciprocal rank* (MRR) is applied to evaluate the quality of query completion. The *mean reciprocal rank* (MRR) takes account of the ranked position of the ground truth. Given the session context C and the ground truth q_T, the reciprocal rank (RR) of an algorithm A is defined as follows:

$$\mathrm{RR}(A) = \frac{1}{hitrank(A, C, q_T)},$$

where $hitrank(A, C, q_T)$ computes the position of the ground truth ranked by the algorithm A. Thus, MRR can be computed as the mean value of RR for all search sessions in the testing set.

4.4 Overall Performance

Table 2 shows the experimental results over different prefix lengths. Our approach is denoted as CC (Classification-based Completion), and the single context means that we use only one preceding query as the context. Note that although both CC and CC (Single Context) utilize only one query as the context in the short context set, results might be different because the latter exploits only a single query in the training stage. Besides, MRR shown in [35] is higher than our reports because they do not consider sessions whose ground truths are not in the candidate lists. In other words, our evaluation can more effectively reflect the real situations.

Based on the context information, all of context-based methods outperform MPC in most cases. The Hyb.C completion has similar performance to the MPC method for short context. This is because short sessions context too few terms for Hyb.C to match the query to the context. With personal history, the Per.C method has better performance than the Hyb.C method because it can capture personal characteristics. The QVMM and CACB methods outper-

Table 2: The MRR performance of seven methods in the overall testing set and three subsets over different lengths of typed prefixes #p. Our approach is denoted as CC (Classification-based Completion), and the single context denoted as SC consists of only one query for both training and testing. The ensemble model of the RC method [21] and our approach is denoted as RC + CC. All improvements of our methods against [21] are significant differences at 95% level in a paired t-test.

Dataset	#p	MPC	Hyb.C [1]	Per.C [35]	QVMM [18]	CACB [25]	RC [21]	CC	CC (SC)	RC + CC
Overall	1	0.1724	0.1796	0.1935	0.2028	0.1987	0.2049	0.2140	0.2159	0.2245
	2	0.2703	0.2733	0.2770	0.2868	0.2828	0.2841	0.2939	0.2965	0.3024
	3	0.4004	0.4025	0.4026	0.4066	0.4014	0.4122	0.4193	0.4230	0.4369
	4	0.5114	0.5137	0.5129	0.5179	0.5126	0.5244	0.5358	0.5406	0.5562
Short Context (1 Query)	1	0.1540	0.1538	0.1740	0.1858	0.1813	0.1842	0.1966	0.1975	0.2055
	2	0.2523	0.2524	0.2591	0.2704	0.2637	0.2635	0.2792	0.2804	0.2864
	3	0.3819	0.3805	0.3846	0.3912	0.3903	0.3934	0.4065	0.4083	0.4177
	4	0.4939	0.4923	0.4962	0.5042	0.4979	0.5072	0.5243	0.5266	0.5390
Medium Context (2 to 3 Queries)	1	0.2041	0.2233	0.2266	0.2317	0.2288	0.2399	0.2438	0.2449	0.2556
	2	0.3015	0.3094	0.3080	0.3153	0.3111	0.3196	0.3226	0.3241	0.3356
	3	0.4293	0.4320	0.4308	0.4356	0.4316	0.4419	0.4435	0.4455	0.4573
	4	0.5368	0.5392	0.5371	0.5398	0.5373	0.5498	0.5539	0.5565	0.5694
Long Context (4 or more Queries)	1	0.1922	0.2207	0.2155	0.2164	0.2146	0.2284	0.2247	0.2310	0.2439
	2	0.2892	0.3000	0.2953	0.3002	0.2987	0.3076	0.3036	0.3121	0.3182
	3	0.4291	0.4396	0.4303	0.4302	0.4304	0.4397	0.4328	0.4449	0.4592
	4	0.5443	0.5497	0.5439	0.5468	0.5449	0.5546	0.5473	0.5625	0.5801

form previous baseline methods since they model query transitions along the whole search session. The reformulation-based completion (RC) performs the best among all baseline methods because it comprises abundant and diverse user reformulation behavior in different levels.

For all testing datasets, our approach beats the entire six comparative baseline methods over all prefix lengths, and significantly outperforms the RC method at 95% level in a paired t-test. Our approach has the greatest improvement in short context, which are the majority in search logs. This is because our model derives high level information (i.e., class distribution of search intents) beyond the limitation of query terms. Hence, the relationships between candidates and search intents can be discovered with very short context. However, the improvements of our approach gradually decrease when the length of context increases. The performance in the long context is even worse than the RC method. The reason is that former queries or click-through information may represent different search intents from latter ones. In contrast, while considering only a single query in the context, the performance in longer context improves simultaneously. It shows that our features can precisely capture users' search intents in the very limited context and avoid the noises of premature queries. The results are also consistent with [5, 29], which have shown the importance of the immediate query in modelling query sessions.

After combining the RC method [21] and our approach, the ensemble model obviously outperforms each of single model. Compared to two single models, the performance with a character as the prefix achieves a 9.57% improvements against the RC method and a 3.98% improvement against the CC method. It is reasonable because two approaches utilize different information to solve the problem so that the ensemble model can obtain the knowledge from both methods. Moreover, the improvements in the dataset of long context against the CC method are greater than the improvements in the datasets of shorter context. This is because the knowledge from the RC method mends the shortcoming of the CC method on long context and improves the performance. Hence, the ensemble model of term-based methods and classification-based completion methods can actually predict users' intended queries more precisely.

4.5 Feature Effectiveness Analysis

In order to analyze the importance of different features, we adopt the leave-one-out feature selection method to verify the feature effectiveness. Figure 1 shows the result of leave-one-out feature selection. For example, we calculate the performance loss of SCE by removing three features of SCE, and then re-training the model. Here we apply the first character as the prefix. Note that features with higher performance loss are more important to the model. It is worth noticing that QCE is the most effective single feature. The result is also consistent to the previous work [4]. This feature can be treated as a measure of query ambiguity [13], and it implicitly leads to different ranking functions for queries with many intents and only few intents. The features about most likely class, including CM, AMO and MO, have the least importance among query-session features in the model. The reason is that the most likely class only cannot well capture the relationship between candidate queries and user's search intents (i.e., session-class distributions). In contrast, KL and CE are the most important query-session features because they estimate how a candidate query matches the session upon the entire class distribution.

Recall that we derive the session-class distribution from three different views and extract distribution-based features for them separately. We also analyze the importance of these extraction methods. As shown in Fig. 1, the features extracted by local-clicked URLs are the most effective among

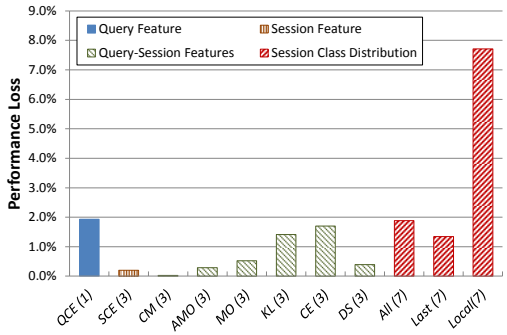
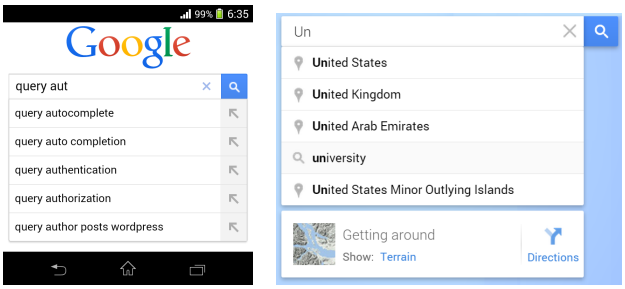


Figure 1: The performance loss of MRR for each individual feature category and each aspect of session-class distribution in leave-one-out feature selection. Note that the numbers in $(\cdot)$ represent the numbers of features in the model. All of abbreviations can be found in Section 3.



(a) Android Smartphone (b) Google Maps

Figure 2: The illustrations of query completion systems on the Android smartphone and the Google Maps.

three views. It can be explained that the local-clicked URLs represent users' information needs correctly. Although we can derive general class distributions for queries in sessions, they may not fit the users' search intents well.

4.6 Keystroke Experiments

Although MRR is widely used as the evaluation measures for query completion, it might not reflect the user's actual demands. Both of two measures are sensitive to different positions of queries, but usually the user only hopes that the query appears in a high position. As shown in previous work [19, 30], the position biases generally exist in query completion and strongly affect its examination. Moreover, users usually can observe only few queries. For example, Figure 2 shows the illustrations of query completion systems on the Android smartphone and the Google Maps. Users can observe only five queries, so other queries ranked lower are meaningless for user experience and evaluation. Actually, users would like use *the least keystrokes* to make the desired query in a higher position. Although the query is ranked in a much low position (and out of the screen) with only one keystroke, the system may be still effective if it can rank the query in a very high position with two keystrokes.

Although this idea has already been proposed in previous work [15], few works utilized such measure to evaluate

Table 3: The keystroke performance of seven methods in the whole testing set. Our approach is denoted as CC (Classification-based Completion).

Measure	No Comp.	MPC	Hyb.C [1]	Per.C [35]
KS@1	11.0034	8.4294	6.8694	6.5761
KS@2	-	6.8625	5.6452	5.5078
KS@3	-	5.9830	4.9616	4.6965
KS@4	-	5.3038	4.5353	4.1793
Measure	**QVMM [18]**	**CACB [25]**	**RC [21]**	**CC**
KS@1	5.8704	6.1135	5.0129	**4.7479**
KS@2	4.1562	4.7813	3.9295	**3.6660**
KS@3	3.7044	4.0173	3.6523	**3.5880**
KS@4	3.6076	3.9138	3.5928	**3.5818**

the actual performance of query completion in a proper way. Here we evaluate our approach and baseline methods according to users' keystrokes. The *keystroke at top-k* (KS@k) is defined as the average keystrokes users spend so that the actual queries can be found in the top-k queries predicted for sessions.

Table 3 shows the keystroke performance of 5 methods in whole testing set. Because the average length of queries is about 11 characters, a user may spend 11 keystrokes averagely in a system without query completion. For query completion methods, it is obvious that all methods can save keystrokes. The results are also consistent to the performance shown in Table 2. Moreover, our approach still outperforms all comparative methods in the keystroke performance. For our approach, it can save 56.85% keystrokes for users. It represents that our approach is actually useful for user to save their time in typing queries.

5. ANALYSIS OF SEARCH INTENT CLASSIFICATION

In this section, we analyze the classes of users' search intents and further explain why our distribution-based features are effective to the query auto-completion.

5.1 Most Likely Class

Recall that the most likely class of the candidate query q is $c_q^* = \underset{c}{\text{argmax}} P(c \mid q)$ and so is session's, we can compute the most likely class of a query or a session from derived class distribution. Figure 3 shows the percentage of sessions whose queries contain different number of topics. We can see that the queries in most sessions contain only a few classes for every length of sessions. That is, the candidates belonging to these few classes may have larger probability to be submitted as the next query. In other words, there are generally a small number of topics focused in a session. It is also the reason why our distribution-based features capturing the class distribution are effective in our experiments.

Next, we would like to know how the distribution-based features work in sessions within different length. Figure 4 shows the percentage of sessions whose most likely classes are identical to the last queries. Among three views of extracting session-class distribution, the view of local-clicked URLs has the highest percentages. It is consistent to the feature effectiveness analysis in Section 4.5. The percentages of shorter sessions are higher than longer ones in all

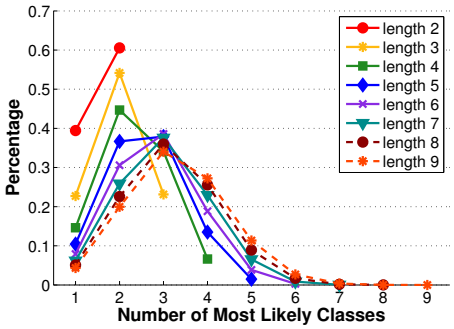


Figure 3: The percentage of sessions whose queries contain different number of most likely classes.

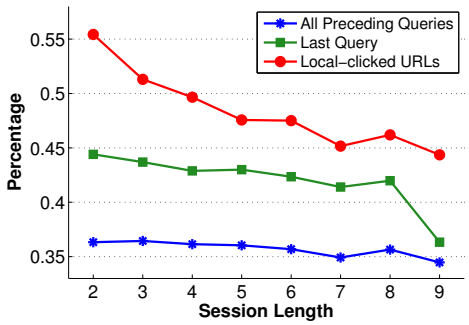


Figure 4: The percentage of sessions with the identical most likely classes to the last queries over different session length.

three views. The reason might be that shorter context has fewer noises to be matched more precisely. It also explains why our approach has greater improvements with shorter context. However, there are more than 40% sessions whose the most likely classes are not identical to the last queries. This might be the reason why features of the most likely class are not so important in the model, as shown in Fig. 1.

5.2 Distribution Matching

Beyond the most likely class, this section discusses the matching between candidate queries and the search session. As the analysis in Section 4.5, *KL Divergence* and *Cross Entropy* are the most effective features among query-session features. That is, estimating how a candidate query matches the session upon entire class distributions is useful to identify the actual submitted query (or the ground truth) in the candidate query list. Table 4 shows the percentages of incorrect candidate queries whose *KL Divergence* and *Cross Entropy* are larger than the ground truths of sessions over three views of the session distribution. For a candidate query, lower feature values represent that the distribution of the query is more similar to the session. Both features can well distinguish the ground truths from other incorrect candidate queries. The *KL Divergence* values of submitted queries are less than at least 64% other incorrect candidates over each view of the session distribution. Among three views of the session distribution, features extracted by local-clicked URLs perform best on identifying the correct queries. More than 80% incorrect candidates are ranked lower than the

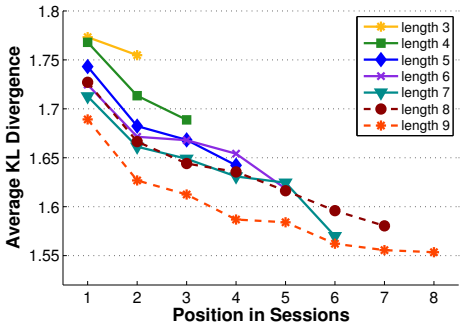


Figure 5: Average *KL Divergence* between local-clicked URLs and the last query over different position in sessions.

ground truths in *KL Divergence*. It is also consistent to the results in Section 4.5. The results show that the derived class distribution can well represent the characteristics of queries and sessions. Therefore, the distribution matching can advantageously identify the actual submitted query.

Table 4: Percentages of incorrect candidate queries whose features are larger than the ground truths of sessions over three views of the session distribution.

Feature	All Preceding Queries	Last Query	Local-clicked URLs
KL Divergence	0.6641	0.6406	0.8118
Cross Entropy	0.5994	0.5688	0.7072

5.3 Local-clicked URLs

As the relevance signals in the search session, local-clicked URLs can provide copious information about users' search intents. The features extracted by such URLs are also effective in our model as shown in Section 4.5. This section discusses why local-clicked URLs are so useful.

The hosts of URLs are worth a thought because URLs under a same host might share similar search intents. For each session whose the last query has clicked URLs (i.e., $\boldsymbol{u_T} \neq \emptyset$), there are 11.89% sessions which have URLs of the identical hosts within previous queries. Even though the space of URL hosts is very sparse and contains hundreds of thousands hosts, there are still numerous sessions with identical hosts. It shows that there will be more sessions with URLs of similar search intents, not only URLs with the identical hosts. It might also be the reason that the features of local-clicked URLs are so useful.

Next, we analyze the distribution matching between the last queries and local-clicked URLs in each position. Figure 5 shows the average *KL Divergence* between local-clicked URLs and the last query over different position in sessions. Note that the last queries are distinct in sessions of different session lengths, so *KL Divergence* values for different lengths are not comparable. From Fig. 5, it is worth noticing that the average *KL Divergence* decreases as the position increases. The class distribution of more recently clicked URLs are more similar to the query. It is also consistent to the results in Section 4. Shorter context might contain more clear information and obtain better improvements.

6. DISCUSSIONS AND CONCLUSIONS

In this paper, we proposed a novel approach for query completion aiming to predict users' next intended queries. Different from conventional context-aware methods discovering term- and query-level relationships between sessions and candidates, we classify users' search intents behind search sessions by deriving class distributions for predicting queries more precisely. We focus on discovering search intents with short context because most of search sessions are very short and limited. The results of extensive experiments demonstrate that our approach significantly outperforms several existing context-aware approaches. Such improvement is consistent across different context lengths. The reasons are as follows: (1) our classification-based model requires less training data. There are only few categories for classification. The sparseness problem can be solved; (2) class distributions of search sessions can well capture users' search intents. Several distribution-based features matching search sessions and candidate queries are considered for query completion. (3) with a high-level aspect, users' search intents can be well captured in the limited context. The results of analysis in Section 5 also support that our features are helpful and important. Moreover, the results of keystroke experiments in Section 4.6 show that our approach can effectively reduce users' keystrokes to complete queries.

As future work, users' reformulation behavior may further interact with search intent classification although we have proved their ensemble model has a excellent performance in Section 4. For example, some users' behavior may imply the change of search intents, then we can dynamically modify the classification-based model.

7. REFERENCES

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

[2] E. Baykan, M. Henzinger, L. Marian, and I. Weber. Purely url-based topic classification. In *WWW '09*, pages 1109–1110. ACM, 2009.

[3] S. M. Beitzel, E. C. Jensen, D. D. Lewis, A. Chowdhury, and O. Frieder. Automatic classification of web queries using very large unlabeled query logs. *ACM TOIS*, 25(2):9, 2007.

[4] P. N. Bennett, K. Svore, and S. T. Dumais. Classification-enhanced ranking. In *WWW '10*, pages 111–120. ACM, 2010.

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

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

[7] P. Boldi, F. Bonchi, C. Castillo, D. Donato, and S. Vigna. Query suggestions using query-flow graphs. In *WSCD '09*, pages 56–63. ACM, 2009.

[8] C. J. Burges, K. M. Svore, P. N. Bennett, A. Pastusiak, and Q. Wu. Learning to rank using an ensemble of lambda-gradient models. *JMLR*, 14:25–35, 2011.

[9] F. Cai, S. Liang, and M. de Rijke. Time-sensitive personalized query auto-completio. In *CIKM '14*, pages 1599–1608. ACM, 2014.

[10] C. Carpineto and G. Romano. A survey of automatic query expansion in information retrieval. *ACM Computing Surveys (CSUR)*, 44(1):1, 2012.

[11] O. Chapelle and Y. Zhang. A dynamic bayesian network click model for web search ranking. In *WWW '09*, pages 1–10. ACM, 2009.

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

[13] K. Collins-Thompson and P. N. Bennett. Estimating query performance using class predictions. In *SIGIR '09*, pages 672–673. ACM, 2009.

[14] N. C. Corporation. Open directory project. http://www.dmoz.org, 2013.

[15] H. Duan and B.-J. P. Hsu. Online spelling correction for query completion. In *WWW '11*, pages 117–126. ACM, 2011.

[16] B. Fonseca, P. Golgher, B. Pôssas, B. Ribeiro-Neto, and N. Ziviani. Concept-based interactive query expansion. In *CIKM '05*, pages 696–703. ACM, 2005.

[17] E. Gabrilovich, A. Broder, M. Fontoura, A. Joshi, V. Josifovski, L. Riedel, and T. Zhang. Classifying search queries using the web as a source of knowledge. *ACM TWEB)*, 3(2):5, 2009.

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

[19] K. Hofmann, B. Mitra, F. Radlinski, and M. Shokouhi. An eye-tracking study of user interactions with query auto completion. In *CIKM '14*, pages 549–558. ACM, 2014.

[20] C.-K. Huang, L.-F. Chien, and Y.-J. Oyang. Relevant term suggestion in interactive web search based on contextual information in query session logs. *JASIST*, 54, 2003.

[21] J.-Y. Jiang, Y.-Y. Ke, P.-Y. Chien, and P.-J. Cheng. Learning user reformulation behavior for query auto-completion. In *SIGIR '14*. ACM, 2014.

[22] I.-H. Kang and G. Kim. Query type classification for web document retrieval. In *SIGIR '03*, pages 64–71. ACM, 2003.

[23] J. Y. Kim, K. Collins-Thompson, P. N. Bennett, and S. T. Dumais. Characterizing web content, user interests, and search behavior by reading level and topic. In *WSDM '12*, pages 213–222. ACM, 2012.

[24] Y. Li, Z. Zheng, and H. K. Dai. Kdd cup-2005 report: facing a great challenge. *SIGKDD Explor.*, 7(2):91–99, 2005.

[25] Z. Liao, D. Jiang, E. Chen, J. Pei, H. Cao, and H. Li. Mining concept sequences from large-scale search logs for context-aware query suggestion. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 3(1):17, 2011.

[26] J. Luo, S. Zhang, X. Dong, and H. Yang. Designing states, actions, and rewards for using pomdp in session search. In *ECIR '15*, pages 526–537, 2015.

[27] Q. Mei, D. Zhou, and K. Church. Query suggestion using hitting time. In *CIKM '08*, pages 469–478. ACM, 2008.

[28] G. A. Miller. Wordnet: A lexical database for english. *CACM*, 38(11):39–41, 1995.

[29] B. Mitra. Exploring session context using distributed representations of queries and reformulations. In *SIGIR '15*, pages 3–12. ACM, 2015.

[30] B. Mitra, M. Shokouhi, F. Radlinski, and K. Hofmann. On user interactions with query auto-completion. In *SIGIR '14*, pages 1055–1058. ACM, 2014.

[31] U. Ozertem, O. Chapelle, P. Donmez, and E. Velipasaoglu. Learning to suggest: a machine learning framework for ranking query suggestions. In *SIGIR '12*, pages 25–34. ACM, 2012.

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

[33] R. L. Santos, C. Macdonald, and I. Ounis. Learning to rank query suggestions for adhoc and diversity search. *Information Retrieval*, pages 1–23, 2013.

[34] D. Shen, J.-T. Sun, Q. Yang, and Z. Chen. Building bridges for web query classification. In *SIGIR '06*, pages 131–138. ACM, 2006.

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

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

[37] Y. Song, D. Zhou, and L. He. Query suggestion by constructing term-transition graphs. In *WSDM '12*, pages 353–362. ACM, 2012.

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

[39] H. Yang, D. Guan, and S. Zhang. The query change model: Modeling session search as a markov decision process. *ACM TOIS*, 33:20, 2015.