Incorporating Intra-query Term Dependencies in An Aspect Query Language Model

Dawei Song

Tianjin Key Laboratory of Cognitive Computing and Application, School of Computer Science and Technology, Tianjin University, Tianjin, China & Department of Computing, The Open University, United Kingdom

dawei.song2010@gmail.com

Yanjie Shi, Peng Zhang*

Tianjin Key Laboratory of Cognitive Computing and Application, School of Computer Science and Technology, Tianjin University, Tianjin, China

{sandy.y.shi, darcyzzj}@gmail.com

Qiang Huang

School of Computing, University of East Anglia, Norwich, United Kingdom

h.qiang@uea.ac.uk

Udo Kruschwitz

School of Computer Science and Electronic Engineering, University of Essex, Colchester, United Kingdom

udo@essex.ac.uk

Yuexian Hou, Bo Wang

Tianjin Key Laboratory of Cognitive Computing and Application, School of Computer Science and Technology, Tianjin University, Tianjin, China

yxhou@tju.edu.cn, bo.wang.1979@gmail.com

corresponding author: Peng Zhang (Email: darcyzzj@gmail.com)

Abstract

Query language modeling based on relevance feedback has been widely applied to improve the effectiveness of information retrieval. However, intra-query term dependencies (i.e., the dependencies between different query terms and term combinations) have not yet been sufficiently addressed in the existing approaches. This paper aims to investigate this issue within a comprehensive framework, namely the Aspect Query Language Model (AM). We propose to extend the AM with a Hidden Markov Model (HMM) structure, to incorporate the intra-query term dependencies and learn the structure of a novel Aspect Hidden Markov Model (AHMM) for query language modeling. In the proposed AHMM, the combinations of query terms are viewed as latent variables representing query aspects. They further form an Ergodic HMM, where the dependencies between latent variables (nodes) are modelled as the transitional probabilities. The segmented chunks from the feedback documents are considered as observables of the HMM. Then the AHMM structure is optimized by the HMM, which can estimate the prior of the latent variables and the probability distribution of the observed chunks. Our extensive experiments on three large scale TREC collections have shown that our method not only significantly outperforms a number of strong baselines in terms of both effectiveness and robustness, but also achieves better results than the AM and another state-of-the-art approach, namely the Latent Concept Expansion (LCE) model.

Keywords: Information Retrieval, Query Language Model, Aspect Hidden Markov Model, Intra-Query Term Dependency, Query Decomposition

1 Introduction

Language modeling (LM) has become a widely used approach in information retrieval (IR) (Ponte and Croft 1998; Miller et al. 1999; Hiemstra and Kraaij 2005; Manning et al. 2008). Query language modelling based on relevance feedback documents aims to further improve the retrieval effectiveness by computing a refined query model that better represents the user's information need (Lafferty and Zhai 2001; Lavrenko and Croft 2001; Zhai and Lafferty 2001). A prominent query language modeling approach is the Relevance Model (RM) (Lavrenko and Croft 2001). Practically, variants of RM have shown encouraging performance in ad-hoc search (Lavrenko and Croft 2001), cross-language retrieval (Laverenko et al. 2002) and topic detection and tracking (Lavrenko et al. 2002), etc.

The relevance model computes $P(w|R)$, which is interpreted as the probability of observing a word w in documents relevant to an information need (R). In practice, it is approximated by $P(w|Q)$ for a query Q, e.g., *P(defense| "star wars NASA").* Computing this probability for every term w in the vocabulary yields an estimate of the true relevance model. In RM, the query terms are considered to be random samples from R, an unknown process from which words can be sampled. The RM deals with the following question: if query terms $q_1, \ldots, q_m$ have been sampled, what is the probability of term w will be sampled next. In (Lavrenko and Croft 2001), two methods (RM1 and RM2) are developed to compute the probability. RM1 assumes that w and q_i are mutually independent given an identical distribution M. RM2 picks a distribution M_j according to $P(M_j|w)$, in which the query terms are assumed to be independent from each other, but still keep their dependencies on w. Operationally, the top ranked documents retrieved by the query Q are used to obtain these distributions M_j. Of particular importance to this article is that the query terms are sampled independently of each other. Indeed, the independence between query terms has been assumed in most other typical query language models, such as the model-based feedback approach in (Zhai and Lafferty 2001).

These models neglect the relationships between query terms in determining the query language model and therefore may lead to inappropriately high probabilities being ascribed to terms that are not aligned with the given retrieval context. In fact, there often exist dependencies between query terms (i.e., intra-query term dependencies). The combinations of related query terms (e.g., "star wars NASA" which is about the U.S. missile defense program) often carry more information than single terms individually. Recent research has shown that completely new meaning may emerge from the term combinations (Bruza et al. 2011). These query term combinations do not have to be grammatically valid phases or adjacent query terms. In addition, there also exist dependencies even between different query term combinations, e.g., "star wars NASA" and "star wars".

In response to the problem, there has been work in the incorporation of term dependencies into language modeling (Cao et al. 2005; Bai et al. 2005; Metzler and Croft 2005, 2007; Nallapati and Allan 2002), e.g., grammatical links (Gao et al. 2004), term co-occurrence and WordNet relations (Cao et al. 2005). However, most of them do not directly incorporate the arbitrary combinations of query terms. A related work is the Cross Term Retrieval model (CRTER), to model the associations among query terms in probabilistic retrieval models (Zhao et al. 2011). The influence to their neighboring text of a query term is approximated by a kernel function, which gradually decreases with the distance to the term. A Cross Term occurs when two query terms appear close to each other and their impact shape functions have an intersection. The model only computes the associations of two query terms. Another related approach to ours is the Latent Concept Expansion (LCE) method based on the Markov Random

Field (MRF) (Metzler and Croft 2007), which takes into account query term combinations in deriving the probabilities of words or multiple words, called latent concepts, given the query. However, LCE is based on the sequential dependence assumption, which assumes that dependencies exist between adjacent query terms, and the dependencies between different query term combinations are not considered. In their experiments, the available topics are split into a training set and a test set, where the training set is used for parameter estimation. This would increase the computational overhead and is less practical when the training data are not available.

Moreover, there has been a trend of decomposing a query into different combinations (subsets) of query terms, and exploiting term relationships derived from the subsets of query terms rather than traditional pairwise term co-occurrence. For example, the initial query "star wars" can be decomposed into "star", "wars", "star wars". A state of the art approach is the Aspect Query Language Model (AM), which views different subsets of a query as its different aspects and incorporates high-order term relationships in a Bayesian-like structure (Song et al. 2012). The association rule mining is used to capture the high-order term associations. It then establishes a query language model by aggregating the high-order term associations between the different query subsets and the observed terms in the relevance feedback documents, and optimizes the prior probabilities of query subsets and the documents by an automated EM learning process facilitated by an on-the-fly training data generation method. In the empirical evaluation, significant improvements over a baseline LM and the RM have been achieved. However it overlooks the intra-query term dependencies, i.e., the dependencies between different query term combinations.

Building upon and extending the AM by incorporating dependencies among query term subsets, in this paper, we propose an Aspect Hidden Markov Model

(AHMM) for query language modelling. Similarly as in the AM, we decompose a query into different combinations (subsets) of query terms and consider them as latent variables over the feedback documents or segmented chunks of the documents (as observables). The AHMM can be constructed to connect a document chunk d and a word w through the latent variables. The dependencies between the latent variables are governed by an ergodic Hidden Markov Model (HMM), where the Viterbi algorithm is applied to optimize parameters involved in the AHMM, based on an on-the-fly training data construction method. Experimental results on various large TREC collections show that our approach outperforms the RM, LCE and AM in terms of both mean average precision and robustness. Some preliminary early work has been reported as a poster paper in (Huang and Song 2008). This paper presents a substantial extension of the previous work and more comprehensive and systematic evaluation results.

2 Preliminaries: The Aspect Query Language Model (AM)

This section gives a brief description of the AM, where the subsets of query terms are viewed as an initialization of a series of corresponding latent variables (representing query aspects) over a number of top-ranked documents from the initial retrieval. We treat the query aspects as latent variables, as they (and their optimal weighting) can only be derived through the top ranked documents we observe (Song et al. 2012).

The AM is a graphical model, with a Bayesian Network like structure, as shown in Figure 1. S_j is a latent variable in the set $\mathbf{S}$ ($\mathbf{S} = \{S_1, \cdots, S_N\}$), and w is a word whose occurrence probability in the expanded query model (formally denoted as θ_Q) to be estimated. The latent variable S_j is generated from the

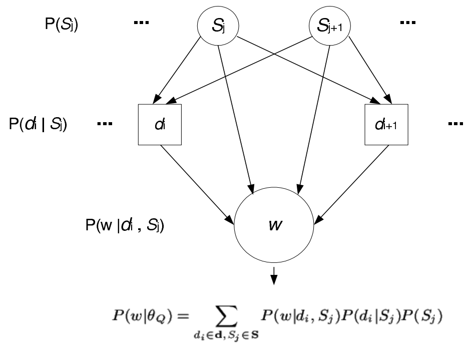

P(S_j)
...
S_j
S_j+1
...
P(d_i | S_j)
...
d_i
d_i+1
...
P(w | d_i , S_j)
w
$P(w|\theta_Q) = \sum_{d_i \in \mathbf{d}, S_j \in \mathbf{S}} P(w|d_i, S_j)P(d_i|S_j)P(S_j)$

Figure 1: Structure of Aspect Model

original query $Q = \{q_1, \cdots, q_M\}$, where each S_j is in general defined as a query term or a combination of query terms. The size of $\mathbf{S}$ is $(2^M - 1)$ if we use all the combinations of query terms. For instance, given $Q = \{q_1, q_2\}$, the set of latent variables can be transformed into $\mathbf{S} = \{\{q_1\}, \{q_2\}, \{q_1, q_2\}\}$. In practice, we can only use the combinations of up to k query terms as latent variables in order to reduce the computational complexity. Indeed, our experimental results indicate little difference in effectiveness between the use of combinations of 2-3 terms and the combinations of more terms as states.

The relationship between a word w and a latent variable S_j is derived through the relevance feedback documents. In practice, such as in the Web search environment, the number of top ranked documents actually observed by users is often small (Joachims et al. 2005), e.g., 5-15 (1 page), which will lead to the data sparsity problem given the large number of latent variables for longer queries. Even for a relevant document, it is not necessarily true that every part within the document is relevant. Thus, in practice, segmented document chunks, instead of whole documents, can be used to connect S_j and w in order to expand the observation space to improve the quality of parameter optimization. Let $\mathbf{d}$ denote the collection of relevance feedback documents or document chunks segmented, e.g., by a sliding window.

Based on the relations among S_j, d_i and w, the expanded query model $P(w|\theta_Q)$ can be estimated by the following equation:

$$P(w|\theta_Q) = \sum_{d_i \in \mathbf{d}, S_j \in \mathbf{S}} P(w|d_i, S_j)P(d_i|S_j)P(S_j) \tag{1}$$

where $P(S_j)$ is the prior distribution of latent variables, $P(d_i|S_j)$ is the probability of an observed document chunk d_i given a latent variable S_j, and $P(w|d_i, S_j)$ is the probability of a word w in a chunk d_i given S_j.

It was argued that the conditional probability $P(w|d_i, S_j)$ is not easy to estimate because of the data sparsity (Song et al. 2012). In order to estimate $P(w|d_i, S_j)$, one can assume that d_i and S_j are independent, where d_i denotes a chunk and S_j denotes a query state. Based on this assumption, we can get $P(w|d_i, S_j) \propto P(w|d_i)P(w|S_j)$. By further assuming that $P(w|d_i)$ is uniform, Equation 1 is simplified by replacing $P(w|d_i, S_j)$ with $P(w|S_j)$, leading to:

$$P(w|\theta_Q) = \sum_{d_i \in \mathbf{d}, S_j \in \mathbf{S}} P(w|S_j)P(d_i|S_j)P(S_j) \tag{2}$$

The Expectation Maximization (EM) algorithm is used to fit the parameters of Equation 2. An on-the-fly training data constraction method is develope-d to automatically label the document chunks in the feedback document set with query term(s). In the *E-step*, the posterior probability of the hidden variable $P(S|d, w))$ is computed. In the *M-step*, the parameters $P(S)$, $P(d|S)$ and $P(w|S)$ are maximized. More technical details can be found in (Song et al. 2012).

As described above, the AM provides a comprehensive framework to integrate the high-order term associations between query term subsets and words observed in the feedback documents. Despite the success of the AM, it does not take into account the intra-query term dependencies (i.e., dependencies between the latent variables in AM). As argued in the Introduction, we believe that incorporating such dependencies into the AM will lead to further performance improvement. To address this issue, in the next section, we propose a novel Aspect Hidden Markov Model (AHMM) that extends the AM for query expansion.

3 Aspect Hidden Markov Model

As emphasised in the previous sections, in order to better estimate the parameters in Eq. 1 to derive the expanded query model $P(w|\theta_Q)$, it is important to take into account the dependencies between the latent variables, which govern the distribution of prior probabilities of variables S_j and the observation probability of the document chunk d_i given S_j. In this section, we present our AHMM approach that extends the original AM and allows a natural incorporation of the dependencies between the latent variables through a Hidden Markov Model (HMM) mechanism. There has been evidence that the source of natural language text can be modelled as an "ergodic" Markov process, meaning the Markov chain is aperiodic (i.e., words can be separated by any number of intermediate words) and irreducible (i.e., we can always get from one word to another by continuing to produce text) (Hoenkamp et al. 2009).

As shown in Figure 2, based on the dependencies among S_j, d_j and w, we extend the AM by adding links between S_j and S_{j+1}.

The HMM is a finite set of states ($\mathbf{S} = \{S_1, \cdots, S_M\}$), each of which is associated with a probability distribution ($\pi = \{P(S_1), \cdots, P(S_M)\}$). Transitions among the states ($A = \{S_{j',j}\}$, $S_j, S_{j'} \in \mathbf{S}$) are governed by a set of probabilities called transition probabilities. For a particular state, an observation d_i can be generated according to the associated probability distribution denoted as $B = \{P(d_i|S_j)\}$. Figure 3 shows an example structure of a three-state ergodic

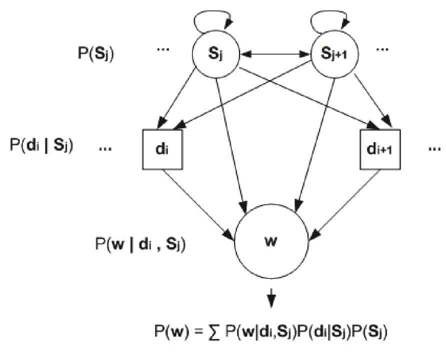


Figure 2: Structure of Aspect Hidden Markov Model

Hidden Markov Model.

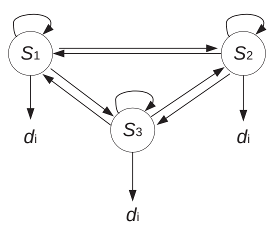


Figure 3: An example of three-state Ergodic Hidden Markov Model

Taking the dependencies between the latent variables into consideration, we need to design a parameter estimation framework based on effective optimization mechanisms in HMM. The application of the HMM can not only estimate the prior distribution of each S_j, but also integrate the dependence between any two latent variables (as states) and their underlying observables (document chunks) through a state transition matrix. When applied to our AHMM, the HMM can deal with the following three aspects:

1. Given the observation chunks $\boldsymbol{d} = \{d_1, \cdots, d_T\}$ and a model $\Lambda = (A, B, \pi)$, to compute $P(\boldsymbol{d}|\Lambda)$.
2. Given the observation chunks $\boldsymbol{d} = \{d_1, \cdots, d_T\}$ and a model $\Lambda = (A, B, \pi)$, to choose a corresponding state sequence $(S_1, \cdots, S_T)$ that is optimal (i.e.,

best "explains" the observations).

3. To adjust the model parameters $\Lambda = (A, B, \pi)$ to maximize $P(\boldsymbol{d}|\Lambda)$.

In order to run the HMM, the widely used Viterbi algorithm can be applied to search the optimal state. The derivation of the Viterbi algorithm is presented as follows:

To find the single best state sequence, denoting $\mathbf{S} = \{S_1 S_2 \cdots S_T\}$, for a given observation sequence $\mathbf{d} = \{d_1 d_2 \cdots d_T\}$, we define a quantity $\delta_t(j)$. It is the highest probability along a single path, at time t, which accounts for the first t observations and ends at the state j.

$$\delta_t(j) = \max_{S_1, S_2, \cdots, S_{t-1}} P(S_1 S_2 \cdots S_t = j, d_1 d_2 \cdots d_t|\Lambda) \tag{3}$$

By induction we can obtain:

$$\delta_{t+1}(j) = [\max_j \delta_t(j) a_{jj'}] \cdot b_j(d_{t+1}) \tag{4}$$

where $a_{jj'}$ is an element of transition matrix A, and $b_j(d_t) = P(d_t|S_j)$. To actually retrieve the state sequence, we need to keep tracking the arrangement that maximizes Eq. 4, for each t and j. A detailed description of the Viterbi algorithm can be found in (Rabiner 1989).

In this paper, the application of HMM does not have to strictly obey the regular rule of HMM, because the query term subsets are used to initialize the states (S_j). This means these states are not strictly "hidden" in a strict sense. In the process of learning the model, we utilize the Baum-Welch algorithm to optimize the state distribution and transition matrix, and use the Viterbi algorithm to search the optimal path and update the probability distribution of each chunk in different states. The update of $P(d_t|S_j)$ is based on:

$$P(d_i|S_j) = \frac{\sum_{t=1,d_t=d_i}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)} \tag{5}$$

where $\gamma_t(j) = P(S_j|d_t, \Lambda)$. The value of $\gamma_t(j)$ is estimated by using an iterative computation (detailed later in Figure 4).

In summary, the HMM would seem to provide a well-established mechanism to effectively utilize the dependencies among the states to estimate the parameters listed in Eq. 1. The detailed description of how the HMM is applied to our AHMM is detailed in the next section.

4 Parameter Learning in AHMM

In order to compute the parameters in Eq. 1 under the AHMM framework, we have designed an operational framework. The details of the framework are shown in Figure 4.

4.1 Data Pre-processing (Step 1)

This step follows a similar approach as in the original AM. Step 1.1 takes the query terms as states, which has been discussed in Section 3. Step 1.2 selects true relevance feedback documents and pseudo-relevant documents. In order to increase the observation space and estimate the relevance of the different parts of a document with respect to the query (as discussed in Section 3), in Step 1.3, an overlapped sliding window is used to segment each document into chunks. This is based on the idea that the chunks can keep a more accurate meaning than a document which is often a mixture of various meanings. It is also motivated by the successful application of the HMM in speech recognition through segmenting the speech signals for short-time fourier transformation. We set the overlapping length to be 4/5 of the window size, a setting often used

1. **Data Pre-processing**

   1.1 Generate states by combining query terms.

   1.2 Select N top-ranked relevant or pseudo-relevant documents according to the initial retrieval results.

   1.3 Segment the selected document into chunks with an overlapped sliding window, whose overlapping length is 4/5 of the sliding window size.

   1.4 Retain the chunks containing any query terms and discard the rest.

   1.5 Assign those chunks containing query terms into various clusters, labelled by the states that share one or more query terms with the chunks.

2. **Optimize Aspect Hidden Markov Model**

   2.1 Set the initial values of the model parameters.

   $P(S_j) = 1/M, \quad (M = |\mathbf{S}|, \ S_j \in \mathbf{S})$

   $P(S_{j'}|S_j) = 1/M, \quad (M = |\mathbf{S}|), \ S_j, S_{j'} \in \mathbf{S})$

   $P(d_i|S_j) = \frac{\sum_{t=1,d_t=d_i}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}$

   where $\gamma_t(j) = P(S_j|d_t, \Lambda)$, and $P(S_j|d_t, \Lambda)$ can be approximately derived by the pseudo code as follows:

   ```
   $d_{i,k} = \{w_1, \cdots, w_k\}, \quad d_i = d_{i,K} \ (K = |d_i|)$
   $P(S_j|d_{i,0}) = P(S_j)$
   for k = 1 : K,
       $P(S_j|d_{i,k}) = \frac{1}{k+1} \frac{P(w_k|S_j)P(S_j|d_{i,k-1})}{\sum_{S_t} P(w_k|S_t)P(S_t|d_{i,k-1})} +$
                   $\frac{k}{k+1} P(S_j|d_{i,k-1})$
   end
   $P(S_j|d_i) = P(S_j|d_{i,K})$
   ```

   The computation of $P(w_k|S_j)$ is detailed in Section 4.2.

   2.2 Apply Viterbi algorithm to searching the optimal state sequences.

   2.3 Collect the labelled chunks of each "state" and update the occurrence probability of the observed term w_k, namely $P(w_k|S_j)$, then $P(d_i|S_j)$.

   2.4 Optimize the model iteratively by repeating Step2.1 $\sim$ Step2.3.

3. **Derive the language model**

   Compute the final probability of $P_{AHMM}(w|\theta_Q)$ using Eq. 1.

Figure 4: Outline of framework

on speech recognition, for optimizing the AHMM. An additional advantage of using overlapped window is to increase the number of the observed chunks and thus reduce the risk of over-fitting when optimizing the AHMM.

Step 1.4 can be seen as a coarse data refinement process. It is reasonable to keep only those chunks containing at least one query terms because we believe they contain more useful information than the chunks without containing any query term. Step 1.5 can be considered as an on-the-fly training data construction process, which essentially clusters the selected chunks into classes labelled by different states. For example, the state corresponding to $\{q_1, q_2\}$ clusters the chunks containing the query term q_1 or q_2 or both, and the state corresponding to q_1 only clusters the chunks containing the query term q_1. According to these clustered chunks, we can compute the initial word probability given a state S_j, denoted as $P(w|S_j)$. These initial computations are then used to optimize the AHMM in Step 2.

4.2 Model Optimization (Steps 2 & 3)

According to the description of the AHMM in Section 3, we initialize the model parameters by setting the state distribution $P(S_j)$ and the state transition probability $P(S_{j'}|S_j)$ to be the chance probability $\frac{1}{M}$. Here M is the number of states in the AHMM. A recursive method is then used to compute $P(d_i|S_j)$, as similarly used in (Blei and Moreno 2001). In this recursive equation, $d_{i,k}$ denotes the first k words in a chunk d_i and $d_{i,K} = d_i$, where K is the window size. $P(S_j|d_{i,0})$ as an initial value is set to be $P(S_j)$. $P(d_i)$ is computed as the occurrence frequency of chunk d_i over the collection of chunks generated by segmenting the feedback documents.

To initialize the $P(w_k|S_j)$, we use the Apriori algorithm for association rule (AR) mining, as in the original AM. We consider the chunks as transactions and terms as items. The dependency between Q_j and w_k corresponds to the associated rule $Q_j \Rightarrow w_k$. Association rule mining has had a proven track record in discovering useful associations from transaction data (Srivastava et al. 2000; Luo and Bridges 2000; Creighton and Hanash 2003). Therefore we consider it an effective way to compute $P(w_k|S_j)$. Setting the initial $P(w_k|S_j)$ to be the derived values of using AR is more sensible than setting them to be uniform values. It can accelerate the convergence and also in some sense prevent the model from converging to a local maximal point. There are two measurements for estimating the degree of dependency: support and confidence.

$$supp(X \Rightarrow Y) = supp(X \cup Y) = \frac{C_{XY}}{N} \tag{6}$$

$$conf(X \Rightarrow Y) = \frac{supp(X \cup Y)}{supp(X)} \tag{7}$$

C_{XY} is the number of transactions which contain all the items in X and Y, and N is the total number of transactions. *Support*, in Eq. 6, is defined as the fraction of transactions in the database which contain all items in a specific relationship, such as $X \Rightarrow Y$ (Agrawal et al. 1993). *Confidence*, in Eq. 7, is an estimate of the conditional probability $P(E_Y|E_X)$, where E_X (E_Y) is the occurrence of X (Y) in a transaction (Hipp et al. 2000). In this paper, we set S_j to be X and w_k to be Y, and the normalized confidence value is used as the probability of word w_k given S_j. The probability $P(w_k|S_j)$ is then applied to the recursive equation to compute $P(S_j|d_i)$.

In Step 2.2, we apply the Viterbi algorithm to searching the optimal state sequence, then we update the HMM iteratively by re-computing the model parameters $\Lambda = \{\pi, A, B\}$. Finally, the two probability parameters $P(S_j)$ and $P(d_i|S_j)$ in Eq. 1 are updated according to the updated HMM. In order to compute the third probability parameter $P(w|d_i, S_j)$, we use the following equation:

$$P(w|d_i, S_j) = P(w|S_j) \cdot \frac{\#w}{\sum_k \#w_k} \tag{8}$$

$\#w$ is the frequency of word w occurring in chunk d_i and the denominator is the total number of words in d_i.

5 Model Integration

As previously described, the HMM is used to learn the structure of the AHMM for query expansion. However, it is inevitable that the probability estimations may not be accurate due to data sparsity (only a few feedback documents are used) and the possible query drift when expanding the original query.

To alleviate this problem, it is disirable to regularize the model estimation with the original query model. Similar consideration has also been taken into account in state-of-the-art approaches, such as RM, AM and LCE, which smooth the derived query model with the original query model.

In this paper, the original query model is defined as:

$$P(q_k|Q) = \frac{\#q_k \cdot IDF(q_k)}{\sum_{j\in\{1\cdots|Q|\}} \#q_j \cdot IDF(q_j)} \tag{9}$$

where $\#q_k$ is the frequency of query term q_k in Q and $IDF(q_k)$ is the inverse document frequency (IDF) of q_k.

To integrate the original query model into the expanded query model derived by AHMM, a traditional approach is the linear interpolation of the two. We refer it as **AHMM-I**. The main limitation of this method is that it involves manual adjustment of the interpolation coefficient to generate optimal retrieval performance. In this paper, we propose an automatic method, denoted as **AHMM-II**, to integrate the original query model directly into the AHMM. Here, we focus on the AHMM-II and compare it with the baselines. More discussion on using AHMM-I will be given in Section 7.4.

Specially, AHMM-II considers the original query model as a state S_Q in the AHMM. Although there is already a state S_j in the AHMM structure which consists of all query terms, the model corresponding to S_j is an expanded model containing additional words rather than only the query terms. To strengthen the effects of original query terms, we propose adding an additional state S_Q which links to the original query terms only. Accordingly Eq. 1 is changed to:

$$P_{AHMM-II}(w|\theta_Q) = \sum_{d_i \in \mathbf{d}} \sum_{S_j \in \{\mathbf{S}, S_Q\}} P(w|d_i, S_j)P(d_i|S_j)P(S_j) \qquad (10)$$

where the state S_Q contains only the query terms and the probability distribution of each query term is computed according to Eq. 9. We can then run the AHMM directly to estimate the combined query language model.

6 Data and Experimental Set Up

6.1 Data

We evaluate our methods using TREC topics 151–200 on AP88–90 (TREC disks 1, 2 and 3); topics 601–700 on ROBUST (TREC disk 4 and 5, excluding the *Congression Record*); and topics 501–550 on WT10G (Disk WT10G), respectively. Only the titles of the TREC topics are used as queries. This is to simulate typical Web search scenarios where users tend to submit short queries typically around 2-3 words. Table 1 provides a summary of the TREC data used. These data sets are selected because they have varied content and document properties. The AP88–90, and ROBUST collections are generally homogeneous news articles while WT10G documents are Web pages with a variety of subjects and styles. In our experiments, a stopword list and Porter stemmer are applied to all data collections. Note that the same experimental setting is also used in (Metzler and Croft 2007). This allows a direct comparison between our method and the LCE model (Metzler and Croft 2007).

Table 1: Overview of TREC collections and topics.

Collection	Description	Size	# of Docs	Topics
AP88-90	Associated Press (88–90)	730MB	242,918	151–200
ROBUST	Robust 2004	1.9GB	528,155	601–700
WT10G	TREC Web collection	10.9GB	1,692,096	501–550

6.2 Experimental Set Up

In order to fully test the performance of the proposed AHMM approach, we apply it to two scenarios: **pseudo-relevance feedback** and simulated **true relevance feedback**. The pseudo-relevance feedback is a way of automatic query expansion by assuming top ranked documents after initial retrieval to be relevant, and the true relevance feedback only selects those top-ranked documents that are truly relevant (simulated according to the TREC ground truth data). The latter is very useful in simulating real user-based retrieval and also shows the potential upper bound of our approach.

Three baselines are used for comparison including a language model based on Kullback-Leibler (KL) divergence and two variants of Relevance Models (RM1 & RM2). For pseudo-relevance feedback, all the methods are tested with various numbers (N = 10, 30, 50) of top-ranked documents from the initial retrieval results. We not only compare our methods with the LCE model based on its results reported in (Metzler and Croft 2007), but also compare our methods with the original Aspect Model. For true relevance feedback, we only use fewer number of feedback documents (N = 5, 10, 15). This is based on the observation that real users often prefer selecting a small number of relevant documents as feedback (Iwayama 2000; Al-Maskari et al. 2008).

All the experiments are carried out using Lemur toolkit 4.0 (Lem). The initial retrieval is performed by using the KL-divergence based language model (KL). The new query language models derived from different query language modeling methods (RM1, RM2, AM and our AHMM) are used to perform the second round of retrieval using KL. For each of the RM1, RM2 and AM, the new query model is generated by linearly combining the expanded model and the original model. For AHMM, its two variants, i.e., AHMM-I (based on linear combination) and AHMM-II (based on integration of the original query into the AHMM) are used. For each model using linear combination, we have tested a range of linear combination coefficients and only the best performing result is reported.

6.3 Performance Measurements

Following the TREC standard, we retrieve 1,000 documents for each query. Our primary evaluation metric is mean average precision (MAP). The Wilcoxon singed rank test is used to measure the statistical significance. We also analyze the robustness of our methods against the baselines, using the robustness measure introduced in (Metzler and Croft 2007). It is defined as the number of queries whose effectiveness are improved/impaired after applying the query language modeling tested, in comparison with the baseline. We let #Pos. denote the number queries whose effectiveness are improved, and let #Neg. denote the number of impaired queries. The bigger the value of #Pos.-#Neg. is, the more robust the model is. A highly robust expansion technique will significantly improve many queries and only minimally hurt a few.

7 Experimental Results

In this section we present our experimental results and compare with the baselines and other state-of-art methods, such as LCE and AM. The details of experimental analysis are described from four perspectives:

- Evaluation in pseudo-relevance feedback.

Table 2: Comparison of AHMM-II, KL, RM1, RM2 and AM, in Pseudo-relevance Feedback

	# **docs**(N)	**KL**	**RM1**	**RM2**	**AM**	**AHMM-II**
AP88-90	10	0.2077	0.2578	0.2639	0.2689	$0.2795^{\alpha,\beta,\gamma}$
	30	0.2077	0.2603	0.2676	0.2706	$0.2814^{\alpha,\beta,\gamma}$
	50	0.2077	0.2625	0.2706	0.2694	0.2736^{α}
ROBUST	10	0.2920	0.3129	0.3143	0.3510	$0.3613^{\alpha,\beta,\gamma}$
	30	0.2920	0.3250	0.3271	0.3427	$0.3561^{\alpha,\beta,\gamma}$
	50	0.2920	0.3238	0.3289	0.3326	$0.3424^{\alpha,\beta}$
WT10G	10	0.2032	0.2131	0.2134	0.2210	$0.2305^{\alpha,\beta,\gamma}$
	30	0.2032	0.2077	0.2079	0.2175	$0.2279^{\alpha,\beta,\gamma}$
	50	0.2032	0.2082	0.2088	0.2093	0.2136^{α}

- Evaluation in true relevance feedback.
- Impact of the window size in document segmentation.
- Acquisition of optimal performance.

7.1 Evaluation in Pseudo-Relevance Feedback

The evaluation results in the context of pseudo-relevance feedback are summarized in Table 2, where we compare the performances of the proposed AHMM with the KL, the two relevance models (RM1 and RM2) and the Aspect Model (AM). We first compare our methods with the baselines according to the MAP values obtained by using three different numbers (i.e., 10, 30, 50) of feedback documents.

As shown in Table 2, our method (AHMM-II) significantly outperforms KL, RM1 and RM2 in all cases. The superscripts α, β and γ indicate statistically significant improvements (at the level of 0.05 corresponding to Wilcoxon-test) over KL, RM1, and RM2, respectively. In Table 3, we list the improvements of AHMM-II over other four methods (KL, RM1, RM2 and AM). When selecting a smaller number of documents (N=10), the AHMM-II shows significant improvements over KL, RM1, and RM2 by 34.6%, 8.4%, and 5.9% on AP88-90;

Table 3: Improvements of AHMM-II over KL, RM1, RM2 and AM, in Pseudo-Relevance Feedback

	# docs	**Impr. using AHMM-II over** (%)			
	($\boldsymbol{N}$)	**KL**	**RM1**	**RM2**	**AM**
AP88-90	10	+**34.6***	+**8.4***	+**5.9***	+3.9
	30	+35.5*	+8.1*	+5.2*	+4.0
	50	+31.7*	+4.2	+1.1	+1.6
ROBUST	10	+**23.9***	+**15.5***	+**14.9***	+2.9
	30	+21.9*	+9.6*	+8.9*	+3.9
	50	+17.3*	+5.7*	+4.1	+2.9
WT10G	10	+**13.4***	+**8.2***	+**8.1***	+4.3
	30	+12.2*	+9.7*	+9.6*	+4.8
	50	+5.1*	+2.6	+2.3	+2.1

$*$ The improvement is statistically significant at the level of 0.05 according to the Wilcoxon signed rank test

23.9%, 15.5%, and 14.9% on ROBUST; and 13.4%, 8.2%, and 8.1% on WT10G, respectively. This indicates the effectiveness and efficiency of our approach, where selecting fewer documents can reduce the computational time and improve the retrieval effectiveness.

In comparison with the original AM (Song et al. 2012), as shown in Table 2, AHMM-II demonstrates a better performance. This verifies our hypothesis that incorporating intra-query dependencies can improve retrieval performance.

Table 4: MAP for LCE and AHMM-II

	LCE	**AHMM-II**	**Impr.** over LCE (%)
AP88-90	0.2692	0.2814	+4.5
ROBUST	0.3601	0.3613	+0.3
WT10G	0.2269	0.2305	+1.6

In addition, we compare the MAPs of AHMM-II and LCE (Metzler and Croft 2007) in Table 4. Due to the lack of the implementation details, we do not implement the actual LCE model. Instead, we use the results reported in (Metzler and Croft 2007). Thus, we are unable to do the statistical significance test. In comparison with the state-of-the-art LCE model, AHMM-II still shows better performance on three data sets. Specifically, AHMM-II achieves 4.5% an improvement over LCE on AP88–90, and marginal improvement on the largest data set WT10G. The results are remarkable, as our method does not require extra training data for parameter optimization while the LCE does.

We now analyze the robustness of different methods. The robustness measure used here is described as the number of queries whose average precisions are improved/hurted as the result of applying the proposed methods over the baselines. A highly robust expansion technique is expected to improve the average precisions for a majority of queries.

Table 5: Comparison of Robustness of AHMM-II vs. KL, RM1, RM2 and AM

Methods	**#Pos.**	**#Neg.**	**#Eq.**	**# Pos.-Neg.**
Test Q151–200 on AP88–90				
AHMM-II vs. KL	35	15	0	+20
AHMM-II vs. RM1	30	20	0	+10
AHMM-II vs. RM2	27	23	0	+4
AHMM-II vs. AM	28	20	2	+8
Test Q601–700 on ROBUST				
AHMM-II vs. KL	65	34	0	+29
AHMM-II vs. RM1	65	33	1	+32
AHMM-II vs. RM2	64	34	1	+30
AHMM-II vs. AM	59	40	0	+19
Test Q501–550 on WT10G				
AHMM-II vs. KL	31	18	1	+13
AHMM-II vs. RM1	29	19	2	+10
AHMM-II vs. RM2	28	22	0	+6
AHMM-II vs. AM	34	15	1	+19

Table 5 provides an analysis of the robustness of AHMM-II *vs.* KL, RM1, R-M2 and AM. *Pos.* denotes the number queries whose effectiveness are improved, and *Neg.* denotes the number of impaired queries. *Eq.* means the same effectiveness. Our method exhibits a strong level of robustness on each data set. For WT10G, our method improves 31, 29, 28 and 34 queries over KL, RM1, RM2 and AM respectively, and only impairs 18, 19, 22 and 15 queries respectively. For ROBUST, AHMM-II respectively improves 65, 65, 64 and 59 queries, but only impairs 34, 33, 34 and 40 in the comparison with KL, RM1, RM2 and AM. For AP88–90, compared with KL, RM1, RM2 and AM, AHMM-II improves 35, 30 , 27 and 28 queries, and impairs 15, 20 , 23 and 20 queries. Note that there are no judgements for topics 151–200 in collection AP90, which can be seen as an interference to the retrieval performance. Our method shows a sound ability,

Table 6: Comparison of AHMM-II, KL, RM1, RM2 and AM, in True Relevance Feedback

	# docs(N)	**KL**	**RM1**	**RM2**	**AM**	**AHMM-II**
AP88-90	5	0.2077	0.3194	0.3184	0.3543	$0.3591^{\alpha,\beta,\gamma}$
	10	0.2077	0.3106	0.3354	0.3816	$0.4001^{\alpha,\beta,\gamma}$
	15	0.2077	0.3218	0.3427	0.3886	$0.4034^{\alpha,\beta,\gamma}$
ROBUST	5	0.2920	0.3810	0.4095	0.4263	$0.4554^{\alpha,\beta,\gamma}$
	10	0.2920	0.4017	0.4305	0.4663	$0.4835^{\alpha,\beta,\gamma}$
	15	0.2920	0.4126	0.4432	0.4806	$0.5014^{\alpha,\beta,\gamma}$
WT10G	5	0.2032	0.2107	0.2232	0.3426	$0.3274^{\alpha,\beta,\gamma}$
	10	0.2032	0.2382	0.2629	0.3691	$0.3788^{\alpha,\beta,\gamma}$
	15	0.2032	0.2571	0.2993	0.3888	$0.3946^{\alpha,\beta,\gamma}$

and obtains a better performance in robustness than the comparative methods.

7.2 Evaluation in True Relevance Feedback

In this section, we compare our approach with KL, RM and AM in the context of true relevance feedback, i.e., only using the truly relevant top-ranked documents as the feedback documents.

Table 6 shows the MAPs of different models. In Table 7, we also list the improvements of AHMM-II over other four methods (KL, RM1, RM2 and AM). Since in true relevance feedback, users often only want to provide a small number of feedback documents relevant to their information need, we set the number of feedback documents as 5, 10, 15, respectively, in this set of experiments. Our approach (AHMM-II), again, achieves significantly better results over the baseline language model, and the two Relevance Models. We can see that AHMM-II also shows better performance than AM. When we are using 5 relevant documents, for AP88–90 and ROBUST, the improvements over the baselines are not as large as the improvements when using 10 and 15 feedback documents. However, for the Web data set (WT10G), the performance seems less dependent on the number of feedback documents used. This appears to be an interesting phenomenon, and a detailed analysis is given as follows.

Table 7: Improvements of AHMM-II over KL, RM1, RM2 and AM, in True Relevance Feedback

	# docs	**Impr. using AHMM-II over** (%)			
	($\boldsymbol{N}$)	**KL**	**RM1**	**RM2**	**AM**
AP88-90	5	+72.9*	+19.9*	+12.8*	+1.4
	10	+92.6*	+28.8*	+19.3*	+4.9
	15	+94.2*	+25.3*	+17.7*	+3.8
ROBUST	5	+56.0*	+19.5*	+11.2*	+6.8*
	10	+65.6*	+20.6*	+12.3*	+3.7
	15	+71.7*	+21.5*	+13.1*	+4.7
WT10G	5	+61.1*	+55.4*	+46.7*	
	10	+86.4*	+59.0*	+44.1*	+2.6
	15	+94.2*	+53.5*	+31.8*	+1.5

$*$ The improvement is statistically significant at the level of 0.05 according to the Wilcoxon signed rank test

In the AHMM-II, by setting the subset of query Q_j as a state in the HMM, estimating the dependencies between Q_j and the observed words and the dependencies between the document chunks and Q_j, we can easily build a good query language model when an appropriate number of relevance feedback documents are selected. However, when a very small number of relevant documents are used, the number of chunks generated may be too small to optimize our model. In pre-processing of the feedback documents, we only select the chunks containing at least one query terms. This further reduces the number of samples for optimizing our model. In addition, the reduction of samples results in less observed terms occurring in those chunks, which may lead to an information loss. As a result, the parameters can be over-estimated, impairing the overall performance. On the other hand, for WT10G, two factors contribute to the more robust performance with respect to the number of feedback documents. The first is the style of data sets. Both AP88–90 and ROBUST are the data sets containing homogeneous news articles, which are often compact documents, while WT10G documents are web pages with a variety of subjects and styles. The second is the size of documents. The average document length in WT10G (378) is about 50% larger than that of the AP88-90 (245) and ROBUST (254).

A more systematic investigation in the impact of data set characteristic is out of scope of this paper and will be left as in interesting future research direction on its own.

Table 8: Comparison of Robustness of AHMM-II vs. RM2

No. of docs (N)	**#Pos.**	**#Neg.**	**#Eq.**	**# Pos.-Neg.**
Test Q151–200 on AP88–90				
5	32	18	0	+14
10	36	14	0	+22
15	35	15	0	+20
Test Q601–700 on ROBUST				
5	73	25	1	+48
10	88	11	0	+77
15	90	9	0	+81
Test Q501–550 on WT10G				
5	32	18	0	+14
10	36	14	0	+22
15	30	20	0	+10

We also compare the robustness of different approaches in the true relevance feedback scenario. Table 8 shows the robustness when using different number of relevance-feedback documents over the three data collections (AP88-90, ROBUST, and WT10G). Here, we compare the AHMM-II with RM2. The comparisons of AHMM-I with other models show a similar trend. The same as we have analyzed above, although the AHMM-based approach impairs a few more queries when selecting a smaller number (i.e., 5) of documents, significant improvements are obtained in all settings. When 10 relevance-feedback documents are selected, AHMM-II improves 36, 88 and 36 queries while only impairs 14, 11 and 14 queries. When 15 documents are selected, AHMM-II improves 35, 90 and 30 queries while impairs 15, 9 and 20 queries.

7.3 Effects of Window Size

In an information retrieval system, a high-quality model can obtain better performance with fewer manually tunned parameters. In the process of building a high-quality model, the parameters used in the model should always occur in a fixed range, or the model should be less sensitive to the parameters.

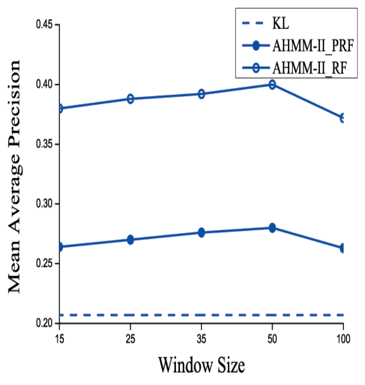
Some state-of-the-art methods, such as the relevance models, have shown good retrieval performance. However, they still need to manually tune or learn some parameters to optimize the retrieval performance for different data sets. In the experiments, our method has one parameter to set: the size of the sliding window. We will show how our method is relatively insensitive to this parameter.

In this paper, we use a sliding window to segment the top-ranked documents into chunks. Thus, we need to compare the retrieval performances based on the different window sizes. Here, we select 10 documents as feedback in the settings of pseudo-relevance feedback and true relevance feedback. Figure 5 shows the impact of window size on MAP when using AHMM-II on three data sets with different window sizes, containing 15, 25, 35, 50, 100 words respectively. In each figure, three curves are shown, which are AHMM-II based on pseudo-relevance feedback (AHMM-II_PRF), AHMM-II based on true relevance feedback (AHMM-II_RF), and the baseline language model (KL), respectively.

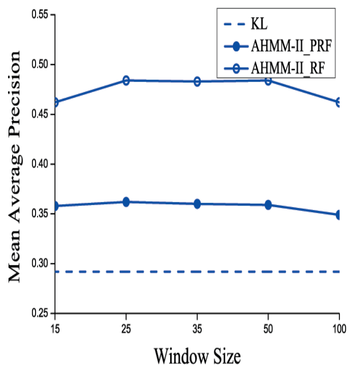
According to the figures, using a sliding window with size in the range of 15–50 words can obtain almost identical retrieval performances. When the s-liding window of 100 words is used, the performance slightly drops compared with using a smaller window size, but in general, the performances stay rather stable with respect to different window sizes. Therefore, our proposed approach seems quite insensitive to the window size. This may be due to the use of overlapping window which can help stablize the effect of window size and improve the acquisition of the dependencies between terms.

7.4 Acquisition of the Optimal Performance

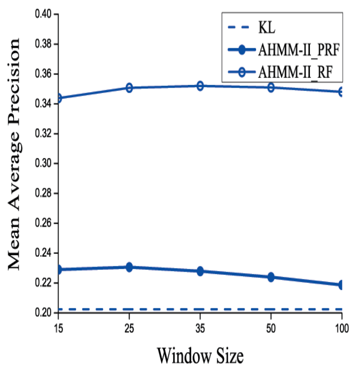
In Section 5, we discussed two options for integrating the derived query model with the original query model, namely, AHMM-I and AHMM-II. AHMM-II is a fully automatic approach and has been described in detail in the previous



(a) Test Q151–200 on AP8890



(b) Test Q601–700 on ROBUST



(c) Test Q501–550 on WT10G

Figure 5: Effect of the window size on MAP

sections. Unlike AHMM-II, the construction of model using AHMM-I, which is based on linear interpolation of two query models, needs some manual adjustment or extra supervised learning to set the interpolation coefficient. Therefore, in this subsection, we take a brief look into the performance of AHMM-I in comparison with AHMM-II. The definition of AHMM-I is given as below:

$$P_{AHMM-I}(w|\theta_Q) = \lambda P(w|\theta_Q) + (1-\lambda)P(w|Q) \tag{11}$$

$P_{AHMM-I}(w|\theta_Q)$ is the model learned from the AHMM. $P(w|Q) = 0$ when the term w does not occur in the original query.

This method needs to determine the linear interpolation coefficient λ. The interpolation method has been widely used in information retrieval. For example, the LCE model in (Metzler and Croft 2007) also needs training to estimate the mixture coefficients used in the MRF model and set the experience value for the related interpolation factor used in the feature functions.

According to Equation 11, a larger λ means a smaller effect of the original query model. Our experiments show that the MAP reaches the largest value when λ is selected in certain fixed range. For both pseudo-relevance feedback and true relevance feedback, the largest MAP values are obtained when λ is set to be 0.98 on AP8890 and ROBUST. For WT10G, the optimal MAP is achieved when λ is set to be 0.85 for pseudo-relevance feedback and 0.9 for relevance feedback. As shown in Section 6.1, the characteristics of the data sets could play an important role because the AP88–90 and ROBUST collections are generally homogeneous news articles while WT10G documents are Web pages with a variety of subjects and styles. As a comparison, we list the MAPs obtained by AHMM-II and AHMM-I (with manually determined best performing interpolation coefficients) in Table 9.

Table 9: Comparison of Optimal MAPs using AHMM-I and AHMM-II

Pseudo-relevance Feedback			
Collection	AHMM-II	AHMM-I	Impr. (%) (AHMM-I over AHMM-II)
AP88–90	0.2814	0.2830	+0.5
ROBUST	0.3613	0.3660	+1.3
WT10G	0.2305	0.2370	+2.8
Relevance Feedback			
Collection	AHMM-II	AHMM-I	Impr. (%) (AHMM-I over AHMM-II)
AP88–90	0.4034	0.4168	+3.3
ROBUST	0.5014	0.5122	+2.1
WT10G	0.3764	0.3859	+2.5

It is found that using AHMM-I with the manually adjusted best performing setting can only obtain slightly and insignificantly better performances than AHMM-II. However, AHMM-I is a somehow heuristic approach compared with AHMM-II which integrates original query model neatly in the AHMM in a fully automatic way. Therefore, the AHMM-II has demonstrated its robustness and effectiveness from both theoretical and practical perspectives.

7.5 Remarks

AHMM-II shows a superior ability to detect and estimate the dependencies between the observed terms and document chunks, and the subsets of query terms based on the following methods:

Remarks (1): In our experiments, for both pesudo-relevance feedback true relevance feedback scenarios, we may be encountered the data sparsity issue for model estimation. In the process of building the model, we need to run the AHMM over a limited number of chunks, which sometimes are generated from only e.g., 10 relevance feedback documents. Therefore, in our experiments, the iteration only runs once because the issue of over-fitting may lead AHMM to converge to a local maximum.

Remarks (2): In addition, we compare the time complexity of our proposed AHMM model with the basic AM model. All the experimental runs were based on the configuration of a computational server with 3200 MHz CPU and 4GB main memory. Our methods were implemented in Perl.

The Step 1 in Figure 4 is the query processing and document processing. The data pre-processing of the AHMM is the same as the AM. Suppose the number of terms in a query Q is $|Q|$. For query decomposition, the time complexity of this part is $2^{|Q|}$. The window size is $k = 25$. The overlapping length is $4/5$ of the window size. For one document D, the document length is $|D|$ and the number of the chunks is $\lceil (|D| - k)/5 \rceil$. For each document, it turns out that the time complexity is $O(2^{|Q|} \cdot (|D|))$ for the data-processing. We measured that the data pre-processing runs about 0.28 second per query (ROBUST2004 collections, topics 601-700).

In both AM and AHMM, we need to optimize the model parameters $P(d|S)$, $P(w|S)$, $P(S)$. Then those learned parameters are used to estimate the models. For AM model, the Expectation Maximization (EM) algorithm is used in the optimization process. For AHMM, a Viterbi process is adopted to optimize those parameters (Figure 4). The computational cost of AM and AHMM depend on the convergence performance of the optimization algorithms. We measured the efficiency of the proposed AHMM and AM by recording the time of optimization process. One of the baseline, e.g., RM1, the average time for query expansion process is about 1.13 second. The average time to complete the optimization process is approximately 0.9 second (for AM) and 1.29 second (for AHMM) per query. This gap is mainly due to that AM takes into account the on-the-fly EM optimization, while AHMM utilizes Viterbi algorithm to search the optimal state sequences. We consider it as a worthwhile tradeoff, as AHMM can improve the performance by about 4% (on average) over AM model when N=30 in Table 3.

Remarks (3): We have also done some evaluation of the AHMM model using the description field of topic as long queries, with the same setting of the maximum length of S_j. The model shows large improvements over the baselines for long queries as well. For ROBUST2004, in the comparison with KL, AM and RM models, our AHMM model improves the performance by 12.1%, 15.1% and 18.2% separately (Song et al. 2013). By adding the HMM layer on top of the AM structure, the performance has been largely improved. This may be because our method has learnt reasonable weights for query terms during the HMM model optimization process. These can reflect the effectiveness of our

AHMM more comprehensively.

8 Conclusions and Future Work

In this paper, we present a novel Aspect Hidden Markov Model for query expansion, with focus on incorporating the dependencies between query terms via an ergodic Hidden Markov Model. The HMM is used to estimate the structure of the AHMM, without needing any pre-existing training data.

Our experimental results show that our method always obtains significant improvements in comparison with a number of strong baselines (KL, RM1 and RM2). Even when compared with the state-of-the-art LCE used in (Metzler and Croft 2007) and the Aspect Query Language Model in (Song et al. 2012), our method shows a better performance in MAP on three large data sets. We have also evaluated the robustness of our method against the comparative models. Our method has shown a better robustness on all the three data sets. In addition, our method not only generates a good performance, but also needs fewer parameters to tune.

In terms of future work, it would be interesting to continue the studies on how to measure the quality of feedback documents and the relevant contents in these documents. In addition, checking the impact of the dataset characteristics on the setting of parameters and further reducing the model's sensitivity to the number of feedback documents are also challenging topics. Another issue that needs further exploration is how to reduce the impact of data sparsity when a small number of documents are selected.

9 Acknowledgments

This work is funded in part by the Chinese National Program on Key Basic Research Project (973 Program, grant no. 2013CB329304), the Natural Science Foundation of China (grant no. 61272265, 61070044, 61105702), and the European Union Framework 7 Marie-Curie International Research Staff Exchange Programme (grant no. 247590).

References

The lemur toolkit for language modeling and retrieval. In *http://www.lemurproject.org.*

R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 207–216, Washington D.C., May 1993.

A. Al-Maskari, M. Sanderson, and P. Clough. Accurately interpreting click-through data as implicit feedback. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval(SIGIR'2008)*, pages 683–684, 2008.

J. Bai, D. Song, P. Bruza, J. Nie, and G. Cao. Query expansion using term relationships in language models for information retrieval. In *CIKM 2005*, pages 688–695, 2005.

D. M. Blei and P. J. Moreno. Topic segmentation with an aspect hidden markov model. In *Proceedings of ACM 24th SIGIR Conference on Research and Development in Information Retrieval(SIGIR'2001)*, pages 343–348, New Orleans, Louisiana, USA, September 2001.

P. Bruza, K. Kitto, L. Sitbon, D. Song, and S. Blomberg. Quantum-like non-separability of concept combinations, emergent associates and abduction. *Logical Journal of the IGPL*, pages 445–457, 2011.

G. Cao, J. Nie, and J. Bai. Integrating term relationships into language models. In *Proceedings of ACM 28th SIGIR Conference on Research and Development in Information Retrieval(SIGIR'2005)*, pages 298–305, 2005.

C. Creighton and S. Hanash. Mining gene expression databases for association rules. bioinformatics. *Bioinformatics*, pages 79–86, 2003.

J. Gao, J. Nie, G. Wu, and G. Cao. Dependence language model for information retrieval. In *Proceedings of ACM 28th SIGIR Conference of Research and Development in Information Retrieval(SIGIR'2004)*, pages 170–177, 2004.

D. Hiemstra and W. Kraaij. A language modeling approach to the text retrieval context. *Digital Libraries and Electronic Publishing*, pages 373–396, 2005.

J. Hipp, U. Guntzer, and G. Nakhaeizadeh. Algorithms for association rule mining - a general survey and comparison. *ACM SIGKDD Explorations NewsLetter*, pages 58–64, 2000.

E. Hoenkamp, P. Bruza, D. Song, and Q. Huang. An effective approach to verbose queries using a limited dependencies language model. In *Proceedings of The 2nd International Conference on the Theory of Information Retrieval(ICTIR'2009)*, pages 116–127, 2009.

Q. Huang and D. Song. A latent variable model for query expansion using the hidden markov model. In *Proceedings of ACM 17th Conference on Information and Knowledge Management (CIKM)*, 2008.

M. Iwayama. Relevance feedback with a small number of relevance judgements: incremental relevance feedback vs. document clustering. In *Proceedings of the*

23rd annual international ACM SIGIR conference on Research and development in information retrieval(SIGIR'2000), pages 10–16, 2000.

T. Joachims, L. Granka, B. Pan, H. Hembrooke, and G. Gay. Accurately interpreting clickthrough data as implicit feedback. In *Proceedings of ACM 28th SIGIR Conference on Research and Development in Information Retrieval(SIGIR'2005)*, pages 154–161, 2005.

J. Lafferty and C. Zhai. Document Language Models, Query Models, and Risk Minimization for Information Retrieval. In *Proceedings of the 24th Annual ACM Conference of Research and Development in Information Retrieval (SIGIR'2001)*, pages 111–119. ACM Press, 2001.

V. Laverenko, M. Choquetto, and W. B. Croft. Cross-lingual language models. In *Proceedings of ACM 25th SIGIR Conference on Research and Development in Information Retrieval*, 2002.

V. Lavrenko and W. Croft. Relevance-based language models. In *Proceedings of ACM 24th SIGIR Conference On Research and Development in Information Retrieval(SIGIR'2001)*, pages 120–127, 2001.

V. Lavrenko, J. Allan, E. DeGuzman, D. LaFlamme, V. Pollard, and S. Thomas. Relevance models for topic detection and tracking. In *Proceedings of the Conference on Human Language Technology (HLT)*, pages 115–121, 2002.

J. Luo and S. Bridges. Mining fuzzy association rules and fuzzy frequency episodes for intrusion detection. *International Journal of Intelligent Systems*, pages 687–703, 2000.

C. D. Manning, P. Raghavan, and H. Schtze. *Introduction to Information Retrieval.* Cambridge University Press, 2008.

D. Metzler and W. B. Croft. A markov random field model for term dependencies. In *Proceedings of ACM 28th SIGIR Conference on Research and Development in Information Retrieval(SIGIR'2005)*, pages 472–479, 2005.

D. Metzler and W. B. Croft. Latent concept expansion using markov random fields. In *Proceedings of ACM 30th SIGIR Conference on Research and Development in Information Retrieval(SIGIR'2007)*, pages 311–318, 2007.

D. R. H. Miller, T. Leek, and R. M. Schwartz. A hidden markov model information retrieval system. In *Proceedings of SIGIR conference on research and development in information retrieval(SIGIR'1999)*, pages 214–221, Berkeley, California, USA, 1999.

R. Nallapati and J. Allan. Capturing term dependencies using a language model based on sentence trees. In *Proceedings of the 11th International Conference on Information and Knowledge Management (CIKM)*, pages 383–390, 2002.

J. Ponte and W. Croft. A language modeling approach to information retrieval. In *Proceedings of the 21st Annual ACM Conference of Research and Development in Information Retrieval(SIGIR'1998)*, pages 275–281, 1998.

L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In *Proc. IEEE*, pages 257–286, 1989.

D. Song, Q. Huang, P. Bruza, and R. Lau. An aspect query language model based on query decomposition and high-order contextual term associations. *Computational Intelligence*, pages 1–23, 2012.

D. Song, Y. Shi, P. Zhang, Y. Hou, B. Hu, Y. Jia, Q. Huang, U. Kruschwitz, D. Roeck, and P. Bruza. Optimization of an integrated model for automatic reduction and expansion of long queries. In *Lecture Notes in Computer Science (LNCS)*, 2013.

J. Srivastava, R. Cooley, M. Deshpande, and P.-N. Tan. Web usage mining: Discovery and applications of usage patterns from web data. *ACM SIGKDD Explorations NewsLetter*, pages 12–23, 2000.

C. Zhai and J. Lafferty. Model-based feedback in the language modeling approach to information retrieval. In *Proceedings of ACM 10th Conference on Information and Knowledge Management (CIKM)*, pages 403–410, 2001.

J. Zhao, J. X. Huang, and B. He. Crter: using cross terms to enhance probabilistic information retrieval. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 155–164, 2011.

APPENDIX

The equation in Figure 4

$P(S_j|d_{i,k}) = \frac{1}{k+1}\frac{P(w_k|S_j)P(S_j|d_{i,k-1})}{\sum_{S_t} P(w_k|S_t)P(S_t|d_{i,k-1})} + \frac{k}{k+1}P(S_j|d_{i,k-1})$

The derivation is showed as follows:

for k = 1 : K

$P(S_j|d_{i,k})$

$= P(S_j|d_{i,k-1}, w_k)$

$= \frac{P(S_j, w_k|d_{i,k-1})}{P(w_k|d_{i,k-1})}$

$= \frac{P(w_k|S_j, d_{i,k-1})P(S_j|d_{i,k-1})}{P(w_k|d_{i,k-1})}$

$= \frac{P(w_k|S_j, d_{i,k-1})P(S_j|d_{i,k-1})}{\sum_{S_t} P(w_k|S_t)P(S_t|d_{i,k-1})}$

When w_k is independent from $d_{i,k-1}$, $P(S_j|d_{i,k}) \propto \frac{P(w_k|S_j)P(S_j|d_{i,k-1})}{\sum_{S_t} P(w_k|S_t)P(S_t|d_{i,k-1})}$

When w_k is independent from $d_{i,k-1}$, $P(S_j|d_{i,k}) \propto P(S_j|d_{i,k-1})$

Then, $P(S_j|d_{i,k}) = \frac{1}{k+1}\frac{P(w_k|S_j)P(S_j|d_{i,k-1})}{\sum_{S_t} P(w_k|S_t)P(S_t|d_{i,k-1})} + \frac{k}{k+1}P(S_j|d_{i,k-1})$

end