Learning User Reformulation Behavior for Query Auto-Completion

Jyun-Yu Jiang, Yen-Yu Ke, Pao-Yu Chien and Pu-Jen Cheng
Department of Computer Science and Information Engineering
National Taiwan University, Taipei 106, Taiwan
{jyunyu.jiang, odie1111, b97901186}@gmail.com, pjcheng@csie.ntu.edu.tw

ABSTRACT

It is crucial for query auto-completion to accurately predict what a user is typing. Given a query prefix and its context (e.g., previous queries), conventional context-aware approaches often produce *relevant* queries to the context. The purpose of this paper is to investigate the feasibility of exploiting the context to learn user reformulation behavior for boosting prediction performance. We first conduct an in-depth analysis of how the users reformulate their queries. Based on the analysis, we propose a supervised approach to query auto-completion, where three kinds of reformulation-related features are considered, including term-level, query-level and session-level features. These features carefully capture how the users change preceding queries along the query sessions. Extensive experiments have been conducted on the large-scale query log of a commercial search engine. The experimental results demonstrate a significant improvement over 4 competitive baselines.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval.

General Terms

Algorithms, Experimentation, Performance.

Keywords

Query auto-completion; Query reformulation.

1. INTRODUCTION

Most of the modern search engines provide the query auto-completion service to help users formulate their queries while they are typing in search boxes. Such service is especially helpful for mobile phone users because entering keywords on small touch screens is time-consuming. Prevalent methods on query auto-completion[1] utilize search logs to generate suggested completions [2, 29, 30]. For an input prefix, the methods extract those queries with matched leading characters from the logs as *candidates*. Since the given prefix could be short and ambiguous, the number of the candidates might be extremely large. Existing methods mostly rank the candidates by their popularity in the logs. Consequently, less popular queries are harder to predict.

A possible solution to the ranking problem is to exploit query context such as previous queries and click-through data in the same session[2]. The idea is reasonable since disambiguating the user's search intent by context has been studied extensively in query suggestion [5,9,16,25,26]. Query suggestion is closely related to query completion. The major difference is that it has got no partial input, i.e., the prefix. In general, if the outputs coming from query suggestion match the prefix, they can also be regarded as the suggested completions. For a given prefix and its context, conventional context-aware methods focus mainly on ranking the candidates according to the similarity or the dependency between the candidate and the given context. For example, some works rank the candidates based on their similarity to the recent queries [2] or their occurrence with previous queries [17]. Some works [16] model query dependencies along the sessions, by which the candidates whose contexts are more similar to the given context will be ranked higher. The work [25] further clusters relevant queries with common click-through data into concepts. Query dependency is then transformed to concept dependency. Despite such methods have shown their effectiveness in producing *relevant* queries to the given context, query completion is required to accurately predict what the user is typing.

Take a query session obtained from a commercial search engine log as an example: *stomach sounds* → *irritable bowel syndrome* → *cramps stomach*. Suppose the last query is the intended query. The prefix is *c*. Its preceding two queries serve as the context. Many relevant queries are generated from conventional context-aware methods such as *colon cancer symptoms* (from [2]), *celiac disease* (from [16]), and *colon cancer* (from [25]) in our experiments. Although the suggested queries are conceptually relevant to the context, the problem becomes more challenging when the goal is to predict the actual query *cramps stomach*. Some works take into account other factors for pursuing better approximation in

SIGIR'14, July 6–11, 2014, Gold Coast, Queensland, Australia.

[1] *Query auto-completion* is hereinafter referred to as *query completion* for simplicity.
[2] A query session is a sequence of queries with the same information need.

prediction such as temporal dynamics [30, 32] and demographics [29,33]. Different from ours, they do not utilize the context information. One possible way to boost the prediction performance by context is carefully investigating how the users refine their queries to get more satisfactory results. If the system gets to know the how, it would be capable of predicting the potential query the user might submit.

In addition to query dependency, context information also demonstrates how the users reformulate their queries repeatedly throughout the search process. The way people reformulate their queries, or user reformulation behavior, has been studied as *query reformulation strategy* in previous work [6,18]. There are two types of query reformulation strategies: semantic and syntactic relations. Semantic relations such as *generalization* (e.g., *lion* $\rightarrow$ *animal*) and *specification* (e.g., *computer* $\rightarrow$ *mac*) originate from linguistics. Syntactic relations refer to the statistical information about consecutive queries along the sessions such as *term-adding*, *term-reuse* and *the change of query lengths*. As the way to do reformulation expresses how one query is changed to another, to some extent it might be helpful in prediction for query completion. In the previous example, the user reuses *stomach* (appearing in the preceding query) to form the last query.

Understanding user search behavior benefits various applications, including query suggestion [10], query classification [8] and Web search-result ranking [37]. The work [31] mines user reformulation activities for query suggestion but it does not consider the context information, which has been shown effective in understanding the user's information need. Though lots of work has studied query reformulation strategies, none of them model user reformulation behavior for query completion. As query reformulation strategies reflect how the users refine their queries and query completion requires accurate prediction of what a user is typing, these motivate us to investigate the feasibility of exploiting the context to learn user reformulation behavior for boosting prediction performance of query completion.

In this paper, we focus primarily on learning user reformulation behavior for query completion. We first conduct an in-depth analysis of how the users reformulate their queries. It is expected that if we know how one query is changed to another, we are able to predict the user's next intended query more accurately. Given a prefix and its context together with a set of candidate queries that start with the prefix[3], the goal of this paper is learning to rank the candidate queries so that the top-ranked queries are more likely to be the query the user is typing. In other words, we want to predict the intended query that has been partially entered by the user. The context includes previous queries and their chick-through data in the same session. As syntactic relations can be easily identified by observing the queries themselves, our ranking model comprehensively considers various factors that are essential in query prediction, including term-level, query-level and session-level syntactic features. There are totally 43 reformulation-related features. Extensive experiments have been conducted on the large-scale query log collected from a commercial search engine. The experimental results reveal that the prediction performance can be significantly improved based on our model, compared to 4 competitive baselines. Such improvement is consistent across different datasets with different session lengths.

In the rest of this paper, we make a brief review on related work in Section 2. Based on a statistical analysis on user reformulation behavior in Section 3, we describe our problem and approach to query completion in Section 4. The experimental results are presented in Section 5. Finally, in Section 6, we give our discussions and conclusions.

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

2. RELATED WORK

Query Auto-completion. The query completion task can be divided into two steps: *filtering* and *ranking*. The former is to generate candidate queries or phrases that start from the given prefix. The latter is to sort the candidates so that the top-ranked ones are more likely what the user intends to input. Most works on query completion extract candidates from query logs. On the contrary, Bhatia *et al.* [4] utilized a document corpus to extract a set of candidate phrases and proposed a probabilistic model to select those highly correlated to the prefix as the suggested completions. The first step (filtering) needs to address two issues, including speed and error-tolerance. Chaudhuri *et al.* [11] captured input typing errors by calculating edit distance and then proposed algorithms for auto-completion based on n-gram and trie traversal techniques. Bast *et al.* [3] designed an indexing structure for speeding up the query process and saving space, compared to other compressed inverted indices. Once the candidates are generated, they can be ranked by manifold approaches for various applications such as Web search [2] and product search [11].

The second step emphasizes how to accurately select what the user is typing from the pool of the candidates. An intuitive way is to rank the candidates by their popularity in the logs; however, less popular queries will become harder to predict. White *et al.* [35] proposed a real-time query expansion model to produce new expansion terms and update following terms to reflect potential completions. To make a better approximation in prediction, Bar-Yossef *et al.* [2] regarded the user's recent queries as the context. Due to the sparseness problem of the query logs, they applied the query expansion technique to augment the candidate and the context. The candidates were then ranked according to their similarity to the context. Some works utilize temporal dynamics to estimate query frequency more precisely. Shokouhi *et al.* [30] presented a time-sensitive approach to rank the candidates according to their expected popularity or frequencies. Strizhevskaya *et al.* [32] also predicted the frequency of the candidate by modeling their frequency trend in the past. Shokouhi *et al.* [29] took personalization into account. They learned personalized rankers based on user age, gender, location, and search history.

Our solution differs from these works in that we try to model user reformulation behavior based on the given context. Although the work [31] also mines user reformulation activities for query suggestion, it does not consider context information.

Query Suggestion. Query suggestion, which draws much attention in IR, is closely related to query prediction, but the goal is much more different. The query suggestion methods focus on finding relevant queries to preceding queries, instead of finding the query that will be actually submitted by the user. Since previous work on context-aware query

suggestion might suggest the intended query that the user is typing, it could be regarded as our related work.

With the query session and click-through data as the context, there are many ways to generate suggested queries such as association rules [14], measuring the similarity among queries [15,36] and query clustering [10,25]. Huang *et al.* [17] and Fonseca *et al.* [13] sought for the queries having high frequency of co-occurrence with the current query[4] in the sessions. Jones *et al.* [22] extracted the queries often adjacent to the current query. Boldi *et al.* [5] built a *query-flow graph* in which edges connected the query pairs submitted together. Based on the click-through data, those queries that were analogous to the user's preceding queries were treated as the suggested queries. Cao *et al.* [10] and Liao *et al.* [25] made use of click-through data to build a bipartite graph and clustered queries according to the graph. Mei *et al.* [26] applied the idea of random walk to the click-through bipartite graph for finding relevant queries. Considering the whole query sessions as the contexts, He *et al.* [16] and Cao *et al.* [9] adopted the variable order Markov model and the hidden Markov model. They modeled the transition probabilities between queries.

Some context-aware query suggestion methods further cope with the problems of data sparseness and new queries when using search logs. For sparse data, He *et al.* [16] allowed the context information to be partially matched, making the suggestion method more flexible and robust. Liao *et al.* [25] grouped queries into various concepts. Ozertem *et al.* [27] and Santos *et al.* [28] built a machine learning framework and applied ranking models to rate queries which were projected into a feature space. The features considered included query co-occurrences and some simple query reformulation strategies [27]. We also handle the two problems. Our solution differs from these works in that we focus on query completion and abundant reformulation-based features, which are potentially beneficial to query prediction.

Query Reformulation. Query reformulation is the process of refining preceding queries to get better results. Most of previous work focused mainly on determining the types of the reformulation strategies [18], learning to predict their types [12], and understanding how the users reformulate their queries [18]. They analyzed query reformulation strategies in two major aspects, including syntactic (format) and semantic (content) analyses. The semantic analysis categorizes reformulation types into *generalization* and *specialization* [1]. The syntactic analysis focuses on exploring the syntactic change between two queries such as adding words, removing words, and acronym expansion. Boldi *et al.* [6] defined four types of query transition. Jansen *et al.* [19] and Huang *et al.* [18] classified query reformulation into 6 and 15 types, respectively. Some works use query reformulation strategies to predict query-term performance. Jones *et al.* [21] and Lee *et al.* [23] determined the effectiveness of a query term so that the retrieval performance could be improved by filtering out ineffective terms or including effective terms. The reformulation behavior and context information are also usually used to personalize the search results. White *et al.* [34] predicted users' interests with search context. Jiang *et al.* [20] captured user's preference through mining click-through data and query reformulation. In our work, we enumerate copious reformulation behavior as robust features and adopt them for query completion.

[4]The current query is the last query a user submitted.

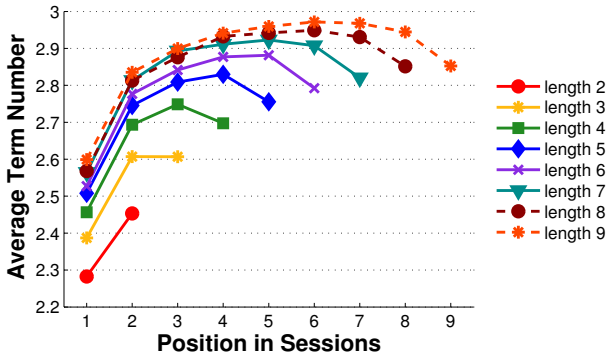

Figure 1: Average number of query terms in each position of the sessions from length 2 to 9.

3. ANALYSIS OF USER REFORMULATION BEHAVIOR

In this section, we analyze how users reformulate their queries, by which we further explain why modeling such reformulation behavior can benefit query completion. This analysis is conducted on the one-week search engine log of a commercial search engine (from 1 May, 2013 to 7 May, 2013). The queries in the log are first segmented into sessions with a 30-minute threshold. Two consecutive queries belong to the same session if they are issued within the threshold. After removing rare queries and single-query sessions, there are finally 63,661,691 sessions and 5,197,821 unique queries in total. Note that the setting is consistent with the experiments presented in Section 5.

3.1 Number of Terms in Queries

Since a query session consists of a sequence of queries $\langle q_1, q_2, \cdots, q_T \rangle$ submitted by a user, each query submission q_i corresponds to a position i in the session. That is, q_1 is the first query submitted or the first position in the session. For the sessions with the same length[5], we calculate the average number of query terms in each position. Figure 1 presents its changes along the sessions from length 2 to 9. It can be found (1) queries in longer sessions tend to contain more terms. One possible explanation is that queries with more terms may carry more complex search intent; therefore users need to reformulate their queries more times; (2) the average number of query terms increases along the session. It's visible from the first position to the second one. The curve drops near the end of the session.

We further correlate this change with the semantic relations, including *specialization* and *generalization*. Intuitively, *specialization* aims to narrow down the search results, thus extra terms, i.e., more constraints, are expected to be added into the original query. On the contrary, *generalization* may lead to a decrease of term number to relax restrictions. We randomly sample 1136 sessions from the dataset, partition them into 2283 consecutive query pairs,

[5]Session length is the number of queries submitted in the session.

Table 1: Percentage of specialization and generalization in our labels

Relation	% in Log	Average Position	Median Position	Change of Term Number	% in Relation	Example
Specialization	27.7%	2.9951	2	Increase	**84.2%**	camera → digital camera
				Decrease	3.7%	perennial plants → stonecrop
				Equal	12.1%	guest book for party → anniversary party guest book
Generalization	12.2%	3.3122	3	Increase	4.0%	airport parking newark → airport parking new york
				Decrease	**82.5%**	great lakes auto → great lakes
				Equal	13.5%	honda blue book → car blue book

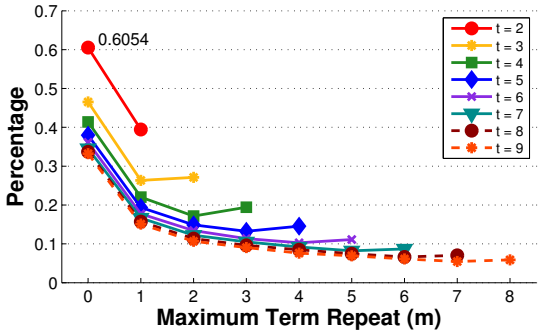

Figure 2: The percentage of sessions whose lengths are longer or equal to t over different m's (maximum term repeat).

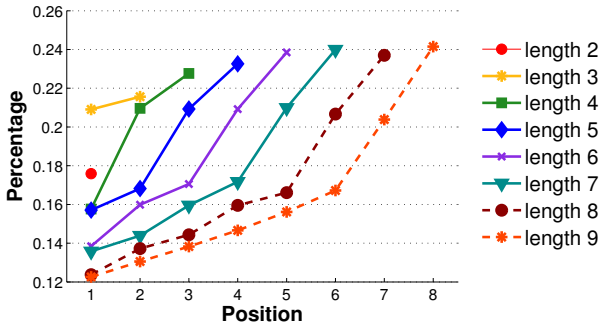

Figure 3: The percentage of queries containing some identical term to the last query over different positions.

and manually label them with *specialization*, *generalization*, or other. Table 1 gives the statistics of our labels, which show that *specialization* (27.7%) is about twice as many as *generalization* (12.2%). It explains that most of time queries are reformulated to be longer ones, as shown in Fig. 1. The number of terms increases for most of *specialization* (84.2%), while the number decreases for most of *generalization* (82.5%). The average and median of positions for *specification* are both smaller than those for *generalization*. That is why the average term number increases along the sessions but drops near the end, as shown in Fig. 1. The observation implies the combination of query length and position number as features are important. To some extent, they might be helpful in detecting the semantic relations of *specification* and *generalization*.

3.2 Term Repeat

This section discusses if people use some terms more frequently in previous queries, they will be also more likely to use that term again.

For a query q_i whose $i > 1$, i.e., the query not submitted in the beginning, we examine each term in this query and count how many times it has been used to form previous queries $\langle q_1, \cdots, q_{i-1} \rangle$ in the same session. Then we take the maximum value m as "maximum term repeat" for query q_i if there is at least one term in q_i used m times in previous queries. Given a threshold t, for any session whose length is longer or equal to t, we extract the first t queries as a sequence and calculate the "maximum term repeat" value for the *t-th* query. We calculate the frequency of "maximum term repeat" from 0 to $t-1$ and divide them by the number of extracted sequences.

From Fig.2, we can observe that most reformulated queries do not share any identical term with their previous queries. Take $t = 2$ as an instance. After submitting the first queries, 60.54% of the users do not repeat any term. Furthermore, if we compare different lengths (i.e., t) of the sessions whose "maximum term repeat" is zero (i.e., $m = 0$), it can be found that when the length of a session increases, the percentage of repeating previously-used terms also increases. It means that a reformulated query is more likely to contain the terms used before when it appears in the latter steps of a session. We also notice that the percentage decreases as "maximum term repeat" increases. However, the proportion of reformulated queries containing terms used in all previous $t-1$ queries increases, compared to that of $t-2$ as shown in Fig. 2. For example, for the curve of $t = 5$, its percentage value at $m = 4$ $(t-1)$ is larger than that at $m = 3$ $(t-2)$. It indicates that some terms are repeatedly used in all queries throughout the session.

At last, for a reformulated query which contains a used term, we would like to know "*in which positions*" that term would appear. We again extract the first $t-1$ queries from the sessions whose lengths are equal to t. Then we count the number of the queries in each position that contain some identical terms to the t-th queries. Figure 3 shows the percentage of queries containing identical terms in each position within sequences from length 2 to 9. It shows that users tend to repeat terms used in the nearest query, and may not stick to the terms in the first query for query reformulation.

3.3 Term Repeat and Clicks

This section discusses if some search results of a query have been clicked, a user may think the query is "effective"

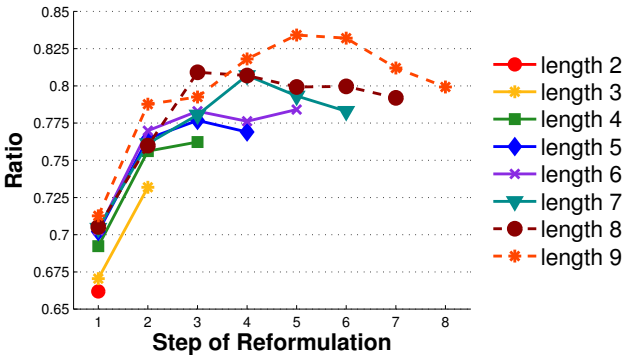

Figure 4: The ratio of two conditional probabilities that if the previous query causes clicks or not, then the latter query would contain some identical terms, over different steps of reformulation.

and reuse some terms of it later. We first segment sessions into consecutive query pairs, and then calculate the probability of a pair containing some identical terms, given that the previous query causes clicks or not, respectively. That is, for a consecutive query pair (q_i, q_{i+1}), we calculate

$$P\left(S\left(q_i\right) \cap S\left(q_{i+1}\right) \neq \varnothing \mid c_i \neq 0\right)$$

and

$$P\left(S\left(q_i\right) \cap S\left(q_{i+1}\right) \neq \varnothing \mid c_i = 0\right),$$

here $S\left(q\right)$ and c_i represent the term set of query q and the number of clicks in position i, respectively.

We find that if the previous query causes clicks, the probability of the latter query containing some identical terms with the previous one is 36.06%. If the previous query does not lead to any clicks, the probability of the latter query containing the same terms is 50.54%. The result shows that if a query does not cause any clicks, its term would be reused probably later. It can be explained: if a query causes clicks, then the user's search intent may somewhat be satisfied, and some other terms will be considered in the next query; on the contrary, if a query does not cause any clicks, the user may stick to some terms while replacing some with others.

Then we calculate the ratio of these two values

$$\frac{P\left(S\left(q_i\right) \cap S\left(q_{i+1}\right) \neq \varnothing \mid c_i \neq 0\right)}{P\left(S\left(q_i\right) \cap S\left(q_{i+1}\right) \neq \varnothing \mid c_i = 0\right)}$$

for each position transition to examine its change throughout the session. From Fig. 4, we find that ratio is relatively low at the first step of reformulation, i.e., repeating terms at the first step of reformulation is more dependent to the clicked result of the first query than that at other steps of reformulation.

Next, we calculate the probability of previous query causing clicks, given that a query pair contains common terms or not, respectively. We find that if the query pairs contain some identical terms, the probability that the previous query causes clicks is 43.4%. If the query pairs do not contain any term in common, the probability that the previous query causes clicks is 57.7%. We also calculate the ratio of the two probabilities for each position transition throughout the whole sessions. The results indicate that the dependence between repeating terms and clicks on the previous query decreases when the number of reformulations increases.

3.4 Discussions

Previous statistical analyses show that there are several hidden reformulation patterns users might potentially follow when interacting with a search engine. Such findings provide useful clues for query completion methods to more precisely predict what the user is typing. For example, Fig. 1 shows that the number of query terms varies according to how many reformulations the user has performed. Thus, query length may change for different position numbers. In general, a longer query would be recommended in the beginning of the session. Figures 2 and 3 explain how the users choose the terms for a new query. Their decisions to term-adding, term-keeping and term-removing are often related to recent queries, instead of the original query, i.e., the first query. To some extent, such decisions also determine the similarity between consecutive queries. Figure 4 further tells us that whether a query causes clicks or not affects if its terms will be reused later. In general, the terms appearing in the recent queries that do not cause any clicks would have a better chance to be included in the new query. Overall, these reformulation patterns carefully capture how the terms are used and how the term-usage is related to session length, query length, number of reformulations and click-through data, which show their good potential for boosting the prediction performance in query completion.

4. QUERY AUTO-COMPLETION WITH REFORMULATION

Formally, a search session is defined as a sequence of queries $\langle q_1, q_2, \cdots, q_T\rangle$ issued by a single user within a time interval. A query is composed of a set of terms $S(q_i)$. Each query q_i has corresponding timestamp t_i (the time issued) and click information c_i (the number of clicks for q_i). Suppose a user intends to enter q_T but only inputs x in the search box, where x is a prefix of q_T. $\langle q_1, q_2, \cdots, q_{T-1}\rangle$ is regarded as the query context. We want to predict q_T for query completion. Assume there is a set of candidates (or candidate queries) $Q_T = \{q'_j\}$ that are probably issued after q_{T-1}. Such candidate set can be collected in different ways. For example, q'_j can be the query following q_{T-1} or the query following q_i $(i \in \{1, \cdots, T-1\})$ in previous search sessions (i.e., the query logs). Note that x should be a prefix of q'_j. Given a search session with $T-1$ queries and its candidate set of queries $Q_T = \{q'_j\}$, our goal is to give a ranking to each query $q'_j \in Q_T$ so that the suggested queries with higher rankings may be more likely to be q_T. In other words, we would like to estimate a probabilistic-like score positively correlated to the probability $P(q_T = q'_j \mid \langle q_1, q_2, \cdots, q_{T-1}\rangle)$.

We propose a supervised approach here. Three kinds of reformulation-related features that possibly affect the prediction accuracy are taken into account. These features capture how the user changes preceding queries along the session. Considering different levels of query reformulation, we divide them into three categories: term-level features, query-level features, and session-level features, as shown in Table 2. There are 43 features in total. With the training data, collected from the real search engine log, almost any learning-to-rank algorithm can be applied to obtain our ranking model. We choose *LambdaMART* [7] because *Lamb-*

Table 2: Defined reformulation behavior and the formulas for calculating reformulation features.

Category	Feature Class	Description	Formulas
Term	Term Combination (16 features)	number of terms	$\lvert \cup_{i=1}^{T} S(q_i) \rvert$, $\lvert S(q_{T-1}) \cup S(q_T) \rvert$
		term keeping	$\lvert \cap_{i=1}^{T} S(q_i) \rvert$, $\lvert S(q_{T-1}) \cap S(q_T) \rvert$, $\text{sgn}(\lvert S(q_{T-1}) \cap S(q_T) \rvert)$
		term adding	$\lvert S(q_T) - S(q_{T-1}) \rvert$, $\text{sgn}(\lvert S(q_T) - S(q_{T-1}) \rvert)$
		term removing	$\lvert S(q_{T-1}) - S(q_T) \rvert$, $\text{sgn}(\lvert S(q_{T-1}) - S(q_T) \rvert)$
		number of used terms	$\lvert S_{\text{used}}(q_T) \rvert$, $\lvert S(q_T) - S_{\text{used}}(q_T) \rvert$
		ratio of used terms	$\lvert S_{\text{used}}(q_T) \rvert / \lvert S(q_T) \rvert$, $1 - \lvert S_{\text{used}}(q_T) \rvert / \lvert S(q_T) \rvert$
		number of repeat times	$\text{Rep}(q_T)$, $\text{Rep}(q_T)/T$, $\text{Rep}(q_T)/\lvert S(q_T) \rvert$
Query	Query Similarity (10 features)	cosine similarity	$\text{sim}_{\cos}(q_{T-1}, q_T)$
		average cosine similarity	$\frac{1}{T-1}\sum_{i=1}^{T-1} \text{sim}_{\cos}(q_i, q_{i+1}))$, $\frac{1}{T-1}\sum_{i=1}^{T-1} \text{sim}_{\cos}(q_i, q_T))$
		trends of cosine similarity	$\text{sim}_{\cos}(q_{T-1}, q_T) / \frac{1}{T-2}\sum_{i=1}^{T-2} \text{sim}_{\cos}(q_i, q_{i+1}))$
			$\text{sim}_{\cos}(q_{T-1}, q_T) / \frac{1}{T-2}\sum_{i=1}^{T-2} \text{sim}_{\cos}(q_i, q_T))$
		Lev. similarity	$\text{sim}_{\text{Lev}}(q_{T-1}, q_T)$
		average Lev. similarity	$\frac{1}{T-1}\sum_{i=1}^{T-1} \text{sim}_{\text{Lev}}(q_i, q_{i+1}))$, $\frac{1}{T-1}\sum_{i=1}^{T-1} \text{sim}_{\text{Lev}}(q_i, q_T))$
		trends of Lev. similarity	$\text{sim}_{\text{Lev}}(q_{T-1}, q_T) / \frac{1}{T-2}\sum_{i=1}^{T-2} \text{sim}_{\text{Lev}}(q_i, q_{i+1}))$
			$\text{sim}_{\text{Lev}}(q_{T-1}, q_T) / \frac{1}{T-2}\sum_{i=1}^{T-2} \text{sim}_{\text{Lev}}(q_i, q_T))$
	Query Length (6 features)	number of terms	$\lvert S(q_T) \rvert$
		average number of terms	$\frac{1}{T-1}\sum_{i=1}^{T-1} \lvert S(q_i) \rvert$, $\frac{1}{T}\sum_{i=1}^{T} \lvert S(q_i) \rvert$, $\lvert S(q_{T-1}) \rvert + \lvert S(q_T) \rvert$
		trends of term number	$\lvert S(q_T) \rvert / \frac{1}{T-1}\sum_{i=1}^{T-1} \lvert S(q_i) \rvert$, $\lvert S(q_{T-1}) \rvert - \lvert S(q_T) \rvert$
	Query Frequency (2 features)	pairwise frequency	$P((q_{T-1}, q_T)\lvert q_T)$, $P((q_{T-1}, q_T)\lvert q_{T-1})$
Session	Click-through Data (6 features)	previous clicks	c_{T-1} , $\text{sgn}(c_{T-1})$
		number of effective terms	$\lvert C_{\text{eff}}(q_T) \rvert$
		ratio of effective terms	$\lvert C_{\text{eff}}(q_T) \rvert / T$, $\lvert C_{\text{eff}}(q_T) \rvert / \lvert S(q_T) \rvert$, $\lvert C_{\text{eff}}(q_T) \rvert / \lvert S_{\text{used}}(q_T) \rvert$
	Time Duration (2 features)	average time duration	$\frac{1}{T-1}\sum_{i=1}^{T-1}(t_{i+1} - t_i)$
		trends of time duration	$(t_T - t_{T-1}) / \frac{1}{T-2}\sum_{i=1}^{T-2}(t_{i+1} - t_i)$
	Position Number (1 feature)	position in the session	(T)

daMART is a boosted version of the *LambdaRank* algorithm improving overall ranking performance [7].

4.1 Term-Level Reformulation Features

A query q_i in a session consists of a set of terms $S(q_i)$. Users might add or remove terms between two consecutive queries. The term-level features try to measure the effectiveness of a term used in a query. The effectiveness of a term can be explained in several ways such as the probability to add the term, delete the term or re-use the term, and the probability to cause clicks on the search results.

***Term Combination*:** For a query q_i and its preceding query q_{i-1} in a session, there are four possible term-usage ways: (1) adding terms only; (2) removing terms only; (3) adding terms and removing terms concurrently while some terms are kept; and (4) adding terms and removing terms concurrently while no term is kept. We encode types of term-usage as categorical features (i.e., four binary features). In Table 2, $\text{sgn}(x)$ is a function calculating the binary features. If $x > 0$, then $\text{sgn}(x) = 1$. If $x = 0$, then $\text{sgn}(x) = 0$.

Set operations among term sets $S(q_i)$ may measure some statistical information. The union of terms learns how many different terms are used throughout the session, which may imply how much information conveyed by user information need. The intersection of terms learns how many terms kept by the user, reflecting how much unchanged information during the session. Besides, it is expected that a term is likely to be used again if it has been used recently in the session. We capture such feature by measuring how often the user reuses previous terms when reformulating a new one. In Table 2, we remove those terms not appearing in previous $T-1$ queries from $S(q_T)$ to form $S_{used}(q_T)$. For every term in q_T, function $Rep(q_T)$ counts its times of occurrence in previous $t-1$ queries. These counts are summed up and then normalized by the length of the session or the number of terms in q_T.

Note that from the analyses in Section 3.2, "maximum term repeat" is highly related to the term-level features. Larger numbers of "maximum term repeat" could increase the size of intersection set. Smaller numbers of "maximum term repeat" could increase the size of union set.

4.2 Query-Level Reformulation Features

These features are about how users reformulate queries without the consideration of what terms are used.

***Query Similarity*:** As each reformulated query may be syntactically similar to its preceding query under the same information need, the similarity between queries could be helpful in determining whether a query is a suitable reformulation of another. In Table 2, term-based cosine similarity (denoted as $Sim_{cos}(q_i, q_j)$) and Levenshtein distance (denoted as $Sim_{Lev}(q_i, q_j)$) [24] are used as similarity metrics, which measure similarity based on term- and character-level representations, respectively. First we compute the Leven-

shtein distance between q_T and its preceding query q_{T-1}, and then we compute the average similarity of the query pairs in the whole session in two ways: (1) the average similarity between q_T and each of the previous $T-1$ queries; (2) the average similarity of consecutive $T-1$ query pairs. At last we calculate the ratio of the similarity between the reformulated query and its preceding one to the average similarity mentioned earlier.

***Query Length*:** Under the same search intent, we assume that the number of terms used in a query throughout a session does not change rapidly. We compute average number of terms in a query for a session. The ratio of term number in q_T to that in previous $T-1$ queries is calculated in order to show the degree of change in the number of terms, that is, the trend of term numbers. The larger ratio value means that the last query is much longer than average length of previous queries.

***Query Frequency*:** Search logs can provide useful information such as the co-occurrence of queries that benefits query suggestion. Here we take the relative frequency of (q_{T-1}, q_T) to all the queries immediately following q_{T-1} and preceding q_T, respectively, as features.

4.3 Session-Level Reformulation Features

The session-level features capture how users perform reformulations along the sessions without considering what queries or terms they exactly submit.

***Position Number*:** Since a session is a sequence of queries, the position of a query in the session determines how many times a user has reformulated the queries. Our analyses given in Section 3.1 show that users' reformulation strategies may change over different positions. The number of position is, therefore, adopted as a feature in our work.

***Click-through Data*:** From Fig. 4, we know that click information in a session is related to term-usage. If a new term is added to a query and the new query does cause user clicks on the search results, then the term is more likely to be effective. With the click-through data, we can calculate how many search results of a query are clicked by the user. In Table 2, c_i is the number of clicks in position i. For each term in q_T, if it has been used in some of the previous queries, we count the number of clicks on the search results of that query and then sum up these counts by $C_{eff}(q_T)$. It is further normalized by the session length T, the number of terms in q_T, and the number of terms in q_T that have appeared previously $S_{used}(q_T)$, respectively.

***Time Duration*:** The duration of time users stay on the search results for a query might affect how they reformulate next query. The features include the average time difference of $T-1$ pairs of consecutive queries, and the ratio of the time difference between q_{T-1} and q_T to the average time difference of the previous $T-2$ pairs.

5. EXPERIMENTS

In this section, we conduct extensive experiments on a real, large-scale dataset to verify the performance of our ranking model in predicting the user's intended query.

5.1 Datasets and Experimental Settings

Our experimental data comprises all of the queries submitted to a commercial search engine from 1 May, 2013 to 7 May, 2013. The query log is first segmented into sessions with a 30-minute threshold as the session boundary. We drop those queries that are misspelled or appear less than 10 times in the whole log. We then discard the sessions that contain only one query since context-aware methods need at least one preceding query as the context. After removing rare queries and single-query sessions, there are 63,661,591 sessions and 5,197,821 unique queries in total. For each session with T queries, we treat the last query q_T as the ground truth, that is, the intended query we want to predict. The preceding queries $\langle q_1, q_2, \cdots q_{T-1} \rangle$ and their click-through information serve as the context.

The cleaned data is further partitioned into two sets. The first 4-day data is used for training. The remaining 3-day data is for testing. Finally, we collect 35,926,476 sessions submitted before 5 May, 2013 as our training set. We use query frequency to generate candidate set. After filtering out those not starting from the prefix, the top-10 queries ranked by frequency form our candidate queries. For example, if the prefix is a, the candidates are the top 10 high-frequency queries starting with a in the training set. To get the training set, we remove the sessions whose ground truths (or answers) are not included in the corresponding candidate sets; that is, we guarantee the candidate set contains the answer. After filtering, there are 4,900,363 sessions in the testing set. 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 [29], the way to generate candidates [29], how to determine the prefix [2], and the dropping of the cases with no answers in the candidate set [29].

To evaluate performance on sessions with different lengths, the testing sessions are divided into three datasets, including "Short Sessions" (2 queries), "Medium Sessions" (3 to 4 queries) and "Long Sessions" (5 or more queries). Besides, we tune our LambdaMART model with parameters of 1,000 decision trees across all experiments.

5.2 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 [2] 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 its similarity to recent queries.

- Query-based VMM (or QVMM). QVMM proposed in [16] 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 a 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.

Table 3: The performance of 5 methods in whole testing set and three subsets. All improvements of our method against the MPC method are significant differences at 95% level in a paired t-test.

Dataset	Measure	MPC	Hyb.C [2]	QVMM [16]	CACB [25]	Our Approach
Whole Testing Set	MRR	0.6415	0.6604 (+2.95%)	0.7137 (+11.25%)	0.7112 (+10.86%)	**0.7433 (+15.87%)**
	SR@1	0.4756	0.5017 (+5.50%)	0.5658 (+18.97%)	0.5593 (+17.61%)	**0.6095 (+28.16%)**
	SR@2	0.6410	0.6625 (+3.36%)	0.7349 (+14.66%)	0.7363 (+14.88%)	**0.7672 (+19.70%)**
	SR@3	0.7623	0.7729 (+1.39%)	0.8293 (+8.79%)	0.8305 (+ 8.94%)	**0.8474 (+11.16%)**
Short Sessions (2 Queries)	MRR	0.6338	0.6335 (−0.04%)	0.7125 (+12.43%)	0.7074 (+11.62%)	**0.7224 (+13.98%)**
	SR@1	0.4654	0.4633 (−0.45%)	0.5636 (+21.10%)	0.5519 (+18.59%)	**0.5794 (+24.49%)**
	SR@2	0.6283	0.6310 (+0.43%)	0.7329 (+16.64%)	0.7348 (+16.95%)	**0.7450 (+18.58%)**
	SR@3	0.7575	0.7567 (−0.10%)	0.8291 (+9.46%)	0.8298 (+9.54%)	**0.8320 (+9.84%)**
Medium Sessions (3 to 4 Queries)	MRR	0.6513	0.6906 (+6.04%)	0.7161 (+9.95%)	0.7160 (+9.93%)	**0.7654 (+17.50%)**
	SR@1	0.4889	0.5443 (+11.33%)	0.5707 (+16.74%)	0.5695 (+16.49%)	**0.6420 (+31.32%)**
	SR@2	0.6552	0.6991 (+6.70%)	0.7369 (+12.47%)	0.7368 (+12.44%)	**0.7892 (+20.45%)**
	SR@3	0.7692	0.7928 (+3.06%)	0.8294 (+7.83%)	0.8305 (+7.98%)	**0.8626 (+12.15%)**
Long Sessions (5 or more Queries)	MRR	0.6522	0.7076 (+8.49%)	0.7130 (+9.32%)	0.7162 (+9.82%)	**0.7842 (+20.24%)**
	SR@1	0.4885	0.5707 (+16.83%)	0.5631 (+15.27%)	0.5676 (+16.20%)	**0.6656 (+36.27%)**
	SR@2	0.6632	0.7149 (+7.79%)	0.7394 (+11.49%)	0.7422 (+11.91%)	**0.8139 (+22.72%)**
	SR@3	0.7674	0.7974 (+3.91%)	0.8300 (+8.16%)	0.8335 (+8.61%)	**0.8798 (+14.65%)**

5.3 Evaluation Metrics

With the ground truth q_T, we can evaluate the quality of query prediction by two metrics, including *mean reciprocal rank* (MRR) and *success rate at top-k* (SR @ k). MRR is the multiplicative inverse of the rank of the actual query q_T in the ranking list. Given the testing set S, the MRR [2] for an algorithm A is defined as follows:

$$\text{MRR}(A) = \frac{1}{|S|} \sum_{(C,q_T)\in S} \frac{1}{hitrank(A, C, q_T)},$$

where C represents the context of a session (i.e., the preceding queries and click-through information); q_T is the ground truth. The function $hitrank(A, C, q_T)$ computes the position of the ground truth in the ordered list ranked by the algorithm A. The *success rate at top-k* (SR @ k) denotes the average percentage of the actual queries that can be found in the top-k queries over the testing data. Both of them are widely used for the task whose ground truth is only one instance such as query completion.

5.4 Overall Performance

Table 3 shows the experimental results in MRR and SR@k. Comparing the baseline methods, we find that all of previous methods perform better than frequency-based MPC in most cases. The difference in MRR and SR@k between the Hyb.C method and the MPC method is small for short sessions. The reason might be because the shorter sessions cannot provide sufficient context information so that the Hyb.C method fails to match the query to the context. On the other hand, while the length of a session increases, the Hyb.C method performs better accordingly. The QVMM method outperforms the Hyb.C method since QVMM models query transitions, rather than the similarity between two consecutive queries. It is not necessary for two semantically-relevant queries to have strong query dependency in search. Similarly, although CACB can deal with the problems of sparseness and new queries by grouping relevant queries into a cluster, its performance still approximates to that of the QVMM method. It is inferred that if a query is conceptually relevant to the context, it is not necessary to be what the user intends to type. Hence, the solutions to query suggestion cannot be fully applied to query completion.

Our approach achieves the best performance (compared with all of the 4 comparative baselines) and significantly outperforms MPC with progress by 9.84% to 36.27% of SR@k. It is worth noticing that our approach consistently performs better when the lengths of query sessions increase. More context is, therefore, preferable in query prediction. We have performed significant tests for the improvements of our approach against MPC using a paired t-test with a significant level of 95%.

5.5 Feature Analysis

To verify the effectiveness of the features, we inspect correlation between the features and $P(q_T \mid \langle q_1 \cdots q_{T-1} \rangle)$. The standard measurement, Kendall's tau, is adopted.

Figure 5 shows the absolute values of correlation for all of the 43 features. It can be found that all of the three types of features, namely term-level, query-level and session-level features, are quite useful. Obviously, the query-frequency feature is the most significant one. The feature measures the dependence between consecutive queries. It explains why the conventional query prediction (or suggestion) methods based [16] on query dependence generally have reasonable performance. Besides, the feature of query length also has strong connection to query prediction.

Many term-level features are helpful in determining how a user reformulates a query in term level such as the union of the terms in a session, if a term stays in the reformulated query (term-keeping), if a term has ever appeared or not in preceding queries (number of used terms and number of repeat times), and the cosine similarity or the Levenshtein

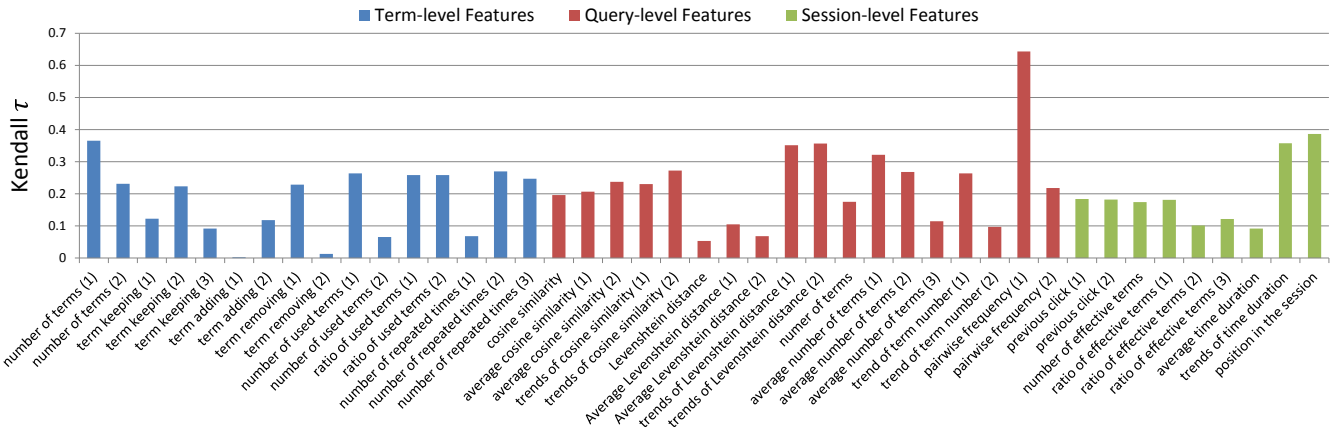

Figure 5: Absolute values of correlation between the features and relevance. The indices in the parentheses represent the order of features in Table 2. For example, *pairwise frequency (1)* represents the feature $P((q_{T-1}, q_T) \mid q_T)$, and *pairwise frequency (2)* is the feature of $P((q_{T-1}, q_T) \mid q_{T-1})$.

distance between two queries (especially between the latest two queries).

In the session-level features, the position of a query in a session is found to be highly related. This observation implies that how users reformulate their queries depends on how many queries they have submitted before. The click-through data like the number of clicks for a query has moderate correlation with reformulated queries. The information about time duration is quite important. This is because the duration of time a user stay on the search result is often related to other features such as the number of clicks, the session length, and the complexity of reformulating a query.

6. DISCUSSIONS AND CONCLUSIONS

In this paper, we propose a supervised approach for query completion aiming to predict the user's intended query with a partial input. Compared to conventional context-aware methods that directly model the term similarities or the query dependencies, our approach tries to learn how users change preceding queries, i.e., user reformulation behavior, along query sessions. The results of extensive experiments show that our methods can significantly improve the performance of the existing context-aware query completion and suggestion methods. Such improvement is consistent across different datasets with different session lengths. This is because (1) The reformulation-based model requires less training data. The number of reformulation features is only dozens or hundreds; (2) The reformulation-based model considers different user behavior for query reformulation. To capture such reformulation behavior, several kinds of features are taken into account such as session-related, query-related, and term-related features. The results of both the user behavior analysis in Section 3 and the feature analysis in Section 5.5 support that all of three-type features are useful and important. Based on the analyses and experimental results, we give an insight into the problems of how users reformulate their queries and why the reformulation-based features are helpful in query completion. Our future work includes the incorporation of semantic features into the model.

7. ACKNOWLEDGMENTS

We would like to thank Dr. Chin-Yew Lin for many valuable suggestions, and the anonymous reviewers for their helpful comments. This work was partially sponsored by the Ministry of Science and Technology, Taiwan, under Grant 102-2221-E-002-156.

8. REFERENCES

[1] J. Akahani, K. Hiramatsu, and K. Kogure. Coordinating heterogeneous information services based on approximate ontology translation. In *Proceedings of AAMAS-2002 Workshop on Agentcities: Challenges in Open Agent Systems*, pages 10–14. Citeseer, 2002.

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

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

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

[5] 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.

[6] P. Boldi, F. Bonchi, C. Castillo, and S. Vigna. From "dango" to "japanese cakes": Query reformulation models and patterns. In *WI '09*, pages 183–190, 2009.

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

[8] H. Cao, D. Hu, D. Shen, D. Jiang, J. Sun, E. Chen, and Q. Yang. Context-aware query classification. In *SIGIR '09*, pages 3–10. ACM, 2009.

[9] H. Cao, D. Jiang, J. Pei, E. Chen, and H. Li. Towards context-aware search by learning a very large variable

length hidden markov model from search logs. In *WWW '09*, pages 191–200, 2009.

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

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

[12] V. Dang and B. Croft. Query reformulation using anchor text. In *WSDM '10*, pages 41–50. ACM, 2010.

[13] 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.

[14] B. M. Fonseca, P. B. Golgher, E. S. de Moura, N. Ziviani, and N. Ziviani. Using association rules to discover search engines related queries. In *LA-WEB '03*, pages 66–71, 2003.

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

[16] 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.

[17] 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. *Journal of ASIST*, 54, 2003.

[18] J. Huang and E. N. Efthimiadis. Analyzing and evaluating query reformulation strategies in web search logs. In *CIKM '09*, pages 77–86, 2009.

[19] B. J. Jansen, D. L. Booth, and A. Spink. Patterns of query reformulation during web searching. *Journal of ASIST*, 60(7):1358–1371, 2009.

[20] D. Jiang, K. W.-T. Leung, and W. Ng. Context-aware search personalization with concept preference. In *CIKM '11*, pages 563–572. ACM, 2011.

[21] R. Jones and D. C. Fain. Query word deletion prediction. In *SIGIR '03*, pages 435–436. ACM, 2003.

[22] R. Jones, B. Rey, O. Madani, and W. Greiner. Generating query substitutions. In *WWW '06*, pages 387–396. ACM, 2006.

[23] C.-J. Lee, R.-C. Chen, S.-H. Kao, and P.-J. Cheng. A term dependency-based approach for query terms ranking. In *CIKM '09*, pages 1267–1276. ACM, 2009.

[24] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. Technical Report 8, 1966.

[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 Trans. Intell. Syst. Technol.*, pages 17:1–17:40, 2011.

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

[27] 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.

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

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

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

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

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

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

[34] R. W. White, P. N. Bennett, and S. T. Dumais. Predicting short-term interests using activity-based search context. In *CIKM '10*, pages 1009–1018. ACM, 2010.

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

[36] W. Wu, H. Li, and J. Xu. Learning query and document similarities from click-through bipartite graph with metadata. In *WSDM '13*, pages 687–696. ACM, 2013.

[37] B. Xiang, D. Jiang, J. Pei, X. Sun, E. Chen, and H. Li. Context-aware ranking in web search. In *SIGIR '10*, pages 451–458. ACM, 2010.