Web Query Recommendation via Sequential Query Prediction

Qi He [†1], Daxin Jiang [§2], Zhen Liao [§3], Steven C.H. Hoi [†4], Kuiyu Chang [†5], Ee-Peng Lim [†6] , Hang Li [§7]

[†]*School of Computer Engineering, Nanyang Technological University, Singapore 639798*

{ [1]qihe, [5]kuiyu.chang}@pmail.ntu.edu.sg, { [4]CHHoi, [6]ASEPLim}@ntu.edu.sg

[§]*Microsoft Research Asia, Beijing, China 100080*

{ [2]djiang, [3]v-zhliao, [7]hangli}@microsoft.com

***Abstract*— Web query recommendation has long been considered a key feature of search engines. Building a good Web query recommendation system, however, is very difficult due to the fundamental challenge of predicting users' search intent, especially given the limited user context information. In this paper, we propose a novel "sequential query prediction" approach that tries to grasp a user's search intent based on his/her past query sequence and its resemblance to historical query sequence models mined from massive search engine logs. Different query sequence models were examined, including the naive variable-length N-gram model, Variable Memory Markov (VMM) model, and our proposed Mixture Variable Memory Markov (MVMM) model. Extensive experiments were conducted to benchmark our sequence prediction algorithms against two conventional pair-wise approaches on large-scale search logs extracted from a commercial search engine. Results show that the sequence-wise approaches significantly outperform the conventional pair-wise ones in terms of prediction accuracy. In particular, our MVMM approach, consistently leads the pack, making it an effective and practical approach towards Web query recommendation.**

I. Introduction

A. Background and Motivation

Web query recommendation is an essential ingredient for a user-oriented search engine. A common fact in Web search is that a user often needs multiple iterations of query refinement to find the desired results from a search engine. This is partially because search queries are often extremely concise (2-3 words on average [15], [37]), and therefore do not adequately and/or distinctively convey users' search intent to the search engine. Query recommendation[1] is thus a promising direction for improving the usability of Web search engines. The explicit task of query recommendation is to help users formulate queries that better represent their search intent during Web search interactions. In addition, query recommendation can potentially be applied unintrusively to existing Web applications such as search relevance enhancement, online advertising, search result presentation, personalized search, and many other Web applications.

Some recent work has used search engine logs to mine "wisdom of the crowd" for query recommendation. For example, in [10], [13], [17], the authors used queries that are adjacent or co-occur in the same query sessions as candidates for recommendation. Although those methods can often provide meaningful recommendations, they only focused on mining the *pair-wise* relations among queries, i.e., predicting the probability of the next user query based only on a *single* preceding/past user query. In this paper, we argue that pair-wise query relations may not sufficiently capture the user context information that is represented by the past queries issued by the same user. Accordingly, we propose a novel *sequential query prediction* approach based on the following intuitions.

First, the pair-wise approach is not sufficient to capture the primary contextual information in a session. Previous empirical studies [14], [32] have estimated the average length of a query session to be $2 \sim 3$. For example, Jansen et al. [14] investigated three main approaches in session segmentation on the search logs of "www.Dogpile.com" and estimated the average length of a query session to be 2.85, 2.31, and 2.31 respectively. In addition, it was shown that AltaVista users got used to submitting slightly longer sessions. These studies suggest that in many cases there are more than one query submitted immediately preceding the current query. Therefore, we need a more general approach not limited to pair-wise relations, but instead is capable of capturing the context information represented by a variable number of queries.

Second, many query sessions can only be correctly modeled by treating the previous queries *sequentially*. In our empirical study, we randomly picked up 20,000 query sessions from our search log data and asked 30 labelers to manually classify them to seven common types of search patterns proposed in [26], [35]. Figure 1 shows the distribution of the search patterns in our user study and Table I gives some example for each type of the search patterns. From Table I, we see

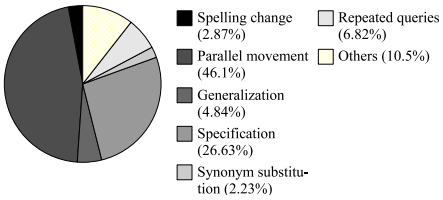


Fig. 1. Distribution of seven types of query session patterns.

that at least three types of patterns, including *Spelling change*,

[1]We use "query recommendation" to refer to "Web query recommendation".

TABLE I
SOME SAMPLE SEARCH SEQUENCE PATTERNS.

search sequence pattern	*example*
Spelling change	goggle⇒ google
Parallel movement	SMTP⇒ POP3
Generalization	Washington mutual home loans⇒ home loans
Specialization	O2⇒ O2 mobile ⇒ O2 mobile phones
Synonym substitution	BAMC ⇒ Brooke Army Medical Center
Repeated query	aim ⇒ myspace ⇒ myspace ⇒ photobucket
Others	muzzle brake ⇒ shared calenders

Generalization, and *Specialization*, are directly related to the order of queries in sessions. To adequately understand such order-sensitive search patterns, which together account for 34.34% of all search patterns as shown in Figure 1, we clearly need to model query sessions as *sequences* of queries instead of *bags* or *pairs* of queries.

Third, looking up the context information may reduce the ambiguity of queries and thus improve the accuracy of query recommendation. Consider the following intuitive example: suppose a user issued a "Java" query, it is hard to determine whether this is about the Java language or Java island. However, if we know that one of the queries preceding "Java" is "Indonesia", then we can determine that the user is more likely to be interested in Java island of Indonesia. In our empirical study, we further quantitatively investigate the correlation between the current query and its context using the entropy measure. A lower entropy indicates a lower ambiguity of a query given its context. For example, suppose the query "Java" appears 100 times, following by "Sun Java" 60 times and "Java island" 40 times, then the prediction entropy of "Java" is 0.29[2]. Now, suppose we are given the context "Indonesia" before "Java", and we observe "Java island" following the sequence "Indonesia ⇒Java" 9 times and "Sun Java" only once, then the entropy drops from 0.29 to 0.14. Figure 2 shows the average prediction entropy of queries given various context lengths (i.e., number of past queries). Not surprisingly, the

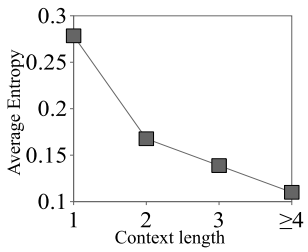


Fig. 2. Average prediction entropy versus context length.

curve drops dramatically when the length of context increases. This observation confirms that the probability of each query conditionally depends on the sequence of past queries as a whole.

B. Overview of Our Approach and Contributions

Our sequence-based approach to query recommendation consists of two phases. In the *offline model learning phase*, we treat each session from the search log data as a sequence of queries and build a probabilistic prediction model. In the *online query recommendation phase*, we feed the observed query context from a user to the prediction model and suggest the top N (e.g. $N = 5$) queries with the highest prediction scores as query recommendations.

In our approach, the online query recommendation phase is straightforward, and the key issue is which model to choose for the particular sequential query prediction problem. We surveyed a wide range of statistical models and narrowed down our choice to the family of Markov models and their extensions. This is because Markov models are parametric approaches to accurately estimate sequence distributions, and have proven successful in modeling complex sequences in the field of natural language processing and biological gene sequence analysis. Among various Markov models and their extensions, the N-gram model [8] is one of the fundamental models. We focused on the N-gram and its variations in this paper instead of other more complicated ones like the Hidden Markov Model (HMM) [25], the Maximum Entropy Markov Model (MeMM) [23], and the Conditional Random Field (CRF) model [20] due to two considerations. First, for the problem of sequential query prediction, we are only interested in predicting the next query a user is likely to ask rather than labeling/predicting an entire follow-up sequence of observations. Second, for our current formulation of sequential query prediction, we can directly model queries or query sequences as states, and do not yet assume them to be generated from some hidden states.

We first examined the naive variable-length N-gram which sticks to the maximum length context. To be specific, suppose we observe a sequence of user input queries $[q_1, \ldots, q_{i-1}]$ and we would like to predict the user's next query q_i, we will search the training evidence of $[q_1, \ldots, q_{i-1}]$ from the fixed i-gram model, where i varies over user inputs. Relying on the full context has the problems of low coverage and over-fitting. A variation of back-off N-gram [18], called the Variable Memory Markov (VMM) model [28], was then investigated. The VMM models allow back-tracking along uncovered suffix contexts. For example, if a context $[q_1, \ldots, q_{i-1}]$ cannot be found in the training data, the next shorter context $[q_2, \ldots, q_{i-1}]$ will be checked recursively. In addition, the VMM models target at determining a bound D on the maximum context length by reducing the information redundancy.

Although our empirical studies showed that the naive N-gram and VMM models are more effective than pair-wise approaches, one challenge still remains: neither model can adapt to the user input on the fly and dynamically determine the optimal length of context used for query prediction. It is a great challenge to determine how many queries in the context we should use to yield the best prediction. On one hand, looking at a small number of queries, e.g., only one preceding query q_{i-1}, will degenerate the model to a pair-wise approach and thereby lose significant context information. On the other hand, blindly incorporating a large number of queries, e.g., all observed and trainable queries in the context, may over-fit the training data and also decrease the coverage of the model. Although the VMM model is more flexible than the naive N-

[2]log base 10 is adopted through the paper.

gram model in the sense that it bounds the optimal length of context from above, it is still non-trivial, if practical, to select a universal upper bound D that will work well for all user queries.

To address the above challenge, we propose a novel sequential probabilistic model, called the *Mixture Variable Memory Markov* (MVMM) model. The basic idea consists of two steps. In the training step, we learn multiple VMM models with different context bounds. Then in the testing step, we construct a mixture model to adapt to the test query sequence on-the-fly. The parameters of the mixture model can be estimated by a simple and effective Newton iteration method.

The contributions of this paper are summarized below: (1) we propose a novel approach of *sequential query prediction* for query recommendation; (2) we build a probabilistic framework for sequential query prediction and develop a new sequential probabilistic model, i.e., MVMM, for solving the query prediction task; and (3) we conduct an empirical study over two pairwise approaches and three sequential prediction models on a large-scale search log containing 1.1 billion unique queries and 2.5 billion sessions. The results show that sequential models have superior performance in terms of prediction accuracy over the baseline methods. Moreover, among the sequential models, our proposed MVMM model achieves the best performance in balancing the tradeoffs between prediction accuracy and prediction coverage.

II. Related Work

Traditional approaches to query recommendation usually rely on user information such as explicit or implicit user feedbacks [27], [22], [9], [36], user profiles [34], [6], and sometimes with the help of semantic analysis via a thesaurus [21]. Some other approaches attempt to understand a user query by analyzing the retrieved search results via hit document content analysis [31], snippets [29], or anchor texts [19].

Several recent work [7], [3], [2], [13], [10], [17] has mined search engine log data for query understanding as well as query recommendation. Compared to traditional methods, log-mining approaches enjoy several advantages: (1) no user effort is needed; (2) search log data contains rich user behavior information complementary to web content information; and (3) models constructed from massive log data are often statistically superior to those built from the relatively limited amount of documents/snippets/anchor content.

In general, two types of information can be extracted from search logs, i.e., click-through information [37], [3], [1] and session information [13], [10], [17]. Both have been used for query understanding.

Approaches based on click-through information assume two queries to be related if they share many clicked URLs. The related queries are usually grouped into clusters and used for recommendations for each other. We call these methods *cluster-based* approaches. For example, Beeferman *et al.* [3], Wen *et al.* [37], and Baeza-Yates *et al.* [1] applied hierarchical, density-based, and k-means algorithms to obtain query clusters, respectively. Although these methods can effectively find similar queries, in query recommendation, it is more interesting to recommend queries that a user may ask next in the query context, rather than suggest queries to replace the current query [5], except for very specialized cases such as spelling correction (e.g., "do you mean ...").

In the more related area of search session mining, two main research problems have been tackled. The first is automatic session extraction [14], [12], wherein Web search patterns such as *spelling change* and *specification* have been used to enhance session extraction [24], [26], [11]. Assuming that query sessions have been reliably extracted, the second research problem, which directly relates to our work, is to predict a user query based on queries already entered in the same session. Existing approaches extract session information and use queries that are adjacent or co-occur in the same sessions as recommendations for each other. For example, Huang et al. [13] used frequently *co-occurring query pairs* in the same session to recommend the next query. Fonseca et al. [10] calculated the *co-occurrence* of queries and implicitly used the order of queries for query expansion. Jones et al. [17] considered frequent *adjacent* query pairs only for query substitution. We call these methods *session-based* approaches.

Our approach can also be classified into the session-based category. However, our work has one fundamental difference from all previous session-based approaches: while all previous work focuses on *pair-wise* query relations and uses only a single preceding query for query prediction, we consider a variable number of preceding queries and effectively capture more complex context information for query recommendation. Moreover, our approach can automatically determine the optimal context length to be used for query prediction.

III. Preliminary Theory for Sequential Query Prediction

A. Notations and Problem Statement

Let Q be the set of all unique queries, and G and T be the training and test set, respectively. Let Q^* be the set of all query sequences over Q, i.e., $Q^* = \{s | s = [q_1, \ldots, q_l], q_i \in Q, 1 \leq i \leq l\}$, and S be the subset of query sequences extracted from G for training purpose by some probabilistic model. In Markov models, S is also the set of states. Note a special case of query sequence s is an empty sequence e where the sequence length l is 0. Let $|Q|$ and $|s|$ denote the cardinality of the query set Q and the length of a query sequence s, respectively. $\not| s \not|$ denotes the frequency of s in a collection. We define the problem of *sequential query prediction* as follows.

Definition 1 (Sequential Query Prediction): Given a collection of search logs G, the task of sequential query prediction is to learn a probabilistic model $\hat{P}$ which provides a probability distribution of a user's next query q_i given a sequence of preceding queries $s = [q_1, \ldots, q_{i-1}]$ raised by the user as the context. In other words, given a context $s \in Q^*$ and a query $q_i \in Q$, the model needs to estimate $\hat{P}(q_i|s)$.

The problem of sequence prediction has been intensively studied in the past years and numerous techniques have been proposed. However, our task of query recommendation has two unique characteristics compared to classical applications of sequence prediction. First, unlike the general requirement of

generating a single item with the highest prediction accuracy, what we need in query recommendation is a ranked list of queries that have both high accuracy and coverage. Second, in classical sequence prediction, the entire sequence is unseen and to be predicted as a whole; while in query recommendation, we predict a user's next query each time a user issues a query to the search engine. Therefore, our prediction task is an incremental process: after each round of user input, we have a growing sequence of observed queries that are increasingly reliable.

B. A Probabilistic Framework for Sequential Query Prediction

In this section, we introduce a probabilistic framework tailored for sequential query prediction. Let $\hat{P}$ be a probabilistic model learned from the training query sequence data G. The prediction performance of $\hat{P}$ with respect to test data T can be measured based on the *average log-loss* $l(\hat{P}, T)$ rate:

$$l(\hat{P}, T) = -\frac{1}{|T|} \sum_{s_t \in T} \frac{1}{|s_t|} \sum_{j=2}^{|s_t|} \log \hat{P}(q_j | [q_1, \ldots, q_{j-1}]), \quad (1)$$

which is intuitively the average of posterior query probabilities over all test sequences $s_t = [q_1, \ldots, q_j]$ of length 2 or larger. It is easy to see that the average log-loss function is directly related to the sum of likelihood $\sum_{s_t \in T} \hat{P}(s_t) = \sum_{s_t \in T} \prod_{j=2}^{|s_t|} \hat{P}(q_j | [q_1, \ldots, q_{j-1}])$. Hence, minimizing the average log-loss function is equivalent to maximizing the likelihood of the test data T[3].

The above log-loss measure can be interpreted within the information theory framework. Suppose G and T are drawn from the same unknown source distribution P, and let $s_t \in T$ be a sequence generating random variable (vector). Clearly, P minimizes the mean log-loss rate over all models:

$$P = \arg\min_{\hat{P}} \{-\mathbf{E}_P \{\log \hat{P}(s_t)\}\},$$

where $\mathbf{E}$ is the expectation operator. Since the true distribution P is often unknown, the difference between P and the estimated distribution $\hat{P}$ from training data gives rise to what is known as "redundancy" in information theory, which can be measured by the *Kullback-Leibler* (KL) divergence:

$$D_{KL}(P||\hat{P}) = \mathbf{E}_P \{\log P(s_t) - \log \hat{P}(s_t)\}.$$

The theoretical implication for our task of sequential query prediction is thus as follows. We aim to uniformly minimize the information redundancy across all possible generating sequence distributions in the search query logs via a probabilistic model $\hat{P}$.

IV. Markov Models for Sequential Query Prediction

In this section, we first introduce two classical Markov models, i.e., the naive variable-length N-gram model and its extension, the VMM model. We then propose a novel mixture VMM model to address the particular challenges in the problem of sequential query prediction.

[3]The first query is assumed to be given, i.e. $\hat{P}(q_1) = 1$, since it is meaningless to make a recommendation before the user submits any query.

A. Variable-length N-gram Model

The N-gram model is a well-known technique widely used in natural language process and genetic sequence analysis. Moreover, Su et al. [33] have successfully applied the simple N-gram models to predict the next user click action based on the server logs. N-gram by definition is a sub-sequence of N items extracted from any given sequence. The model built from N-grams is also known the $(N-1)$-order Markov model, i.e., the current query depends only on the last $N-1$ queries. The use of N-gram models for a general prediction is rather simple. For a set of unique query sequences Q^*, an N-gram model over Q^* consists of a set of states $S = \{s | (s \in Q^*) \wedge (|s| = N-1)\}$. Given a sequence $[q_1, \ldots, q_{i-1}]$, an N-gram model predicts q_i using the previous $N-1$ states $[q_{i-N+1}, \ldots, q_{i-1}]$. Learning for an N-gram model simply refers to estimating the conditional probability distribution of $P(q_i|s)$ for each state $s \in S$. Given the collection of search session data G, we can easily learn an N-gram model via the *Maximum Likelihood Estimation* approach.

However, selecting a universal N is rather difficult in practice, especially when the user could submit various number of queries as the context. For sequential query prediction, we actually train a series of N-gram models of various length. If the user has submitted $i-1$ past queries, an N-gram model of $N = i$ will be selected for prediction. We call such a model *variable-length N-gram*[4]. By setting N to be 2, sequential query prediction via N-gram model degenerates to the *Adjacency* pair-wise method.

Remark. Although the naive N-gram model has been successfully used in language modeling, it has some serious limitations when applied to our sequence query prediction task. First, we have to train many N-gram models of various N, each of which over Q has a size of the order $|Q|^{N-1}$. In practice, search logs over a time period could contain billions of queries, which makes an N-gram model with a large N impractical. Second, an N-gram model with a large N may severely overfit the training data, thereby yielding low prediction coverage, i.e., training samples decreases exponentially with increasing user contexts, as shown later in the experiment sections. On the other end of the spectrum, an N-gram model with a small N (i.e., $N = 2$, degenerated to *Adjacency*) loses too much context information. Therefore, in practice, a compromise must be made in selecting a suitable N-gram model, where N might be less than the number of queries submitted by the user.

B. VMM Model

We consider the Variable Memory Markov (VMM) Model [28] more suitable for the task of query prediction than the naive N-gram model since it does not fix the length of context to be the number of past test queries. Indeed, the VMM model is a variation of back-off N-gram by bounding the context length on a need basis and allowing partial matches for uncovered context when applied to a test query. VMM has been used successfully for general sequence prediction [4].

[4]We use "(naive) N-gram model" to refer to "variable-length N-gram model" where no confusion will be caused.

A VMM learning algorithm can often achieve a *bounded redundancy*, if its context length does not exceed D.

1) Learning VMM via Prediction Suffix Tree: In the following, we introduce the Prediction Suffix Tree (PST) [28], [30] algorithm to build a VMM, which enjoys decent time/space complexity bounds[5]. In the original PST algorithm [28], up to 5 parameters must be tuned. For simplicity, we only tune the ϵ parameter, which controls the PST growth rate and will be introduced later.

We give a simple example for illustrating the PST algorithm below. Given a set of query sessions for training as shown in Table II, $Q = \{q_0, q_1\}$, and the maximum length of context D is 2 since the last query in any query sequence has no prediction evidence. Therefore, $Q^* = \{q_0q_0, q_0q_1, q_1q_0, q_1q_1, q_0, q_1\}$. The VMM PST model is learned in the following 3 stages:

TABLE II
SAMPLE SESSION TRAINING DATA.

s	$\#s$	s	$\#s$	s	$\#s$	s	$\#s$
$q_1q_0q_0$	3	$q_1q_0q_1$	7	q_0q_0	78	q_1q_0	5
$q_0q_1q_0$	1	$q_0q_1q_1$	1	q_1q_1	3	q_0	10

(a) Extract a candidate suffix set $S' \subset Q^*$ from the training sequences. If the PST is D-bounded, all sequences in S' have lengths $\leq D$. A user threshold could be set to filter those infrequent training sequences. For each candidate $s \in S'$, we associate the conditional probability to each predictable query $q \in Q$ by counting the occurrences of s and $[s, q]$. For example, $P(q_0|[q_1, q_0]) = 3/10$. Without filtering, we have $S' = \{q_1q_0, q_0q_1, q_0, q_1\}$.

(b) Determine the suffix set[6] $S \subset Q^*$ for training the VMM model via variable-length modeling. The process of deciding the suffix set S exactly follows the construction process of a PST via a depth-first search, where each node corresponds to a state in the learnt VMM. Given two sequences s and s', s' is a descendant of s in PST if s is a suffix of s'. Starting from the empty sequence e, we evaluate each $s \in S'$ and add s and its suffixes to the PST if and only if s satisfies either of the following 2 criteria:

- $|s| = 1$: add all unique queries (in this example, q_1 and q_2) into the PST.
- KL divergence of the predictive probability of s including its parent is greater than a threshold ϵ.

The criteria for adding a sequence varies across different applications. For example, Schutze and Singer [30] added one more condition for Part-of-Speech tagging: if none descendant of s up to a certain length can be added to the tree, s will be added. This condition is not suitable for sequential query prediction because it will almost add every candidate sequence into the PST since the average length of query sessions is short (less than 3) in search logs. On another front, Ron et al. [28] uses a different threshold instead of the KL divergence. By setting $\epsilon = 0.1$, we have for our example $S = \{q_1q_0, q_0, q_1\}$, $D_{KL}(q_0||q_1q_0) = 0.3449$, and $D_{KL}(q_1||q_0q_1) = 0.0837$.

(c) Finally, the conditional probabilities for unobserved sequences are uniformly assigned a minimum constant probability of $1/|Q|$. After smoothing, the conditional probabilities are normalized to sum to 1. In our example no unobserved events exist.

Figure 3 plots the PST for our toy example, where the conditional probabilities given the empty sequence e is based on the *priori* probability of each query.

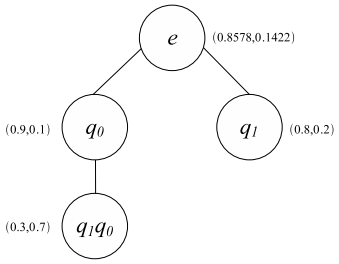


Fig. 3. PST built from sample data in Table II. Each node is labeled with a sequence s and its estimated probabilities ($P(q_0|s)$, $P(q_1|s)$).

2) Online Query Recommendation: The primary advantage of PST lies in its extremely fast online prediction speed ($O(D)$). Given any test query sequence context s, its corresponding path or maximum length suffix s' in the PST can be traversed in linear time. For the example in Figure 3, the probability of a test sequence such as $[q_0, q_1, q_0, q_1, q_1, q_0]$ is $1 \times 0.1 \times 0.8 \times 0.7 \times 0.2 \times 0.8$, and the labels of the states that are used for the predictions are $s^0 = e$, $s^1 = q_0$, $s^2 = q_1$, $s^3 = q_1q_0$, $s^4 = q_1$, $s^5 = q_1$. Note that the first query q_0 is assumed to be deterministic.

Clearly, query recommendation is a simple extension of the PST traversal. For example, when a user submits a query q_0, we will recommend the query q_0 to the user if only one query is required for recommendation. If the user has submitted 2 past queries $[q_1, q_0]$, the query q_1 will be recommended.

C. Mixture Variable Memory Markov Model (MVMM)

In this section, we first examine the drawbacks of classical general-purpose sequential probabilistic models, as motivation for developing our new approach, i.e., the Mixture Variable Memory Markov (MVMM) model. Finally, we present techniques for learning the MVMM parameters and discuss its application to query recommendation.

1) Limitations of the VMM Model: Although the VMM model with its variable length context is superior to the naive N-gram model, it still suffers some practical limitations. Most VMM learning algorithms, including the PST, are essentially D-bounded back-off N-gram models based on maximum likelihood estimation. We claim that for such a D-bounded VMM, there are at least two nontrivial shortcomings.

(a) The PST learning algorithm parameters are hard to be optimized in practice. For example, the overall PST performance is very sensitive to the growth parameter ϵ. A slight change to ϵ would result in vastly different D-bounded VMM models. There are two extreme settings of ϵ, as shown in

[5]A D-bounded PST incurs a training time and space complexity of $O(|Q^*| \cdot Dn^2)$ and $O(|Q^*| \cdot Dn)$, respectively, where n is the average query session length. Prediction is linear in D.

[6]Given a suffix set S, if context s appears in S, then all suffixes of s must also be in S.

Figure 4, $\epsilon = +\infty$ and $\epsilon = 0$, which will generate an Adjacency (2-gram) model and an infinitely bounded VMM model, respectively. Clearly, a moderate value of ϵ is desirable

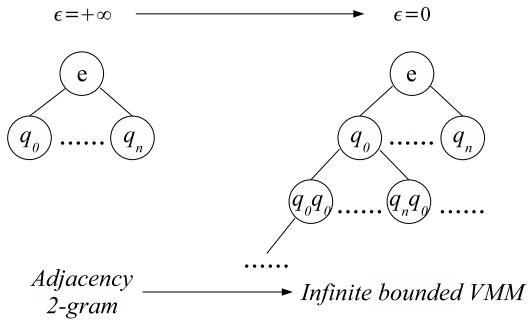


Fig. 4. Two extreme cases of VMM.

to avoid the loss of context information or over-fitting at both extremes. Unfortunately, the best ϵ must be determined experimentally.

(b) The PST algorithm requires an extra escape trick. Following the example in Figure 3, when the user submits q_1q_1, the state used for prediction is $s = q_1$ as it is the longest suffix of q_1q_1 which can be found in the PST state set. However, there is actually a context disparity between the user input context q_1q_1 and the closest training context q_1. We thus need a smoothing strategy to eliminate such disparities while generating the probability given the pseudo context q_1 instead of the true underlying q_1q_1 context. The context disparity could be a consequence of either a low value of ϵ or the lack of corresponding training contexts.

2) The MVMM Approach: To overcome the shortcomings of VMM, we propose a Mixture Variable Memory Markov model (MVMM), which is a linearly weighted combination of multiple VMM models of varying bounds D.

(a) Selecting ϵ. Although we can arrive at a decent ϵ via expensive cross-validation process, we could still lose the user's online context information due to the D bound. For the example of Figure 3, irregardless of whether the user context is q_1q_1 or $q_1q_1q_1$, the state q_1 will always be selected. In other words, once a D-bounded VMM has been trained, any online test context of length $> D$ will be lost. On the other hand, an infinitely bounded VMM can model any user context, at the expense of over-fitting the training data.

In fact, the choice of ϵ is a typical model selection problem for balancing the prediction accuracy with recall/coverage. A larger bound D will invariably model more complex contexts, thereby improving the prediction accuracy, while a smaller bound D will result in better recall. Given any online user context, the ideal solution is to dynamically choose the appropriate model of bound D generated on-the-fly, which is of course infeasible in practice.

A practical compromise is to train various D-bounded VMM models ahead of time. One implicit assumption made here is that *test model selection is dictated solely by test context lengths*, which is fairly reasonable. Each of the K D-bounded VMM models, $\{\hat{P}_D, D = 1, \ldots, K\}$ are trained with a range of ϵ values.

After training, the test probabilities are estimated using a mixture of all D-bounded VMM models as follows:

$$\hat{P}(T) = \sum_D w(D,T) \cdot \hat{P}_D(T), \tag{2}$$

where $w(D,T)$ is a mixing weight function. For a context $s = [q_1, \ldots, q_l], s \in T$, we compute its probability by

$$\hat{P}_D(s) = \prod_{i=2}^{l} \hat{P}_D(q_i|[q_1, \ldots, q_{i-1}]). \tag{3}$$

The above process leads to a Mixture Variable Memory Markov model (MVMM). Given a test sequence s, each D-bounded VMM will output the best matching state s_D. The remaining issue of MVMM is to determine the weighting function $w(D,T)$, which should intuitively be proportional to the degree of agreement between s and s_D. We model the distribution of $w(D,T)$ with a 1-D Gaussian function:

$$w(D,T) = \frac{1}{\sigma_D\sqrt{2\pi}} \exp\Big(\frac{-d(T)^2}{2\sigma_D^2}\Big), \tag{4}$$

where $d(T)$ is defined as the edit distance between s and s_D, and σ_D is the sample data variance.

(b) The context disparity. In the event that a user context is new and never seen before in the training data, the PST should still output a partial match. The standard way of handling zero-frequency queries is to smooth the probabilities for the unobserved queries, or *escape* or skip to the next matching query. We chose the latter approach, the context escape mechanism, for our D-bounded VMM, which works as follows.

For each unobserved context $s = [q_1, \ldots, q_l]$, we allocate a probability $\hat{P}(escape|s)$ for the case that q_1 is possibly new. The residual probability $1 - \hat{P}(escape|s)$ is shared by all other queries $q \neq q_1$ that appear before $[q_2, \ldots, q_l]$. The conditional probability for any query q is thus defined recursively as follows:

$$\hat{P}(q|s) = \begin{cases} \hat{P}_D(q|s), & \text{if } s \text{ can be found in } S; \\ \hat{P}_D(escape|s) \cdot \hat{P}(q|[q_2, \ldots, q_l]), & \text{otherwise.} \end{cases} \tag{5}$$

The escape mechanism, originally used for smoothing unobserved queries in VMM, is borrowed here to recursively bridge the context disparity between $[q_1, \ldots, q_l]$ and $[q_2, \ldots, q_l]$. The probability of escaping from context $s = [q_1, \ldots, q_l]$ to its suffix context $s' = [q_2, \ldots, q_l]$ is defined as

$$\hat{P}_D(escape|s) = \frac{\not|[e, s']\ \not|}{\sum_{q \in Q}\ \not|[q, s']\ \not| + \not|[e, s']\ \not|}. \tag{6}$$

The context escape mechanism actually penalizes partial match in VMM. That is to say, the approximated escape probability of $\hat{P}(q|s)$ should be less than the data-estimated probability of $\hat{P}(q|s')$ if s is unobserved. For a single VMM model, such escape is pointless because the conditional distribution of s will not be affected after re-normalization. But for a mixture VMM model, since we can only combine the prediction results based on weighted generative probabilities, the escape mechanism will thus penalize the partial matching models.

3) Learning the Mixture Parameters of MVMM: One important issue of the MVMM model is to determine appropriate weighting parameters for combining multiple VMM models. This motivates us to investigate effective techniques for learning the weighting parameters from the training data automatically.

Recall that the goal of our learning task is to minimize the *redundancy* in Eq.(1). By following the same principle, we can formulate the problem of learning the optimal weighting parameters $\mathbf{w}$ of MVMM as follows:

$$\mathbf{w} = \arg\min_{\mathbf{w}} D_{KL}(P||\hat{P}_{\mathbf{w}}) \tag{7}$$

where P is the true distribution for generating the sequential data and $\hat{P}_{\mathbf{w}}$ is the MVMM generative probabilities. We can rewrite the right hand side of the above optimization explicitly as:

$$\min_{\mathbf{w}} \sum_{T=1}^{n} P(X_T) \log \frac{P(X_T)}{\sum_{D=1}^{K} w(D, X_T)\hat{P}_D(X_T)} \tag{8}$$

where $P(X_T)$ is the generative probability estimated from the training data and $\hat{P}_D(X_T)$ is the generative probability of the sequence X_T by a D-bounded VMM model. By adopting the parametric model in Eq.(4) for $w(D, X_T)$, we can rewrite the optimization as follows:

$$\max_{\boldsymbol{\sigma}\in\mathbf{R}^D} \sum_{T=1}^{n} P(X_T) \log \sum_{D=1}^{K} \frac{1}{\sigma_D\sqrt{2\pi}} \exp\left(\frac{-d(X_T)^2}{2\sigma_D^2}\right)\hat{P}_D(X_T) \tag{9}$$

Since the objective function is convex, we can find the global solution for the optimization problem by solving it iteratively with the classical gradient descent (Newton) method:

$$\boldsymbol{\sigma}^{(t+1)} = \boldsymbol{\sigma}^{(t)} - [Hf(\boldsymbol{\sigma}^{(t)})]^{-1}\nabla f(\boldsymbol{\sigma}^{(t)}) \tag{10}$$

where f is the objective function and H is the Hessian matrix.

After the mixing weights have been computed, the online portion for query recommendation is relatively straightforward: *Depending on the length of user context, a linear weighting could be quickly computed on all (partially matched) VMM components. The predicted queries of all VMM components are combined and re-ranked w.r.t. their generative probabilities and model weights. Finally, the top N results are recommended to the user.*

V. Experiments

We benchmark the *coverage* and *accuracy* metrics of the three sequential models, N-gram (Section IV-A), VMM[7] (Section IV-B), and MVMM (Section IV-C.2) against two pair-wise baseline models.

A. Dataset

1) Data Format: We used a 150-day search logs extracted from a commercial search engine, whose format is shown in Table III.

[7]VMM refers to a general VMM model under any parameter settings. Specifically, VMM (0.05) names a VMM model with $\epsilon = 0.05$ of infinite order, and 2-bounded VMM (0.05) means a VMM model with $\epsilon = 0.05$ and $D = 2$.

TABLE III
A raw data example of the search logs.

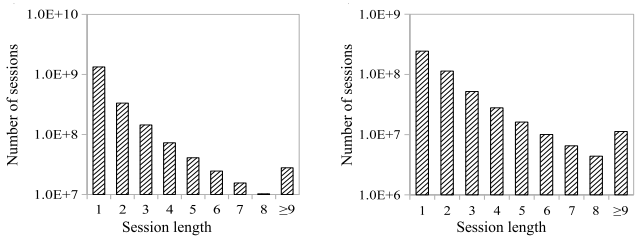
machine ID	query timestamp	query	# clicked URLs	timestamp of click on URL 1	URL 1	...
xxx	00:08:41	q_1	1	00:09:06	aaa.com	-
xxx	00:10:55	q_2	2	00:11:23	bbb.com	...

2) Session Segmentation: Both machine IDs and timestamps were used as cues to detect any change in a user's search context. A typical user search log is comprised of a stream of queries and clicks, with each query followed by a variable number of clicks. A user search log could be further segmented into *sessions*, with each session relating to one specific user information need. Since session segmentation is beyond the scope of this paper, we adopt the 30-minute rule convention [38], [14] by cutting at time-points where more than 30 minutes have passed between an issued query and URL click.

We use the first 120-day (4 months) data for training and the following 30-day (1 month) data for testing. Table IV lists some session statistics while Figure 5 plots the histogram for session count versus session lengths. Note that there are quite a number of (tens of millions) long sessions comprising more than four queries, two of which are shown in Table V.

TABLE IV
Summary statistics of segmented sessions.

Data	# Sessions	# Searches	# Unique queries
training	2,002,409,554	3,860,798,910	1,125,875,693
test	486,184,930	1,102,802,397	356,070,833



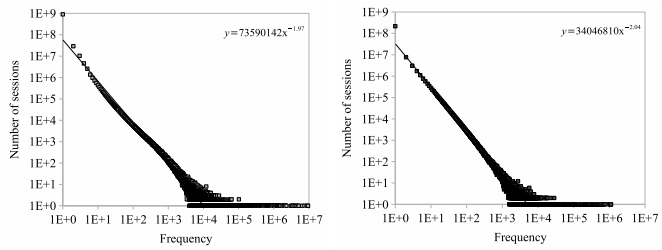
(a) Training data (b) Test data

Fig. 5. Session count versus Session length.

TABLE V
Sample sessions.

Length	Session
2	sign language ⇒ learn sign language
3	kidney stones ⇒ kidney stone symptoms ⇒ kidney stone symptoms in women
4	Nokia N73 ⇒ Nokia N73 themes ⇒ free themes Nokia N73⇒ Nokia N73 free themes download
5	www.disney.com ⇒ Disney channel ⇒ Disney channel games ⇒ Disney channel games kids ⇒ play Disney channel game

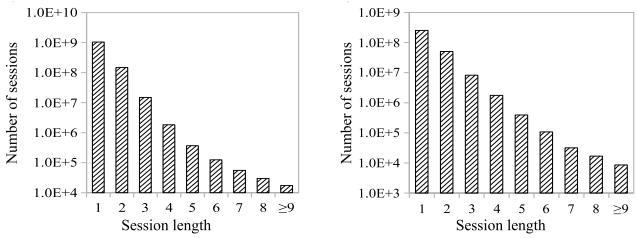
3) Session Aggregation: After session segmentation, identical sessions from different users are aggregated. Figure 6 plots the aggregated session count versus aggregated session frequency, which clearly follows the power law distribution.



(a) Training data (b) Test data

Fig. 6. Power law distribution of unique aggregated sessions.

4) Data Reduction: From Figure 6, we observe a large number of aggregated sessions (40%) with frequency less than or equal to 5. These are most likely rare (one-time) and/or erroneous sessions, which can be safely discarded. The remaining set of aggregated sessions[8] largely maintain the same distribution as the original set, as shown in Figure 7, except for the super-long sessions that were discarded.



(a) Training data (b) Test data

Fig. 7. Session count versus Session length after data reduction.

5) Aggregating Training Contexts: For each aggregated session, the training data context is build as shown in the following example. Consider an aggregated session sequence $[q_1, q_2, q_3, q_4, q_5]$ with a frequency of 10. Four training contexts can be derived: $[q_1]$, $[q_1, q_2]$, $[q_1, q_2, q_3]$, and $[q_1, q_2, q_3, q_4]$, each with a support of 10; the support for predicting q_2 from observing $[q_1]$ is 10, the support for predicting q_3 after observing $[q_1, q_2]$ is also 10, etc. Training contexts are again aggregated over all sessions and fed into the training model.

6) Ground Truth for Test Set: The same approach for building the set of aggregated training contexts was used to create the set of aggregated test contexts, which is also considered the ground truth for the test set. Specifically, given a test user context $s = [q_1, q_2, \ldots, q_m]$, we counted the frequencies of all queries that immediately follow s in the test data; the top n queries are considered to be ground truth. In our experiments, we set n to 5 since we are interested in recommending up to 5 queries.

B. Baseline methods

We use the following two pair-wise methods as baselines.

- Adjacency (*Adj.*): Given a test query q, this method computes a ranked list of queries that immediately follows q in the training set. This approach was used in [17] for a slightly different purposes: implicit query substitution/expansion (with no user choice, i.e., forced).
- Co-occurrence (*Co-occ.*): Given a test query q, this method computes a ranked list of queries that co-occurs with q in the training set. This approach was used by [13] for real time query term suggestion, i.e., while the user is typing the query.

[8] 60.48% and 64.72% of training and test data remained, respectively

C. Evaluation Metrics

Query prediction results can be evaluated in terms of *coverage* and *accuracy*.

1) Coverage: Query coverage is defined as the ratio of predictable query sequences over all sequences in the test set. For any given query prediction model created from a finite training set, there are bounds to be some queries that exist exclusively in the test set. A good query prediction model aims to achieve as high a coverage as possible on the test set. Suppose a test query sequence $[q_1, q_2]$ does not appear in the training data, then it will not be covered by the 3-gram model. However, if q_2 is a valid context in the training data, it would be covered by the *Adj.* and *Co-occ.* pair-wise models.

2) Accuracy: Query prediction accuracy is computed using the Normalized Discount Cumulative Gain (NDCG) [16], which measures the degree of correspondence between two ranking lists by assuming that higher ranked positions carry more importance. The NDCG value of a ranking list at position n is calculated as

$$N(n) = Z_n \sum_{j=1}^{n} \frac{2^{r(j)} - 1}{log(1+j)}, \tag{11}$$

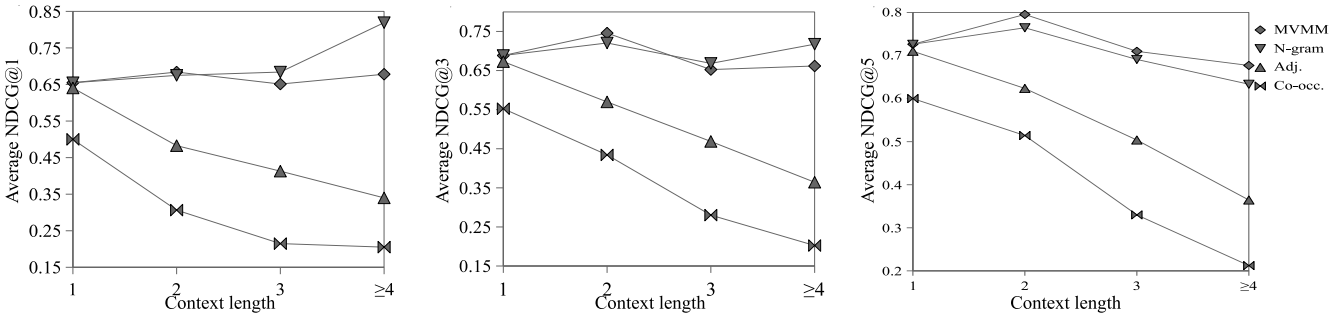
where $r(j)$ is the rating of the j-th query in the ranking list, and the normalization constant Z_n is chosen so that the perfect list gets a NDCG score of 1. In Eq. 11, $2^{r(j)} - 1$ is the gain (G) of the j-th query, $\frac{2^{r(j)}-1}{log(1+j)}$ is the discounted gain (DG), $\sum_{j=1}^{n} \frac{2^{r(j)}-1}{log(1+j)}$ is the discounted cumulative gain (DCG) at position n of the list, and finally $Z_n \sum_{j=1}^{n} \frac{2^{r(j)}-1}{log(1+j)}$ is the normalized discounted cumulative gain (NDCG) at position n of the list, which is called NDCG@n.

To calculate the NDCG score, we define the weightage of each ranked position as $\{5, 4, 3, 2, 1\}$ for the queries at position 1, 2, 3, 4, and 5 in the ground truth test context, respectively. Queries beyond the top 5 list in the ground truth are assigned a weight of 0. For example, given the ground truth context $[q_1, q_2, q_3, q_4, q_5]$, the rating of q_1 is 5, q_5 is 1, and $[q_i : \forall i > 5]$ is 0. The NDCG scores over all test contexts are averaged to yield a single qualitative metric for each evaluated query prediction approach.

D. Query Prediction Accuracy

We benchmark the query prediction accuracy of our proposed method MVMM against a few baseline methods in a two-part comparison.

First, we evaluate our MVMM sequence approach against conventional pair-wise approaches. We trained a MVMM made up of a mixture of 11 VMM models by varying ϵ along the range of $\{0.0, 0.01, \ldots, 0.1\}$. For comparison, only



(a) NDCG@1 (b) NDCG@3 (c) NDCG@5

Fig. 8. Query prediction accuracy of pair-wise (*Adj.*, *Co-occ.*) and sequence based (MVMM, N-gram) methods.

3 representative VMM models will be shown, namely, VMM (0.0), VMM (0.05), and VMM (0.1).

Figures 8(a)-(c) show the NDCG@1, NDCG@3, and NDCG@5 metrics over different context lengths for the various approaches. From the results, we can make the following observations.

- Sequence based methods achieve up to 40% higher accuracy than the pair-wise methods at all positions and across all context lengths.
- The Adjacency method has a consistently better (10%) accuracy than the Co-occurrence method. This is because the former considers the position relationship of queries while the latter does not. This observation somewhat lends credit to our choice of considering full sequence order in our VMM based methods.
- The accuracy metrics of *Adj.* and *Co-occ.* decreases monotonically with increasing context length. This is expected because longer query contexts usually correspond to more specific information need, which cannot be modeled by simple pair-wise generalizations that only look at the most recent query for prediction. In other words, more query history is thrown away with increasing context length.
- The accuracy curves of N-gram and MVMM are less consistent. Generally, both peaked at context length 2, except for N-gram's NDCG@1, which improves with increasing context length.

Next, we benchmark MVMM against VMM with varying parameter settings, as shown in Figures 9(a)-(c). Some insights can be obtained on this comparison, listed as follows.

- VMM (0.05) is the overall winner across all context lengths for up to top 3 (NDCG@3) predicted queries.
- The accuracy of VMM (0.1) deteriorates significantly with increasing context lengths.
- The full-size PST training model, VMM (0.0), suffers from over-fitting, which is especially obvious if only 1 query is predicted (NDCG@1).
- From the above, we can deduce that the best accuracy is achieved at $\epsilon \approx 0.05$. The VMM model is rather sensitive to the parameter selection: a slight change in ϵ could result in significantly different performance.
- After adapting the model selection to the user's context, the MVMM model was able to achieve comparable accuracy as the best VMM (0.05), even beating the latter for the top 5 suggested queries (NDCG@5). This is a huge practical advantage for MVMM, which does away the hassle of finding the best parameter via trial-and-error for VMM.

E. Query Prediction Coverage

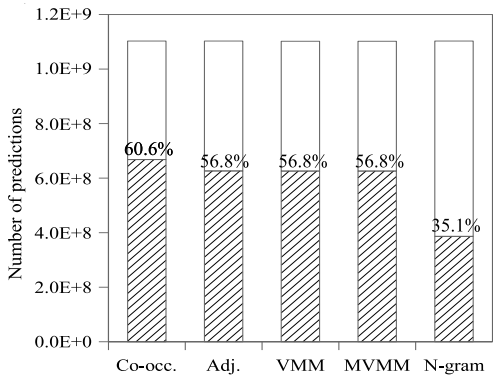
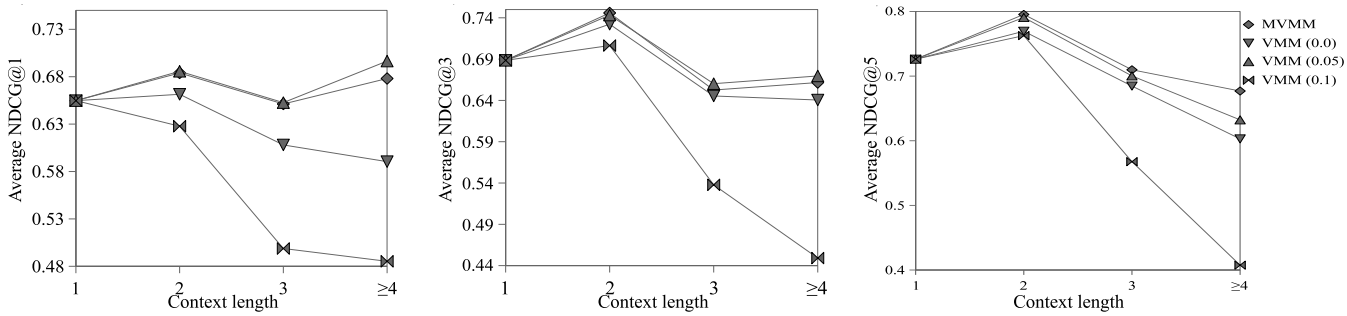


Fig. 10. Coverage of various methods on test data.

Figure 10 shows the coverage achieved by the different prediction methods, from which we can make the following observations.

- The best achieved coverage is 60.6%, using *Co-occ.* as expected, which does not consider ordering information. The 3 sequence approaches *Adj.*, VMM, and MVMM tied for a close second at 56.8%.
- The coverage of VMM and MVMM is equal to *Adj.* due to the partial match strategy adopted by VMM based methods.
- The coverage of N-gram is by far the worst, because it models full and fixed context sequences, which are often too specific (over-fitting) given the limited training data.

Figure 11 plots the coverage versus context length curve. Naturally, coverage of all three methods decreases with increasing test context length. This is because the longer the context, the lower the chance that a matching context can be found from the training data. However, among the three methods, the coverage of VMM and MVMM decreases sub-linearly with increasing context length, maintaining a respectable 45%. On the other hand, N-gram quickly deteriorates to less than 1% coverage for context length longer than 3.



(a) NDCG@1 (b) NDCG@3 (c) NDCG@5

Fig. 9. Query prediction accuracy of MVMM and VMM.

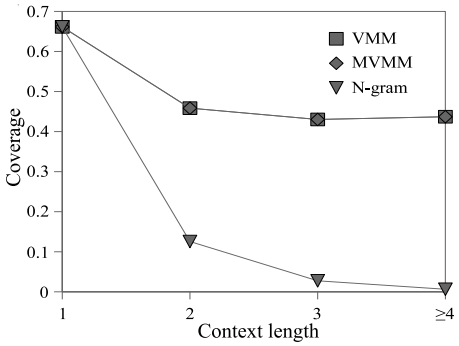


Fig. 11. Coverage versus context length for sequence-wise models.

In summary, N-gram, despite its slightly higher accuracy, is not practical due to its outrageously low coverage. Co-occurrence and Adjacency yield decent coverage but low accuracy. Therefore we can conclude confidently MVMM provides the best accuracy and reasonably good coverage.

To conclude our coverage analysis, we summarizes in Table VI the main reasons for which a test query q cannot be predicted given user context s.

TABLE VI
REASONS FOR UNPREDICTABLE QUERIES.

Models	Reasons
Co-occ.	(1) q is a new query; or (2) q only appears in training sessions of length one.
Adj.	(1), (2) above; or (3) q only appears at the last position of the training sessions.
VMM	(1), (2), (3) above.
MVMM	(1), (2), (3) above.
N-gram	(1), (2), (3) above; or (4) user context s is not a trained N-gram state.

F. Space Complexity Analysis

We are mainly concerned with the memory requirements of the VMM model and its variation, MVMM.

1) Training: MVMM requires roughly K (number of mixture models) times the memory footprint of a single VMM. Fortunately, each of the K models can be independently trained in parallel over a distributed computing facility such as a grid or cluster.

2) Online Deployment: The PST learnt by a trained VMM model must be loaded into RAM for real-time online query prediction. Therefor, the size of the PST (# nodes) gives a reasonable estimate of the space requirement of each VMM model. To avoid loading all K PSTs into memory for the MVMM, we can actually combine all into a single PST, where each node requires just 4 extra bits ($2^4 > 11$) to record its source VMM models. For example, the PST of 2-bounded VMM (0.1) has 6,910,940 nodes and the PST of 3-bounded VMM (0.2) contains 6,854,439 nodes, yet their mixture model (MVMM)'s PST contains just 7,211,288 nodes. Apparently, the MVMM used in our experiments has the same number of PST nodes as VMM (0.0), which is already the full-sized infinite bounded VMM. The memory footprint of each method is summarized in Table VII. We see that the MVMM only requires marginally more memory compared to the standard VMM models. In generally, all VMM models require approximately twice the amount of memory compared to pair-wise or N-gram models, due to the need to maintain a PST in memory. If a memory space is limited, a D-bounded VMM or MVMM can be used.

TABLE VII
MEMORY FOOTPRINT FOR ALL METHODS (UNIT: MEGABYTE).

MVMM	VMM (0.0)	VMM (0.05)	VMM (0.1)	*Adj.*	*Co-occ.*	N-gram
363.4	348.6	330.5	329.7	151.6	233.7	170.1

G. Time Complexity Analysis

The primary disadvantage of MVMM compared to VMM lies in its K-fold training time. For example, each of the K=11 VMM mixture components in MVMM has to be first trained independently, before the cost function can be optimized. Figure 12 plots the graph of training time versus amount of training data for all methods. We see that the MVMM training time is an order of magnitude over the various VMM models. N-gram and *Adj.* have the lowest training time since they have the fewest training evidences, and *Co-occ.* has a slightly higher time complexity because it always recommends more queries. All the three VMM models incurred comparable training time,

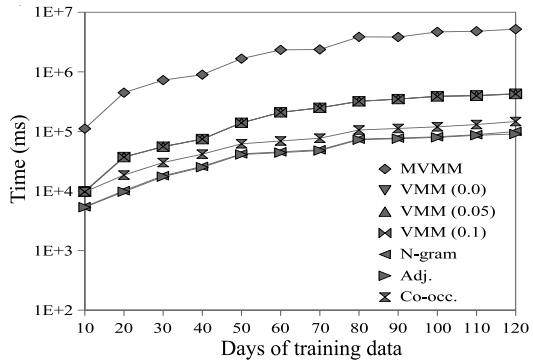


Fig. 12. Training time scales linearly with data for all methods.

due to similar ϵ settings. The VMM models consume more time compared to pair-wise and N-gram methods because the former requires time to construct the PST trees.

However, despite the above differences, the most important conclusion made from Figure 12 is that all methods have a linear time complexity w.r.t. data size. Furthermore, as mentioned before, we can always train the MVMM model in parallel. Note that all methods have comparable online prediction time complexity of $O(D)$, where D is the maximum length of training contexts, typically around 5, making comparison unnecessary.

H. User Evaluations

We conduct user evaluation tests on our query recommendation system for 4 methods: *Adj.*, *Co-occ.*, N-gram, and MVMM following the procedure outlined below.

Step 1: We randomly selected 2,000 query sequences from the test data, 500 for each context length of 1, 2, 3, and 4. We mixed all four types of query sequences, instead of conduct the real user evaluations separately on each of them, simply because the conclusions are rather similar to the previous data centric evaluations based on various context lengths. We then applied the four methods to each test query sequence to predict the top 5 queries. Altogether, a combined 26,193 predicted queries were returned by the four methods.

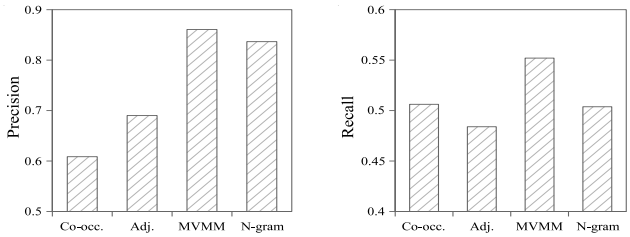
Step 2: We created a subset of predicted queries from the 26,193 predicted queries, and asked 30 volunteers to label each of them as approved or rejected. Specifically, given a predicted query at a specific position, the labelers were asked to judge whether the predicted query is appropriate in the context. For example, the following four predicted queries were approved by volunteers.

- Predicted query is "youtube", which follows immediately after the typo "youtub".
- Predicted query "Verizon" seems to be semantically related to the preceding query "GE".
- Predicted query "Hertz car rental" is more specific than the preceding query "budget car rental".
- Predicted query "New York Times" is related to the preceding query "NY Times".

Table VIII shows the distribution of user annotations over all four methods, with MVMM leading the pack.

TABLE VIII
USER LABELING DISTRIBUTION OVER FOUR METHODS.

	Co-occ.	*Adj.*	N-gram	MVMM
# predicted queries	7892	6656	5715	6086
# "approved" queries	4803	4593	4781	5238



(a) Precision (b) Recall

Fig. 13. Overall user evaluation performance.

Step 3: Only user-approved queries were collected as the user-centric ground truths. Duplicated queries were removed from the ground truth set. In the end, there are 9,489 unique ground truth queries. The standard precision and recall metric is used to evaluate the overall performance. For example, the *Co-occ.* predicted 7,892 queries, among which 4,803 queries were approved. Its precision is $4803/7892 = 60.86\%$, and its recall is $4803/9489 = 50.62\%$. Figure 13 shows the precision and recall of predicted queries for each method. Not surprisingly, although *Co-occ.* and *Adj.* could predict more queries than the other models, they have lower precision and recall. In contrast, the sequence based models have much higher precision and moderately higher recall. Overall, MVMM was the best performer achieving 86.1% precision and 55.2% recall.

Figure 14 further depicts the precision scores across the top 5 positions for each method. MVMM slightly beats N-gram, but both work much better than the *Adj.* and *Co-occ.* models. We see that the sequence-based models perform very well at

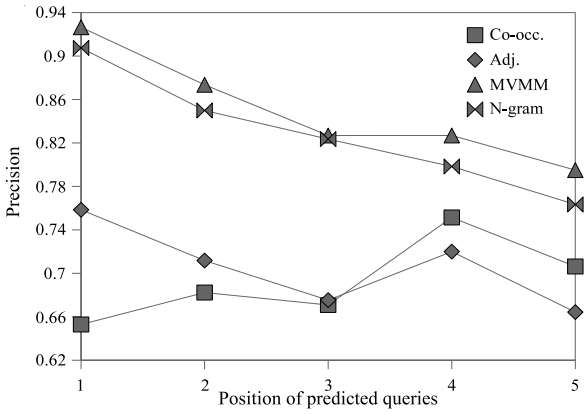


Fig. 14. Precision over top 5 positions.

the first position, which is supposed to be the most important position for query recommendation. In contrast, *Adj.* and *Co-occ.* perform quite inconsistently at the different positions.

VI. Conclusions

In this paper, we have introduced a novel approach, called sequential query prediction, for understanding users' search intent and recommending queries. We have applied sequential probabilistic models to this problem and developed a powerful mixture model called MVMM, which is based on a set of Variable Memory Markov models and is particularly suitable for the task of online query recommendation. Finally, we extensively evaluated our proposed methods on an extremely large data set using various data and user centric metrics.

From our experimental results, we can conclude the following: (1) Ordered queries within the same session are highly correlated, and should be sequentially utilized to understand the user information needs, (2) The proposed MVMM achieved the best balance among accuracy and coverage both in terms of data (objective) and user (subjective) centric evaluation metrics. A thorough time and memory complexity analysis of our MVMM was also performed, and it was found to be practical and effective for real-time deployment (constant time in D, the maximum context length), making it ideally suitable for real-time search engine query recommendation.

To the best of our knowledge, search query sequences of such massive scale have never been successfully modeled before. Our research should provide numerous useful insights towards the next generation personalized Web search engines.

As future work, we plan to further study all the different N-gram variations, as well as other more sophisticated Markov models such as HMM in the general or domain-specific search. This include modeling hidden states that represent true user intent, which could be an underlying semantic concept, especially with the help of domain knowledge. It remains to be seen whether more sophisticated models can further raise the performance bar for query recommendation (in domain-specific search). For deploying the work on a real system, the analysis on the frequency of retraining the data to adapt to new query trends would be also necessary.

References

[1] R. Baeza-Yates, C. Hurtado, and M. Mendoza. Query recommendation using query logs in search engines. In *International Workshop on Clustering Information over the Web (ClustWeb, in conjunction with EDBT)*, 2004.

[2] R. Baeza-Yates and A. Tiberi. Extracting semantic relations from query logs. In *SIGKDD*, pages 76–85, 2007.

[3] D. Beeferman and A. Berger. Agglomerative clustering of a search engine query log. In *KDD*, pages 407–416, 2000.

[4] R. Begleiter, R. El-Yaniv, and G. Yona. On prediction using variable order markov models. *Journal of Artificial Intelligence Research*, 22, 2004.

[5] 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 *SIGKDD, accepted*, 2008.

[6] P. A. Chirita, C. S. Firan, and W. Nejdl. Personalized query expansion for the web. In *SIGIR*, pages 7–14, 2007.

[7] H. Cui, J.-R. Wen, J.-Y. Nie, and W.-Y. Ma. Probabilistic query expansion using query logs. In *WWW*, pages 325–332, 2002.

[8] F. J. Damerau. Markov models and linguistic theory. *Mouton*, The Hague, 1971.

[9] L. Fitzpatrick and M. Dent. Automatic feedback using past queries: social searching? In *SIGIR*, pages 306–313, 1997.

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

[11] S. Han, A. Goker, and D. He. Web user search pattern analysis for modeling query topic changes. In *Proceedings of the user modeling for context-aware applications, a workshop of the 8th international conference on user modeling*, 2001.

[12] D. He and D. J. H. Ayse Goker. Combining evidence for automatic web session identification. *Information Processing and Management*, 38:727C742, 2002.

[13] 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 the American Society for Information Science and Technology*, 54(7):638–649, 2003.

[14] B. J. Jansen, A. Spink, C. Blakely, and S. Koshman. Defining a session on web search engines. *Journal of The American Society for Information Science and Technology*, 58(6):862C871, 2007.

[15] B. J. Jansen, A. Spink, and T. Saracevic. Real life, real users, and real needs: a study and analysis of user queries on the web. *Information Processing and Management*, 36, 2000.

[16] K. Jarvelin and J. Keklinen. Ir evaluation methods for retrieving highly relevant documents. In *SIGIR*, pages 41–48, 2000.

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

[18] S. M. Katz. Estimation of probabilities from sparse data for the language model component of a speech recognizer. In *ASSP-35*, pages 400–401, 1987.

[19] R. Kraft and J. Zien. Mining anchor text for query refinement. In *WWW*, pages 666–674, 2004.

[20] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, pages 282–289, 2001.

[21] S. Liu, F. Liu, C. Yu, and W. Meng. An effective approach to document retrieval via utilizing wordnet and recognizing phrases. In *SIGIR*, pages 266–272, 2004.

[22] M. Magennis and C. J. van Rijsbergen. The potential and actual effectiveness of interactive query expansion. In *SIGIR*, pages 324–332, 1997.

[23] A. Mccallum, D. Freitag, and F. Pereira. Maximum entropy markov models for information extraction and segmentation. In *ICML*, pages 591–598, 2000.

[24] S. Ozmutlu. Automatic new topic identification using multiple linear regression. *Information Processing and Management*, 42:934C950, 2006.

[25] L. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of IEEE*, 77(3), 1989.

[26] S. Y. Rieh and H. I. Xie. Patterns and sequences of multiple query reformulations in web searching: A preliminary study. In *Proceedings of the 64th ASIST Annual Meeting*, volume 38, pages 246 – 255.

[27] J. Rocchio. Relevance feedback information retrieval. In *The Smart Retrieval System-Experiments in Automatic Document Processing*, pages 312–323, 1971.

[28] D. Ron, Y. Singer, and N. Tishby. Learning probabilistic automata with variable memory length. In *COLT*, pages 35–46, 1994.

[29] M. Sahami and T. D. Heilman. A web-based kernel function for measuring the similarity of short text snippets. In *WWW*, pages 377–386, 2006.

[30] H. Schutze and Y. Singer. Part-of-speech tagging using a variable memory markov model. In *ACL*, pages 181–187, 1994.

[31] X. Shen, B. Tan, and C. Zhai. Context-sensitive information retrieval using implicit feedback. In *SIGIR*, pages 43–50, 2005.

[32] A. Spink and B. J. Jansen. Web search: Public searching of the web. *New York: Kluwer*, 2004.

[33] Z. Su, Q. Yang, Y. Lu, and H. Zhang. Whatnext: A prediction system for web requests using n-gram sequence models. In *WISE*, volume 1, pages 214–221, 2000.

[34] K. Sugiyama, K. Hatano, and M. Yoshikawa. Adaptive web search based on user profile constructed without any effort from users. In *WWW*, pages 675–684, 2004.

[35] J. Teevan, E. Adar, R. Jones, and M. A. S. Potts. Information re-retrieval: repeat queries in yahoo's logs. In *SIGIR*, volume 38, pages 151–158.

[36] E. Terra and C. L. Clarke. Scoring missing terms in information retrieval tasks. In *CIKM*, pages 50–58, 2004.

[37] J.-R. Wen, J.-Y. Nie, and H.-H. Zhang. Clustering user queries of a search engine. In *WWW*, pages 162–168, 2001.

[38] R. W. White, M. Bilenko, and S. Cucerzan. Studying the use of popular destinations to enhance web search interaction. In *SIGIR*, pages 159–166, 2007.