Optimization of an integrated model for automatic reduction and expansion of long queries

Optimization of an Integrated Model for Automatic Reduction and Expansion of Long Queries

Dawei Song[1,2], Yanjie Shi[1], Peng Zhang[1], Yuexian Hou[1], Bin Hu[3], Yuan Jia[4], Qiang Huang[5], Udo Kruschwitz[6], Anne De Roeck[2], and Peter Bruza[7]

[1] Tianjin University, Tianjin, China
[2] The Open University, Milton Keynes, UK
[3] Lanzhou University, Lanzhou, China
[4] Chinese Academy of Social Sciences, Beijing, China
[5] University of East Anglia, Norwich, UK
[6] University of Essex, Colchester, UK
[7] Queensland University of Technology, Brisbane, Australia
{dawei.song2010,sandy.y.shi,darcyzzj,krete1941}@gmail.com
bh@lzu.edu.cn,summeryuan_2003@126.com,h.qiang@uea.ac.uk
udo@essex.ac.uk,anne.deroeck@open.ac.uk,p.bruza@qut.edu.au

Abstract. A long query provides more useful hints for searching relevant documents, but it is likely to introduce noise which affects retrieval performance. In order to smooth such adverse effect, it is important to reduce noisy terms, introduce and boost additional relevant terms. This paper presents a comprehensive framework, called Aspect Hidden Markov Model (AHMM), which integrates query reduction and expansion, for retrieval with long queries. It optimizes the probability distribution of query terms by utilizing intra-query term dependencies as well as the relationships between query terms and words observed in relevance feedback documents. Empirical evaluation on three large-scale TREC collections demonstrates that our approach, which is automatic, achieves salient improvements over various strong baselines, and also reaches a comparable performance to a state of the art method based on user's interactive query term reduction and expansion.

1 Introduction

Long queries can be viewed as a rich expression of a user's information need and have recently attracted much attention [1, 12]. An example long query (the description field of TREC topic 382) is shown in Figure 1.

Long queries may only form a fraction of queries actually submitted by searchers on the Web but they do represent a significant part [6]. For more specialized search engines long queries are very common, e.g. the queries submitted to the legal search service *Westlaw* are on average about 10 words long [15]. Medical search engines have also been developed to handle long queries

Identify documents that discuss the use of hydrogen as a fuel for piston driven automobiles (safe storage a concern) or the use of hydrogen in fuel cells to generate electricity to drive the car

Fig. 1. An example of a long query

such as plain English text [14]. Furthermore, one could also see pseudo-relevance feedback as an attempt to make the query longer.

Intuitively, the useful hints and rich information carried by a long query can be leveraged to improve search performance. However, the rich information is also likely to bring interference caused by possible irrelevant terms and the verbosity surrounding the key concepts in the long queries [1]. One type of "bad" terms are those completely irrelevant to user's information need, such as "identify" and "documents" in the above example, and can be viewed as noise. The other type are terms that are weakly relevant, such as "drive" and "storage" in the example. They may result in query shift although they do carry some useful information. Table 1 shows the retrieval performances of TREC Topics 251 $\sim$ 300 on ROBUST05 collection using the title and description fields of each topic, respectively. The title field's average length is about 3 terms per topic. The description field is much longer, on average about 17 terms per topic. We can find that using title field as query generates higher precision and recall than the use of description field, over two typical IR models (vector space model based on TF-IDF and language model using Kullback-Leibler (KL) divergence), and a widely used pseudo-relevance feedback model - the Relevance Model (RM) [13].

ROBUST05						
	Title+TFIDF	**Des.+TFIDF**	**Title+KL**	**Des.+KL**	**Title+RM**	**Des.+RM**
Precision	**0.1721**	0.1154	**0.1951**	0.1544	**0.2297**	0.1994
Recall	**0.3729**	0.3143	**0.3876**	0.3651	**0.3892**	0.3806

Table 1. The retrieval performances of queries 251-300 on ROBUST05 collection

Therefore, to improve the retrieval effectiveness for long queries, it is crucial to identify and boost the important terms and meanwhile eliminate the negative effects of "bad" terms in the original or expanded queries.

1.1 Related Work

One way of alleviating the problem is to use interactive techniques and let users select some terms or combinations of various terms, as key concepts, from long queries, through interactive query reduction (IQR) and interactive query expansion (IQE) [11, 10, 4, 18, 9]. IQR in general aims to help users remove noisy information - the first type of interference we described earlier. However, it is not good at handling the second type of "bad" terms. This is because users often can only decide, approximately, whether a term is important or not, rather than accurately quantifying the importance of these terms. Moreover, it cannot introduce additional relevant terms to the original query. Thus interactive query expansion (IQE) is used to help users remove irrelevant terms suggested by automatic query expansion techniques. Further, [11] integrates IQR and IQE into a single framework for Selective Interactive Reduction and Expansion (SIRE). Although both IQR and IQE can result in salient improvements for document retrieval with long queries, such user-dependent approaches have their limitations. Too much user interference may increase the time and cognitive overhead, if users are required to read through each long query and the search results from a baseline system and decide which terms should be selected. On the other hand, users may not be able to easily give an accurate estimation of the weight for each selected term, which can be effectively computed from automatic methods. In any case, it is well recognized that users are reluctant to leave any explicit feedback when they search a document collection [3, 8, 16].

Therefore, it makes sense to process long queries automatically to estimate the importance of query terms. In [1], a method to identify key concepts in verbose queries using supervised learning is proposed. However, it relies on prelabeled training data. In term of automatic query expansion, there has been many approaches in the literature. One of the state of the art automatic query expansion methods is the Relevance Model. Although it does not distinguish the combinations of multiple query terms as IQR or IQE does in [18], it estimates the probability distribution of expanded terms. Thus it can improve the positive effects of some key terms and reduce the negative effects caused by noisy and/or redundant information.

However, as shown in Table 1, existing automatic approaches such as TFIDF, as a basis for query reduction, and RM, as a basis for selecting expansion terms, are less effective for long queries. We think a possible reason is that these approaches treat the query terms as independent of each other. In reality, particularly for long queries, the cohesion of query terms, largely determined by the intra-query term dependencies, plays a key role in deciding which terms should be used to represent the information need.

The overall aim of this paper is to develop an effective automatic query reduction and expansion approach for long queries, which should take into account the intra-query term dependencies.

1.2 Our Approach

In this paper, we propose a method that effectively integrates query expansion and query reduction. First, an existing robust query expansion approach, called the Aspect Query Language Model (AM) [19], is used as a basis for an overall framework to derive an expanded query model (Query Expansion). In AM, the query terms are considered as latent variables over a number of observed top ranked documents after initial retrieval. Secondly, a Hidden Markov Model (HMM) is established over the AM structure, leading to an Aspect Hidden Markov Model (AHMM), to estimate the dependencies between the latent variables (as states of the HMM), and in turn to identify an optimal probabilistic

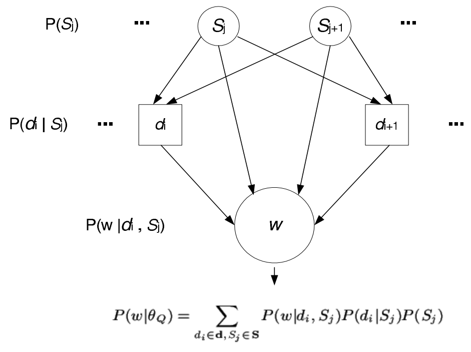
P(S)
P(d | S)
P(w |d , S)
S
S+1
d
d+1
w
$P(w|\theta_Q) = \sum_{d_i \in \mathbf{d}, S_j \in \mathbf{S}} P(w|d_i, S_j)P(d_i|S_j)P(S_j)$

Fig. 2. Structure of Aspect Query Language Model

distribution over the states (Query Reduction). The details of our approach will be introduced in the next section.

2 Model Construction

2.1 The Aspect Query Language Model (AM)

This section gives a brief description of the AM [19], where the subsets of query terms are viewed as latent variables (representing query aspects) over a number of top-ranked documents from the initial retrieval. The query aspects are treated as latent variables, as they (and their optimal weighting) can only be derived through the top ranked documents that we observe.

The AM structure is shown in Figure 2. 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 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 the word w and the latent variable S_j is derived from the relevance feedback documents. In practice, such as in Web search, the number of top ranked documents actually observed by users is often small [7], which will lead to the data sparsity problem given the large number of latent variables for a long query. Furthermore, not all the top ranked documents are truly relevant to the query. Even for a relevant document, it is not necessarily true that every part within the document is relevant. Thus, smaller chunks of the documents (e.g., segmented through a sliding window) are used to connect S_j and w in order to expand the observation space to overcome the data sparsity problem and improve the quality of parameter optimization.

Based on the structure in Figure 2, the following formula can be derived:

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

$P(S_j)$ 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.

An on-the-fly training data construction method is developed to automatically label the document chunks with query term(s). The Expectation Maximization (EM) algorithm is then used to fit the parameters of Equation 1. Despite the proven effectiveness of the AM, it does not take into account the intra-query term dependencies (i.e., dependencies between the latent variables).

2.2 Aspect Hidden Markov Model (AHMM)

We now present an AHMM approach that extends the original AM and allows a natural incorporation of the intra-query dependencies through the HMM mechanism. There has been evidence that the source of natural language text can be modelled as an "ergodic" Markov process, meaning the corresponding 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) [5]. As shown in Figure 3, 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 4 shows an example structure of a 3-state ergodic HMM.

We now 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. Given the observation chunks $\boldsymbol{d} = \{d_1, \cdots, d_T\}$ and a model $\Lambda = (A, B, \pi)$, the HMM can choose a corresponding state sequence $(S_1, \cdots, S_T)$ that is optimal (i.e., best "explains" the observations). For the purpose, the Viterbi algorithm is used. Due to space limit, the algorithm is not detailed here (See [17] for details).

In this paper, the query terms are used as HMM states (S_j), which are not really "hidden" in a strict sense. We adopt the HMM structure for the effective optimization mechanisms that HMM provides. In the process of learning the

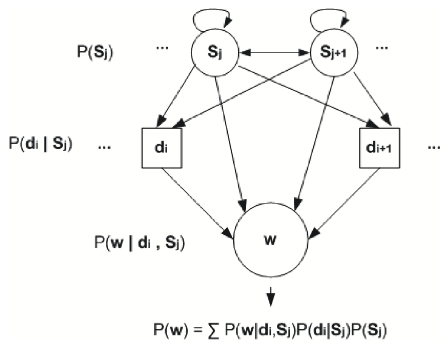

Fig. 3. Structure of Aspect Hidden Markov Model

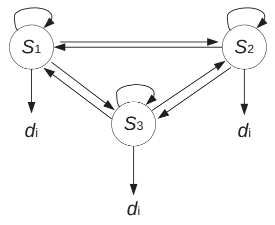

Fig. 4. An example of three-state Ergodic Hidden Markov Model

model, we utilize the Baum-Welch algorithm to optimize the state distribution and transition matrix and the Viterbi algorithm to search the optimal path [17] to update the probability distribution of each chunk in different states. The update of $P(d_t|S_j)$ is based on:

$$P(d_i|S_j) = \frac{\sum_{t=1,d_t=d_i}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)} \tag{2}$$

where $\gamma_t(j) = P(S_j|d_t, \Lambda)$. The value of $\gamma_t(j)$ is estimated by using an iterative computation detailed in Fig. 5.

In summary, the HMM would seem to provide a mechanism to estimate the parameters listed in Eq. 1.

3 Model Optimization

For estimation of the probabilities in Eq. 1 within the AHMM structure, we adapt the original parameter estimation algorithms in AM [19] to AHMM, as shown in Figure 5.

```
1. Pre-processing
   1.1 View the single query terms in query $Q$ as "hidden" states.
   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.
   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 "hidden" 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.
       $P(S_j) = 1/M, \quad (M = |\mathbf{S}|, \quad S_j \in \mathbf{S})$
       $P(S_{j'}|S_j) = 1/M, \quad (M = |\mathbf{S}|), \quad S_j, S_{j'} \in \mathbf{S})$
       $P(d_i|S_j) = \frac{\sum_{t=1,d_t=d_i}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}$
   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 3.2.
   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 Step 2.1 $\sim$ Step 2.3.
3. Derive the language model
   Compute the final probability of $P_{AHMM}(w|\theta_Q)$ using Eq. 1.
```

Fig. 5. Outline of framework

3.1 Data Pre-processing (Step 1)

The same data pre-processing procedure used in AM is applied here. Step 1.1 takes the query term combinations as states, as discussed in Section 2.1. Step 1.2 selects relevance feedback documents. In Step 1.3, each feedback document is segmented with an overlapped sliding window, as discussed in Section 2.1.

Step 1.4 can be seen as a coarse data refinement by keeping only the chunks containing at least one query term. Step 1.5 can be considered as an automatic on-the-fly training data construction process, which labels the selected chunks with different states. With these automatically labeled 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.

3.2 Model Estimation (Steps 2 & 3)

According to the description of AHMM in Section 2.2, 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 HMM. A recursive method is used to compute $P(d_i|S_j)$. This method for $P(d_i|S_j)$ was also used in [2] for an online estimation. In this recursive equation, $d_{i,k}$ denotes the first k words in 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 $P(S_j)$. The conditional probability $P(w_k|S_j)$ is computed as:

$$P(w_k|S_j) = \frac{\sum_{d_i} \#w_{k,i} * P(d_i|S_j)}{\sum_{w_k} \sum_{d_i} \#w_{k,i} * P(d_i|S_j)} \tag{3}$$

where $\#w_{k,i}$ is the occurrence frequency of w_k in d_i. 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 AHMM. Since the contribution of $P(d_i|S_j)$ has been considered in Eq. 3 to compute $P(w_k|S_j)$, Eq. 1 can be simplified as:

$$P(w|\theta_Q) = \sum_{S_j} P(w_k|S_j) * P(S_j) \tag{4}$$

4 Experimental Setup

We evaluate our method using TREC topics 251–300 on the TREC5 collection (TREC disk 2 and 4), topics 303–689 (50 selected queries) on the ROBUST05 collection (AQUAINT collection), and topics 301–450 and 601–700 on the ROBUST04 collection (TREC disk 4 and 5 excluding the *Congressional Record*). The *description* field of the topics are used as queries (with an average query length of 15-17 words). They are selected because they have varied content and document properties. The Robust tracks are known to be difficult, and conventional IR techniques have failed on some of them [11]. In our experiments, a standard stopword list and the Porter stemmer are applied to all data collections. Note that the same experimental setting was also used in [11] for user interactive IQR and IQE. This allows a direct comparison of our method with this interactive approach.

Three baselines are used for comparison including a language model based on Kullback-Leibler (**KL**) divergence, the Relevance Model (**RM**) and the AM. We also compare our methods with the user based interactive system based on its results reported in [11]. For pseudo-relevance feedback, all the methods are tested with a certain number N of top-ranked documents from the initial retrieval results. Here, we only select $N = 30$ documents as feedback, as we think this is close to real web search scenarios. Note that we have tested a range of N (10, 20, ..., 100), and the performance of our approach is in general quite stable with respect to N. The documents are then segmented into chunks using a sliding window of 15 words with 1/3 overlapping between two consecutive windows. After applying each query expansion algorithm, we choose the top ranked 100 terms as expansion terms. All the experiments are carried out using the Lemur toolkit 4.0[1]. The initial retrieval is run by a widely used language model based

[1] `http://www.lemurproject.org`

on KL-divergence. The expanded queries derived from three different query language modeling methods (RM, AM and AHMM) are used to perform the second round of retrieval using KL divergence. Our primary evaluation metric is mean average precision (MAP). The Wilcoxon singed rank test is used to measure the statistical significance of the results.

5 Result and Analysis

5.1 Illustration of AHMM Optimization

	#Iter				
Term	1	2	3	4	5
automboil	0.0503	0.0489	0.0446	0.0435	0.0459
fule	0.0423	0.0543	0.0677	0.0834	0.0928
discuss	0.0377	0.0368	0.0410	0.0414	0.0439
safe	0.0823	0.0785	0.0798	0.0787	0.0778
electr	0.0646	0.0814	0.0922	0.0845	0.0893
car	0.0717	0.0926	0.1056	0.1061	0.1039
gener	0.1102	0.0999	0.0855	0.0853	0.0884
document	0.0545	0.0606	0.0635	0.0628	0.0624
concern	0.0333	0.0359	0.0388	0.0379	0.0397
storage	0.0653	0.0602	0.0543	0.0463	0.0382
hydrogen	0.0944	0.0816	0.073	0.0717	0.06
driv	0.0492	0.0556	0.0633	0.0633	0.0662
driven	0.0635	0.0704	0.0688	0.0695	0.0661
piston	0.0706	0.0597	0.0516	0.0522	0.0511
cell	0.1100	0.0836	0.0703	0.0734	0.0743
AP	**0.3982**	**0.4267**	**0.4447**	**0.4619**	**0.4483**

Table 2. Illustration of AHMM optimization

When the AHMM is applied, its probability parameters is iteratively optimized. Let us consider the example in Figure 1 again. Table 2 shows the changes of the probability of each query term and the query's retrieval performance (average precision) for each iteration. In the original query, some terms, such as *cell*, *safe*, *storage* and *gener*, seem only weakly relevant to the intention of the query. After a few iterations, their probability values are reduced. Simultaneously, the corresponding values of some key terms, such as *car* and *fuel*, are increased, This table also shows the positive trend of the performance when more iterations are run. The performance peaks at iteration 4 but then drops slightly at the 5th. This may be due to the overfitting caused by the data sparsity when optimizing the model. An in-depth investigation on this issue is left as future work.

Collection	**KL**	**RM**	**AM**	**AHMM**	**Improv. (%) of AHMM over KL & RM & AM**		
TREC5	0.1353	0.1484	-	**0.1715**	+26.7*	+15.6*	-
ROBUST05	0.1544	0.1994	-	**0.2435**	+57.7*	+22.1*	-
ROBUST04	0.2439	0.2375	0.2314	**0.2735**	+12.1*	+15.1*	+18.2*

$*$ Statistically significant at $p = 0.05$ (Wilcoxon signed rank test).

Table 3. Performance comparison (MAP) of KL, RM, AM and AHMM

Collection	**UIS-IQE**	**AHMM**	**Improv. (%)**
TREC5	0.168	0.1715	+2.08
ROBUST05	0.237	0.2435	+2.74
ROBUST04	0.292	0.2735	-6.3

Table 4. Performance comparison (MAP) with user based interactive query expansion

5.2 Comparisons in Query Expansion

The results of KL baseline, RM, AM and our approach are listed in Table 3. We can find that the AHMM shows good performance on all three data collections, and generates significant improvements over the KL baseline, RM and AM.

As a further comparison, we list the performances using user based interactive system (UIS) [11] in Table 4. In [11], the interactive query expansion (IQE) relies on users' help to select query expansion terms. In the comparison with IQE, our AHMM can also generate better performance on two data collections, TREC5 and ROBUST05, while the MAP value of our approach is lower than IQE on ROBUST04. Since the query set for ROBUST04 is a known difficult one, our model, as an automatic method, still gives a good performance.

5.3 Robustness of the Model

To further test the robustness of our model, we run another set of experiments, where we remove some "noisy" information in advance from the queries, through a process of *pre- query reduction*. This test is based on the assumption that the retrieval performance of the AHMM should keep a stable status, i.e., it is robust regardless of the selective removal of some noisy query terms.

Unlike the interactive query reduction, which is based on user manual selection, in our experiment, we use a simple automatic method for the pre- query reduction. We first collected all the queries we are using. Since there is an overlap between the query sets for ROBUST04 and ROBUST05, we obtain 300 queries altogether. We then counted the query frequency of each query term.

Our idea is to remove several query terms with high query frequencies, which can be viewed as "stop words" in the query set, such as "identifi", "document", etc. We set a threshold in term of the ratio of the query frequency to the total number of queries. In our experiments, it is set to be 0.1, a small number, because we want to be conservative and not to risk too much of mistakenly removing some useful terms.

Collection	KL	RM	AM	AHMM	Improv. (%) of AHMM over KL	& RM	& AM
TREC5	0.1482	0.1605	-	**0.1725**	+17.7*	+7.5*	-
ROBUST05	0.1611	0.2228	-	**0.2502**	+55.3*	+12.3*	-
ROBUST04	0.2500	0.2456	0.2364	**0.2758**	+10.3*	+12.3*	+16.7*

$*$ Statistically significant at $p = 0.05$ (Wilcoxon signed rank test).

Table 5. Performance comparison (MAP) of KL, RM, AM and AHMM, after pre-query reduction

Table 5 summarizes the performance of KL, RM, AM and the AHMM after applying the query term reduction. We observe a performance improvement on all models and collections, compared with the results without applying pre- query reduction (Table 3). The AHMM still outperforms the other three methods. The performance difference of AHMM with and without applying pre- query reduction is smaller than that of the other models. The observations reflect the robustness of the AHMM.

In both scenarios (i.e., with and without pre- query reduction), for ROBUST04, a well-known difficult collection where RM and AM underperform the KL baseline, AHMM achieves significant improvements over all the other models. The AM itself performs less well for long queries (in contrast of the good performance for short queries as reported in [19]). By adding the HMM layer on top of the AM structure, the perfomance is largely improved. This indicates our method learns more reasonable weights for query terms than the AM through the HMM-based model optimization process.

6 Conclusions and Future Work

We have presented an AHMM method for effective query expansion and query reduction for long queries, by estimating the intra-query term dependencies and the relationships between query terms and other words through observed relevance feedback documents. Our experimental results show that our method achieves significant improvements in comparison with three baselines: KL, RM and AM, showing the effectiveness and robustness of the proposed approach. Even when compared with the user-based interactive system used in [11], our approach, which is automatic, still shows a comparable performance. In the future, it would be interesting to study how to effectively and efficiently combine the user interactions with our automatic algorithm. In addition, to tackle the data sparsity problem when selecting fewer number of documents, we will consider smoothing our model with the background collection model.

Acknowledgments. This work is funded in part by the Chinese National Program on Key Basic Research Project (973 Program, grant no. 2013CB329304 and 2014CB744604), the Natural Science Foundation of China (grant no. 61272265), and the European Union Framework 7 Marie-Curie International Research Staff Exchange Programme (grant no. 247590).

References

1. Bendersky, M., Croft, W. B.: Discovering key concepts in verbose queries. In: SIGIR 2008, pp. 491-498 (2008)
2. Blei, D. M., Moreno, P. J.: Topic segmentation with an aspect hidden Markov model. In: SIGIR 2001, pp. 343-348 (2001)
3. Dumais, S., Joachims, T., Bharat, K., Weigend, A.: SIGIR 2003 workshop report: implicit measures of user interests and preferences. In: SIGIR Forum, pp. 50-54(2003).
4. Harman, D.: Towards interactive query expansion. In: SIGIR 1998, pp. 321-331 (1998)
5. Hoenkamp, E., Bruza, P., Song, D., Huang, Q.: An effective approach to verbose queries using a limited dependencies language model. In: ICTIR 2009, pp. 116-127 (2009)
6. Huston, S., Croft, W. B.: Evaluating verbose query processing techniques. In: SIGIR 2010, pp. 291-298 (2010)
7. Jansen, B. J., Spink, A., Bateman, J., Saracevic, T.: Real life information retrieval: A study of user queries on the web. In: SIGIR Forum, pp. 5-17 (1998)
8. Jansen, B. J., Spink, A., Saracevic, T.: Real life, real users, and real needs: a study and analysis of user queries on the web. Information Processing and Management, 207-227(2000)
9. Kelly, D., Dollu, V. D., Fu, X.: The loquacious user: a document-independent source of terms for query expansion. In: SIGIR 2005, pp. 457-464 (2005)
10. Kumaran, G., Allan, J.: A case for shorter queries, and helping users create them. In: HLT-NAACL 2007, pp. 220-227 (2007)
11. Kumaran, G., Allan, J.: Effective and efficient user interaction for long queries. In: SIGIR 2008, pp. 11-18 (2008)
12. Kumaran, G., Carvalho, V. R.: Reducing long queries using query quality predictors. In: SIGIR 2009, pp. 564-571 (2009)
13. Lavrenko, V., Croft, W. B.: Relevance-based language models. In: SIGIR 2001, pp. 120-127 (2001)
14. Luo, G., Tang, C., Yang, H., Wei, X.: Medsearch: a specialized search engine for medical information retrieval. In: CIKM 2008, pp. 143-152 (2008)
15. Manning, C. D., Raghavan, P., Schütze, H.: Introduction to Information Retrieval. Cambridge University Press (2008)
16. Markey, K.: Twenty-five years of end-user searching, Part 1: Research findings. In: Journal of the American Society for Information Science and Technology, pp. 1071-1081 (2007)
17. Rabiner, L. R.: A tutorial on hidden markov models and selected applications in speech recognition. In: Proceedings of the IEEE, vol. 77, pp. 257-286 (1989)
18. Ruthven, I.: Re-examining the potential effectiveness of interactive query expansion. In: SIGIR 2003, pp. 213-220 (2003)
19. Song, D., Huang, Q., Bruza, P., Lau, R.: An aspect query language model based on query decomposition and high-order contextual term associations. In: Computational Intelligence, pp. 1-23 (2012)