Inf Retrieval J (2015) 18:26–50
DOI 10.1007/s10791-014-9246-7

A new approach to query segmentation for relevance ranking in web search

Haocheng Wu · Yunhua Hu · Hang Li · Enhong Chen

Received: 1 December 2013 / Accepted: 17 September 2014 / Published online: 28 September 2014

Abstract In this paper, we try to determine how best to improve state-of-the-art methods for relevance ranking in web searching by query segmentation. Query segmentation is meant to separate the input query into segments, typically natural language phrases. We propose employing the re-ranking approach in query segmentation, which first employs a generative model to create the top *k* candidates and then employs a discriminative model to re-rank the candidates to obtain the final segmentation result. The method has been widely utilized for structure prediction in natural language processing, but has not been applied to query segmentation, as far as we know. Furthermore, we propose a new method for using the results of query segmentation in relevance ranking, which takes both the original query words and the segmented query phrases as units of query representation. We investigate whether our method can improve three relevance models, namely *n*-gram BM25, key *n*-gram model and term dependency model, within the framework of learning to rank. Our experimental results on large scale web search datasets show that our method can indeed significantly improve relevance ranking in all three cases.

Keywords Web search · Query segmentation · Relevance ranking · Query processing · Re-ranking · BM25 · Term dependency model · Key *n*-gram extraction

H. Wu (✉) · E. Chen
University of Science and Technology of China, Hefei, China
e-mail: ustcwhc@outlook.com

E. Chen
e-mail: cheneh@ustc.edu.cn

Y. Hu
Alibaba.com, Beijing, China
e-mail: wugou.hyh@taobao.com

H. Li
Noah's Ark Lab of Huawei Technologies, Hong Kong, China
e-mail: hangli.hl@huawei.com

1 Introduction

Queries in web search are usually classified as one of three different categories: single phrase, combination of phrases, and natural language questions. Meanwhile, natural language questions only consist of a small percentage of queries. Traditionally, a query is viewed as a bag of words or a sequence of *n*-grams, and relevance models such as BM25 (Robertson and Walker 1994), term dependency model (Metzler and Croft 2005; Bendersky et al. 2011b), key *n*-gram model (Wang et al. 2012) utilize the words or *n*-grams as units of query representation.

A question that naturally arises here is: is it possible to improve search relevance by conducting query segmentation first and then using the results for query representation in relevance models? For example, if the query is "let it go mp3 download", then one may want to divide the query into three segments: "let it go/mp3/download". On the other hand, if the query is "hot dog", then one may want to view it as a phrase rather than two separate words. The assumption is that relevance can be improved by query segmentation in both cases.

Methods have been proposed for query segmentation. For example, Bergsma and Wang (2007) propose performing query segmentation by using a classifier. Hagen et al. (2011, 2012) suggest using unsupervised methods, more specifically, heuristic functions to conduct the task. Their methods outperform many existing methods and are considered state-of-the-art. For other methods see the work of Jones et al. (2006), Tan and Peng (2008), Zhang et al. (2009), Brenes et al. (2010), Huang et al. (2010), Risvik et al. (2003), Yu and Shi (2009).

Efforts have also been made to improve relevance ranking by using query segmentation. In most cases, the phrases in the segmented queries are directly used in the representations of queries. Most studies show that query segmentation is helpful, but either by using particularly small datasets (Bendersky et al. 2009; Roy et al. 2012; Hagen et al. 2012) or using nonstandard relevance ranking models (Li et al. 2011). There are also studies indicating that query segmentation does not help as expected (Roy et al. 2012).

In this paper, we study the problem of query segmentation for relevance ranking in web search. Our goal is to enhance state-of-the-art methods for the task and to investigate the problem with large scale experiments.

We first propose a new method for query segmentation, on the basis of re-ranking, which is proven to be powerful for making structure predictions on sequence data in natural language processing, for example, part-of-speech tagging. The idea is to first pass the sequence and make a prediction using a generative model to obtain the top *k* candidates and then to rank the candidates to select the best result using a discriminative model. Although the approach is not new, it does not seem to have been applied to query segmentation. We consider a specific implementation of the approach, which uses Hagen et al. (2011)'s method (unsupervised learning) to find the top *k* candidates and then uses SVM (supervised learning) to find the best segmentation among the candidates. We make a comparison with the methods proposed by Hagen et al. (2011, 2012) and Bergsma and Wang (2007).

Next, we propose a new method for using query segmentation in searching. We take both the original query words and the query phrases obtained in segmentation as units and represent the query as bag of "units" or sequence of "units" (e.g., "hot", "dog", "hot dog" are viewed as units) in the relevance models. We specifically construct *n*-gram BM25, key *n*-gram model (Wang et al. 2012), and dependency model (Metzler and Croft 2005; Bendersky et al. 2011b) by using the query representation, respectively. We next take the scores of the models as features of learning to rank and employ LambdaMART (Burges 2010) to construct the final relevance ranking models, respectively. We make comparisons with methods in which the same models are employed but no query segmentation is carried out, as well as methods in which the same models are employed but only segmented query phrases are utilized (similar to some of previous work).

We make use of benchmark datasets from previous query segmentation work. The first dataset contains 500 queries with each one segmented by three annotators (Bergsma and Wang 2007). The second dataset contains 4,850 queries with each one segmented by ten annotators (Hagen et al. 2011). We also use a large-scale dataset from a commercial web search engine for relevance ranking experiments. The dataset consists of 240,000 random queries, associated web pages (on average 43 pages for each query), and relevance judgments on the pages with respect to queries.

We have made several conclusions from the experimental results. (1) Our method of re-ranking in query segmentation works significantly better than state-of-the-art methods on two public benchmark datasets. (2) Our method of using query segmentation in relevance ranking can help improve search relevance in all three cases in which the relevance schemes are *n*-gram BM25, key *n*-gram model, and dependency model. The improvements in terms NDCG are statistically significant. (3) It is better to utilize both *n*-grams of original query words and *n*-grams of segmented query phrases in the relevance models. Our contributions are mostly empirical. The methods in (1) and (2) are novel to search, but they are not difficult to devise.

Our contribution in this paper includes (1) the proposal of a re-ranking approach to query segmentation and verification of its effectiveness, (2) the proposal of a method of using query segmentation and verification of its effectiveness, and (3) the confirmation of the effectiveness of query segmentation in web searching through large scale experiments.

The rest of the paper is organized as follows. Section 2 introduces related work. Section 3 describes our method of query segmentation based on re-ranking. Section 4 explains our method of using query segmentation in relevance ranking. Sections 5 and 6 present the experimental results on query segmentation as well as relevance ranking with query segmentation. Section 7 concludes the paper.

2 Related work

2.1 Query segmentation

One of the earliest works on query segmentation is by Risvik et al. (2003). They assume that a segment in a query should satisfy two conditions: it frequently occurs in different sources and has a high level of mutual information (MI) between words within the segment. They calculate the likelihood of a segment by using the product of MI within the segment and the frequency of the segment in a query log. Jones et al. (2006) also use mutual information to segment a query into phrases. They focus on two-word phrases and view as a phrase any two consecutive words for which the MI between them is above a threshold. Huang et al. (2010) make use of a web scale language model to address long query segmentation. They use MI scores obtained from the language model to create a segmentation tree and find the best segmentation by pruning the tree. None of these methods is evaluated on a public dataset.

Bergsma and Wang (2007) have published the first benchmark dataset for research on query segmentation, referred to as Bergsma–Wang–Corpus (BWC). The dataset contains

500 queries and each query is segmented by three annotators. Furthermore, 500 labeled training queries and 500 labeled development queries are added for supervised learning approaches. The dataset is used in several previous works (Brenes et al. 2010; Hagen et al. 2011, 2012; Tan and Peng 2008; Zhang et al. 2009; Li et al. 2011; Roy et al. 2012). Hagen et al. (2011) have released a larger dataset referred to as Webis-QSec-10 (WQS) containing 4,850 queries. The queries are sampled from the AOL query log, with segmentations annotated by ten annotators.

Both supervised and unsupervised methods have been developed for query segmentation. Bergsma and Wang (2007) adopt a supervised method, which exploits a classifier using indicator features such as positions and POS tags, statistical features such as phrase frequencies on the web or in the query log and dependency features such as the frequency of two adjacent words. Yu and Shi (2009) employ a supervised method based on conditional random fields (CRF) for which the model is learned from query logs. Bendersky et al. (2011a) propose jointly performing query segmentation, capitalization, and POS tagging on the query by leveraging CRF. Hagen et al. (2011, 2012), Tan and Peng (2008) and Zhang et al. (2009) exploit unsupervised methods, which employ heuristic functions and do not need model training. Hagen et al. and Tan et al. make use of web *n*-grams from a large web corpus and titles of Wikipedia articles. Zhang et al. compute segment scores from the eigenvalues of the correlation matrix with regard to the given query.

The advantage of the supervised approach is that it may achieve higher accuracy in segmentation, while its disadvantage is that labeled data is needed for training the model. On the other hand, the unsupervised approach might be less powerful, but it has lower development cost.

Hagen et al. (2011, 2012) report that their two methods outperform the existing unsupervised and supervised methods on the two public datasets BWC and Webis-QSec-10-Corpus. We take their two methods as baselines in this paper.

2.2 Relevance ranking

Relevance ranking is one of the key problems in web searching. Given a query, documents containing the query words are retrieved, each document is assigned a score by the relevance model and the documents are sorted on the basis of their scores. The relevance score of a document represents the relevance degree of the document with respect to the query. Traditionally, the relevance ranking model is manually created, with a few parameters to be tuned. For example, BM25 (Robertson and Walker 1994), Language Model for IR (Lafferty and Zhai 2001; Ponte and Croft 1998), and term dependency model (Metzler and Croft 2005; Bendersky et al. 2010, 2011b) are considered state-of-the-art schemes.

In the methods described above, *n*-gram (e.g., unigram, bigram, and trigram) is usually used as a unit for calculating the relevance between query and document. In fact, the query and document can be represented as two *n*-gram vectors, and the relevance between them can be calculated as the similarity between the two vectors (e.g., cosine similarity) (Xu et al. 2010). Intuitively, if the *n*-grams of the query occur more frequently in the document, then it is more likely that the document is relevant to the query. Methods have also been proposed to enhance relevance by conducting better *n*-gram based query document matching. For example, Wang et al. (2012) propose a method to extract key *n*-grams from the document (webpage) and then utilize the extracted key *n*-grams to augment query document matching.

In web searching, the title, anchor texts, URL, body and associated queries of a web page can be used as multiple fields (pseudo texts) of the page in calculation of the relevance score (Ogilvie and Callan 2003; Robertson et al. 2004; Wang et al. 2012). Title, URL and body are from the web page itself and reflect the author's view on the page. Anchor texts are from other pages and represent other authors' view on the page. Associated queries are from searchers and represent searchers' view on the page. Ogilvie and Callan (2003) suggest linearly combining language models in different fields and making use of the combined model in searches. Robertson et al. (2004) propose BM25F, as extension of traditional BM25, to utilize information from all fields of a webpage in a single model. Wang et al. (2012) separately calculate the BM25 score for each field and then use the scores as features in the ranking model.

Supervised learning techniques, called learning to rank, have recently been proposed and have proven useful in the automatic combination of relevance models to create the final ranking list of documents with respect to query, particularly in web search (Li 2011a, b; Liu 2011). Among the learning to rank methods, the method of LambdaMART (Burges 2010) is regarded as state-of-the-art.

2.3 Enhancing relevance ranking by query segmentation

Recently, the question of whether query segmentation can enhance relevance ranking has attracted researchers' interest. Several research groups have studied the problem (Bendersky et al. 2009; Hagen et al. 2012; Roy et al. 2012; Li et al. 2011). The conclusions from the investigations are not very consistent, however.

Bendersky et al. (2009) apply query segmentation into relevance ranking. Their goal is to investigate the effect of incorporating query segments into the dependency model based on Markov Random Fields (MRF) (Metzler and Croft 2005). They employ a linear model to combine the MRF scores of term matches, ordered segment matches and unordered segment matches. Their experiments on different TREC collections indicate that query segmentation can indeed enhance the performance of relevance ranking. Li et al. (2011) have studied query segmentation in web search by exploiting click-through data. They incorporate query segments into a language model and employ the EM algorithm to estimate the parameters. Experiments demonstrate the effectiveness of their method. There is difference between our method and the methods of Bendersky et al. (2009) and Li et al. (2011). Both the relevance ranking method and query segmentation method are supervised in our case, while their methods are all unsupervised.

Hagen et al. (2012) and Roy et al. (2012) have studied whether modifying queries by adding quotations of phrases into them can improve relevance ranking in a web search engine. Note that using quotations is nearly equivalent to utilizing segmented queries.[1] They submit the results of query segmentations by different methods to the search engine. Their results show that in most cases query segmentation can help generate better relevance ranking. However, it is sometimes better not to conduct query segmentation. They only treat the search engine as a black box and do not make use of the results of query segmentation inside the search engine.

[1] In practice, <1.12 % queries include quotations (White and Morris 2007).

3 Our method of query segmentation

3.1 Problem

Query segmentation separates the query into disjoint segments so that each segment roughly corresponds to a phrase (note that it is not necessary to be a phrase in a natural language). Given a query $Q = w_1 w_2, \ldots, w_n$ of length n where $w_i, i = 1, \ldots, n$ denotes a word. A segmentation of Q is represented as $S = s_1 s_2 \ldots s_m$ of length m where $s_i, i = 1, \ldots, m$ denotes a segment. There are 2^{n-1} possible segmentations and $(n^2 + n)/2$ possible segments for Q. Therefore, query segmentation is equivalent to selecting the best segmentation from among different possibilities given the query.

For convenience, we sometimes use breaks (boundaries between a pair of adjacent words) to represent a segmentation. A segmentation can also be represented as $B = b_1 b_2, \ldots, b_{(n-1)}$ of length $n - 1$ where $b_i \in \{1, 0\}, i = 1, \ldots, n-1$ denotes a break, 1 stands for making the break, and 0 stands for not making the break. There are $n - 1$ breaks for query Q of length n.

3.2 Method

We take a two-stage re-ranking approach to query segmentation. The two-stage approach is widely used in other tasks in natural language processing. To the best of our knowledge, this is the first time it has been used for query segmentation. The basic idea of re-ranking is as follows. The generative approach is usually faster but less accurate in structure prediction (in our case segmentation). In contrast, the discriminative approach is slower but more accurate. The re-ranking approach tries to leverage the advantages of both, by first quickly finding the top k candidates with the generative approach and then reliably selecting the best result from the top k candidates with the discriminative approach.

In the first stage, we employ Hagen et al. (2011)'s unsupervised method of Wikipedia-Based Normalization (WBN) to find the top k candidate segmentations. Dynamic programming is applied to generate the top k segmentations with the highest scores, which reduces the time complexity from $O(2^n)$ to $O(n^2)$ where n denotes query length. In the second stage, we construct a feature vector for each candidate of segmentation, employ the supervised learning method of SVM to re-rank the candidates, and finally find the best segmentation as output. The time complexity of the second stage is $O(kn)$. Therefore, the time complexity of the overall process still remains as $O(n^2)$.

Given a segmentation, WBN assigns a weight to each segment and sums up all the weights as the score of the entire segmentation. We choose the segmentations with the highest k scores. The score of segmentation S is defined as follows:

$$score(S) = \begin{cases} \sum\limits_{s \in S, |s| \geq 2} weight(s) & \text{if weight(s)} > 0 \text{ for} \\ & \text{all s} \in \text{S and } |\text{s}| \geq 2 \\ -1 & \text{else.} \end{cases}$$

where s is a segment and segments with length one are ignored. The weight of segment s is defined as below,

$$weight(s) = \begin{cases} |s|^2 + |s| \cdot \max_{t \in s, |t|=2} freq(t) & \text{if } s \text{ is Wikipedia title} \\ |s| \cdot freq(s) & \text{else.} \end{cases}$$

where t denotes a substring of s and *freq* denotes the frequency of a string in the corpus. The frequency is obtained from the Microsoft Web NGram Service.[2]

There are also other methods that can achieve good results in query segmentation. For example, the methods described in Sect. 2.1 can all be employed here in principle. We choose the WBN method in this paper, because it performs the best on the two public datasets.

We investigate the top ranked segmentations by the WBN method, and find that for a given query the probability of finding the ground truth appearing in the top six ranked segmentations is 94 %. Therefore, we only select the top six segmentations in our experiments. (See details in Sect. 5.1.4)

We adopt SVM (Joachims 2002) as the method to train the re-ranking model. We take the segmentations in the ground truth as positive examples, and the other segmentations among the top six segmentations as negative examples. Training data is then constructed, and Table 1 gives an example. The re-ranking model is trained with the training data.

3.3 Features of re-ranking model

Table 2 shows the features used in the re-ranking method. We cluster the features into four groups.

The first group of features are obtained from the WBN method described in Sect. 3.2. They represent statistics of the results created by WBN. The use of the features can leverage the power of WBN. See the features with prefix “Sgmt” in Table 2 .

The second group of features utilize MI, which has proven to be useful for query segmentation (Risvik et al. 2003; Jones et al. 2006; Huang et al. 2010). The assumption is that a good segmentation should have low MI values between segments and high MI values within a segment. See the features with prefix “MI” in Table 2.

The third group of features represent the characteristics of segmentation. All the features are of the Boolean type. The features offer additional information for segmentation decisions, which may not be easily incorporated into the model of the first stage. See the features with prefix “Is” in Table 2.

We observe that the top candidate segmentations with the highest scores tend to be close to the ground truth. The fourth group of features consider the similarities between the current segmentation and the top segmentation. Suppose that the current segmentation is $S\,(B = b_1 b_2, \ldots, b_{n-1})$ and the top segmentation is S^h $(B^h = b_1^h b_2^h, \ldots, b_{n-1}^h)$. We measure the similarities between the two segmentations in several ways. See the features with prefix “Top” in Table 2.

We focus on the study of query segmentation in English in this paper. Our method can be directly applied to other word-segmented languages (e.g. French, German). Our method can also be applied to non-word-segmented languages (e.g. Chinese, Japanese, Korean), after word segmentation is conducted on queries and documents. This is also true for handling queries and documents in mixtures of languages. We leave the investigation on the problem to future work.

[2] http://research.microsoft.com/en-us/collaboration/focus/cs/web-ngram.aspx.

Table 1 Examples of positive instance and negative instances

No.	Segmentation	Human label	Label
1	Beijing/seven eleven stores	×	−1
2	Beijing/seven eleven/stores	○	+1
3	Beijing seven eleven/stores	×	−1
4	Beijing seven/eleven stores	×	−1
5	Beijing/seven/eleven stores	×	−1
6	Beijing seven/eleven/stores	×	−1

The segmentation consistent with human label is regarded positive, and the others are regarded as negative

4 Our method of relevance ranking

We propose a new method for employing query segmentation in relevance ranking. It seems to be new, as far as we know, although the idea is quite simple.

4.1 General principle

In principle, using segmented queries is useful for finding relevant information. For example, when the query is "china kong movies description",[3] it is better to segment the query to "china kong/movies/description" and take "china kong" as a unit in the query representation. On the other hand, there is no guarantee that query segmentation can be performed perfectly, no matter which method is employed. It would also be necessary to retain the original query words in the query representation.

Our approach makes use of both query words and segmented query phrases as units of query representation and employs query representation in the relevance model. More specifically, given a query, our approach conducts segmentation on the query with a query segmentation method, which is described in Sect. 3. It then builds two feature vectors to represent the query. The first vector is comprised of query words and the second vector is comprised of segmented query phrases. The two feature vectors are then utilized in the relevance model. If we do not use the second vector, then our approach degenerates to the approach of not using query segmentation. If we do not use the first vector, then our approach becomes equivalent to existing methods that only use segmented query phrases.

For example suppose the query is "beijing seven eleven stores" and it is segmented into "beijing", "seven eleven", "stores" by a segmentation method. Our approach uses "beijing", "seven", "eleven", "stores" to represent the first feature vector in which the corresponding elements are one and the other elements are zero. It further uses "beijing", "seven eleven", "stores" to represent the second feature vector in which the corresponding elements are one and the other elements are zero. This is for unigrams, and we can easily extend to bigrams and trigrams, as shown in Table 3. We can see that, with query segmentation, "seven eleven" appears as a unit in the phrase based *n*-grams, which can represent a concrete meaning. This is not the case for "beijing seven" and "eleven stores" in the word based *n*-grams.

[3] China Kong is an American actor and a producer from the 1980s.

Table 2 Features of re-ranking model for query segmentation

Feature	Description
Sgmt_rank	The rank of segmentation
Sgmt_score	The score of segmentation
Sgmt_weight_on_len	The sums of weights of segments in different lengths. There are six such features, corresponding to segment lengths of 1, 2, 3, 4, 5 and beyond
Sgmt_first_weight	The weight of the first segment
Sgmt_avg_weight	The average weight of segments
Sgmt_seg#	The number of segments
Sgmt_avg_seg_len	The average length of segments
Sgmt_max_wiki_seg_len	The maximum length of segments which are Wikipedia titles
Sgmt_one_word_seg#	The number of one-word segments in segmentation
MI_max_adj_seg	The maximum value of mutual information of all adjacent segments. If the segmentation has only one segment, then this feature value equals zero
MI_max_adj_word	The maximum value of mutual information of adjacent words. If the segmentation has only one segment, then this feature value equals zero
MI_min_adj_word_in_seg	The minimum value of mutual information of the adjacent words in a segment. A segment with only one word is not considered here
Is_single_seg_word	There are words which tend to form a single word segment, such as “and”, “vs”. There are eighteen such words (and, vs, versus, compare, free, online, download, search, play, lyric, song, music, mp3, movie, why, what, when, where). If one of them occurs, then the value of the corresponding feature becomes one
Is_two_word	If the first (or last) segment has two words, then the feature value becomes one
Is_cap	If there is a subsequence of words with their first letters capitalized, then the feature value becomes one
Is_multi_plus_ones	If one segment is multi-word segment and the other segments consist of only one word in the segmentation, then the feature value becomes one
Top_split	If splitting one segment in S^h will make it equivalent to S, then the feature value is one. For i, $b_i = 1$ and $b_i^h = 0$ hold and for $j(j \neq i)$, $b_j = b_j^h$ holds
Top_merge	If merging two segments into one in S^h will make it equivalent to S, then the feature value is one. For i, $b_i = 0$ and $b_i^h = 1$ hold and for $j(j \neq i)$, $b_j = b_j^h$ holds
Top_move	If moving a break from one place to the other in S^h will make it equivalent to S, then the feature value becomes one. For i, $b_i b_{i+1} = 01(\text{or}10)$ and $b_i^h b_{i+1}^h = 10(\text{or}01)$ hold and for $j(j \neq i, i+1)$, $b_j = b_j^h$ holds
Top_same_break#	The number of identical breaks with the top segmentation
Top_same_seg#	The number of identical segments with the top segmentation

4.2 Method

We describe how to leverage query segmentation in relevance ranking, when the relevance scheme is *n*-gram BM25 (Robertson and Walker 1994), key *n*-gram model (Wang et al. 2012) and term dependency model (Bendersky et al. 2011b), three state-of-the-art models, respectively.

In web searching, web pages (documents) are represented in several fields. We consider the use of the following fields: URL, title, body, meta-keywords, meta-description, anchor texts and associated queries in search log data. Each document is represented by its fields and is indexed in the search system. In search, BM25 model, key *n*-gram model or term dependency model is created for each field of document.

We utilize *n*-gram BM25 (simply referred to as NBM25), which is a natural extension of the traditional BM25 based on unigrams. Given the query representation described above as well as the document representation in the index, we calculate the *n*-gram BM25 score for each field of the document with respect to each feature vector of the query. Therefore, each field has six BM25 scores calculated based on word-based unigrams, word based bigrams, word based trigrams, phrase based unigrams, phase based bigrams, and phrase based trigrams, respectively. To calculate a BM25 score we need to use the term frequencies of *n*-grams, document frequencies of *n*-grams, number of documents, and document length. The first three numbers can be easily obtained, but the last one can only be estimated since the document is not segmented. We use the *n*-gram document length (Wang et al. 2012) to approximate the document length. Finally, we employ LambdaMART (Burges 2010) to automatically construct the ranking model with all the *n*-gram BM25 scores of all the fields as features. Since there are seven fields and each field has six BM25 features, there are in total forty-two features in the ranking model.

When exploiting the key *n*-gram scheme (KN), we extract key unigrams, bigrams, and trigrams from the body of the web page and create an additional field with all the extracted key *n*-grams combined together, in the same way as in Wang et al. (2012). We then calculate the *n*-gram BM25 scores for all the fields including the key *n*-gram field, similar to the *n*-gram BM25 model above. We employ LambdaMART (Burges 2010) to automatically build the final ranking model with all the *n*-gram BM25 scores of all the fields as features, as proposed by Wang et al. (2012). There are a total of forty-eight features.

For the term dependency scheme (DM), we only make use of unigrams and bigrams in query representation, following the practice in Bendersky et al. (2011b). Each unigram or bigram has seven weights calculated by using other data sources such as web *n*-gram, query log, and Wikipedia. Each unigram is assigned a normalized frequency in a field of the document, and each bigram is assigned with the normalized frequency of its consecutive occurrences in a field of the document and the normalized frequency of its inconsecutive occurrences within a window of size eight in a field of the document. The product of weight and normalized frequency of a unigram or bigram is calculated. The sums of weighed normalized frequencies are calculated over the unigrams and bigrams and they are taken as features of unigrams and bigrams, respectively. Since there are seven weights and three normalized frequencies, there are twenty-one features for each field (URL, title, body, meta-keywords, meta-description, anchor texts and associated queries) and each query vector (query word based and query phase based). We again employ LambdaMART (Burges 2010) to automatically construct the final ranking model, which is similar to the coordinate descent method utilized by Bendersky et al. (2011b). In total, there are 294 features in the model.

Table 3 Example of query representation

	Original query Beijing seven eleven stores Word based *n*-grams	Query segmentation Bejing/seven eleven/stores Phrase based *n*-grams
Unigram	Beijing, seven, eleven, stores	Beijing, seven eleven, stores
Bigram	Beijing seven, seven eleven, eleven stores	Beijing seven eleven, seven eleven stores
Trigram	Beijing seven eleven, seven eleven stores	Beijing seven eleven stores

The time complexity of our method of relevance ranking is the same as the time complexity method of solely using the traditional relevance schemes of NBM25, KN and DM, because our method makes use of more features. How to improve the efficiency of the process is still an interesting topic for future research that we will not address at this time. Our method uses phrase based *n*-grams and thus one possibility is to collect the *n*-grams in advance and store them in a special index.

5 Experiments on query segmentation

In this section, we report the experimental results of query segmentation.

5.1 Experiment setup

5.1.1 Datasets

We use two public datasets in our experiments: BWC (Bergsma and Wang 2007) and Webis-QSec-10 Corpus (WQS) (Hagen et al. 2011). BWC consists of 500 queries sampled from the AOL query log dataset (Pass et al. 2006). Each query has three segmentations labeled by three annotators. WQS consists of 4,850 queries randomly sampled from the AOL the query log dataset, and each query is labeled by ten annotators.

In BWC (500 queries), there are 220 queries (44 %) for which the three annotators have an agreement, and there are 454 queries (91 %) for which at least two of the three annotators have an agreement. In WQS (4,850 queries), there are only 167 queries (3.4 %) for which all the ten annotators have an agreement, and there are 3769 queries (78 %) for which half of the annotators have an agreement.

Hagen et al. (2012) propose a break fusion method for determining the gold standard of a dataset labeled by multi labelers. We adopt the method, since it is reasonable and easy to implement. For each position between a pair of adjacent words, if at least half of the labelers insert a break, then the method also inserts a break.

Table 4 shows the distributions of segments in different lengths of the two datasets, as well as the average segment lengths. Notice that BWC favors longer segments, while WQS favors shorter segments.

5.1.2 Evaluation measures

There are five widely used measures for evaluation of the performance of a query segmentation method (Hagen et al. 2011): *segmentation accuracy* stands for the ratio of segmented queries exactly matching with the ground truth, *segment precision* stands for the ratio of correct segments among all generated segments, *segment recall* stands for the ratio of correct segments among all segments in the ground truth, *segment F-Measure* stands for the harmonic mean of the former two measures and *break accuracy* stands for the ratio of correct breaks between two adjacent words.

5.1.3 Baselines

As baselines, we choose two methods by Hagen et al. and one method by Bergsma et al.: the WBN method (Hagen et al. 2011) (denoted as WBN), the Wikipedia-Title method (Hagen et al. 2012) (denoted as WT) and the Noun-Phrase method (Bergsma and Wang 2007) (denoted as NP). WBN and WT have the best performances on the BWC and WQS datasets respectively. In our implementation, we use the Microsoft Web N-Gram Service[4] to calculate the web n-gram frequencies and query n-gram frequencies, use the Wikipedia database[5] to decide whether an n-gram matches a Wikipedia title, and use Stanford Parser[6] to collect part-of-speech information.

5.1.4 Parameter tuning

No parameter needs to be tuned in the unsupervised methods of WBN and WT. There are three parameters to be tuned in NP and four parameters to be tuned in our method.

Both the NP method and our method use SVM to train the model for query segmentation, and the tool we use is SVMLight (Joachims 2002). There are three parameters $\{c, j, b\}$[7]. We set the range of c as {0.01, 0.02, 0.05, 0.1,…, 20, 50}, the range of j as {1, 1.5, 2, 2.5,…, 8}, and the range of b as {1, 0}. We conduct a four-fold cross validation to choose the best parameter settings for our method and NP with respect to the two datasets, while taking segmentation accuracy as evaluation measure. We find that the best settings for NP are {0.1, 1, 0} for BWC and {0.05, 1, 1} for WQS, and the best settings for our method are {2, 1, 1} for BWC and {20, 2, 0} for WQS.

There is one more parameter k for our method for selecting the top k segmentations. Table 5 shows the probability of the correct segmentations appearing among the top k candidate segmentations by WBN. We can see that the probability reaches 0.94, when k is 6. Thus, we choose $k = 6$ to make a good trade-off between accuracy and efficiency in our experiments.

5.2 Results and discussions

We compare the effectiveness of our method and the three baselines on query segmentation using the two datasets.

Table 6 shows the results on the two datasets in terms of the five measures. The results of three baselines are comparable with those reported by Hagen et al. (2012). It is evident that our method of re-ranking outperforms the baselines of WBN and WT in terms of all

[4] http://research.microsoft.com/en-us/collaboration/focus/cs/web-ngram.aspx.

[5] http://en.wikipedia.org/wiki/Wikipedia:Database_download.

[6] http://nlp.stanford.edu/software/tagger.shtml.

[7] c: trade-off between training error and margin. j: cost-factor, by which training errors on positive examples outweight errors on negative examples. b: use biased hyperplane or not.

Table 4 Distributions of segments in different lengths in two datasets

Dataset	# of queries	# of words per query	% of segment length				# of words per segment
			1	2	3	4+	
BWC	500	4.3	32	55	9	4	1.9
WQS	4,850	4.1	67	26	6	1	1.4

measures except segment recall,[8], especially on the WQS dataset. All the improvements are statistically significant on two-sided sign-test $p < 0.01$. The result demonstrates that our method is effective and can enhance the accuracy of query segmentation.

We examine the weights of the linear SVM model in our method, in which a higher weight indicates a more important contribution. First, the "rank" and "score" features have the highest weights, which can ensure that the re-ranking method has similar performances as WBN and WT. Besides, the features of "weight on segment length" also have high weights. The features can capture the tendencies of segment lengths in different datasets and thus can help improve the performances in different datasets. (Recall that in BWC and WQS segments with different lengths are preferred.)

In addition, our method of re-ranking can leverage information which WBN and WT cannot, such as MI. For example, for the query "play disney channel games", both WBN and WT treat "disney channel games" as a segment, since it is also a Wikipedia title. However, it seems that the user is searching for games on the Disney Channel instead of searching for the Wikipedia page. (In fact, there is a webpage entitled "Games | Disney Channel" which can perfectly meet the need.) Therefore, the annotators label the query as "play/disney channel/games". The feature of "min MI of words in segment" can help our method to effectively deal with the problem. The adjacent words "channel games" has a small MI value, indicating that they should be separated. This is the main reason that our method works better than the baselines.

6 Experiments on relevance ranking

In this section, we report the experimental results of relevance ranking using query segmentation.

6.1 Experiment setup

6.1.1 Dataset

We conduct experiments on relevance ranking using a large data set collected from a commercial search engine. The data set contains queries, documents, and relevance judgments.

The queries of all lengths are randomly sampled from the search log of the search engine. They include single word queries for which segmentation is actually not necessary. Special symbols in the queries are replaced with blank spaces. We leverage the index of the

[8] This is because the NP method tends to generate shorter segments (Roy et al. 2012). while most human labeled segments are shorter than three words (See Table 4).

Table 5 The probability of correct segmentation appearing in the top k candidate segmentations by the method of WBN for the BWC dataset

Top N	1	2	3	4	6	8	10
Prob.	0.50	0.73	0.81	0.87	0.94	0.98	0.98

search engine in our experiments, which indexes over 88 billion documents. The labeled documents are sampled from the top ranked search results of the commercial search engine. The relevance judgments are represented at five levels including "Perfect(4)", "Excellent(3)", "Good(2)", "Fair(1)", and "Bad(0)". The relevance judgment of each query-document pair is the average relevance score by three labelers.

The whole data set is split into five subsets: The Training data set is for learning, the Validation data set is for parameter tuning and Test1, Test2, Test3 data sets for evaluation. Training, Validation and Test3 are comprised of general queries (randomly sampled from the search log), associated documents and their relevance judgments. Test1 consists of head queries (randomly sampled from the search log and with high frequencies, i.e. at least five searches a week), associated documents, and their relevance judgments. Test2 consists of tail queries (randomly sampled from the search log and with low frequencies, i.e. less than five searches a week), associated documents, and their relevance judgments. The queries of the subsets do not have any overlap with each other. Statistics on the datasets are given in Table 7.

6.1.2 Evaluation measure

To evaluate the relevance performance of different ranking models, we calculate Normalized Discounted Cumulative Gain (NDCG) (Järvelin and Kekäläinen 2000) at positions 1, 5 and 10.

6.1.3 Our method and baselines

We use seven fields of a web page (document): url, title, body, meta-keywords, meta-description, anchor texts, and associated queries in search log. In the key *n*-gram model, there is an additional key *n*-gram field.

We test the effectiveness of our method of using query segmentation in three relevance ranking schemes: BM25, key *n*-gram model and term dependency model.

1. *n*-gram BM25 model: *n*-gram BM25 is utilized, where *n*-gram includes unigram, bigram and trigram, denoted as NBM25. The *n*-gram BM25 scores are calculated separately on each field, and are regarded as individual features.

We compare the performances of BM25 and BM25F Robertson et al. (2004) based on unigrams in our experiment, and find that they are comparable with each other.[9] Therefore, we choose *n*-gram BM25 as our first baseline (Table 8).

2. Key *n*-gram model: the key *n*-gram model of Wang et al. (2012) is employed, where *n*-gram includes unigram, bigram, and trigram, denoted as KN.
3. Term dependency model: the term dependency model is employed similarly to the method of Bendersky et al. (2010), denoted as DM.

[9] We tune the parameters of BM25 and BM25F on the Validation dataset, choose the best parameters and then test the performance on Test3 dataset.

Table 6 Our method of re-ranking consistently makes improvements upon the baselines on the BWC and WQS datasets in terms of nearly all measures

Corpus	Performance measure	Algorithm			
		NP	WT	WBN	Our
BWC	Query acc	0.548	0.414	0.572	**0.602***
	Seg prec	0.651	0.538	0.692	**0.715***
	Seg rec	**0.742**	0.658	0.664	**0.700**
	Seg F	0.694	0.592	0.677	**0.707***
	Break acc	0.834	0.762	0.830	**0.848***
WQS	Query acc	0.512	0.508	0.362	**0.560***
	Seg prec	0.666	0.680	0.561	**0.710***
	Seg rec	**0.796**	0.728	0.456	0.749
	Seg F	0.726	0.703	0.503	**0.729**
	Break acc	0.783	0.784	0.680	**0.800***

Bold: The highest performance in terms of the measure

* Statistically significant improvement on all baselines (two-sided sign-test, $p < 0.01$)

Six query segmentations methods are considered in our approach. (1) The NP method (Bergsma and Wang 2007) trained with the BWC dataset, (2) the NP method trained with the WQA dataset, (3) the WBN method (no training is needed), (4) the WT method (no training is needed), (5) our re-ranking method trained with the BWC dataset, and (6) our re-ranking method trained with the WQS dataset. They are denoted as NP@BWC, NP@WQS, WBN, WT, RR@BWC, and RR@WQS.

We consider three baselines: NBM25, KN, and DM, in which no query segmentation is employed. In other words, only query words are used in the relevance ranking schemes of BM25, key *n*-gram model, and term dependency model.

6.1.4 *Parameter tuning*

We use LambdaMART to train different gradient boosted trees as relevance ranking models. There are four parameters in LambdaMART: $\{nt, nl, lr, mil\}$, which stands for number of trees, number of leaves, learning rate, and minimum instances per leaf, respectively. We choose nt from {10, 50, 100}, nl from {2, 4, 16}, lr from {0.1, 0.2, 0.3, 0.4}, and mil from {10, 50, 100} for each ranking model using the Validation data.

6.2 Main results

Table 9[10] shows the results in terms of NDCG on Test1 (head queries), Test2 (tail queries) and Test3 (random queries). We use the following notations. For example, NBM25-RR@BWC-WP means that the relevance scheme is *n*-gram BM25, the segmentation method is RR@BWC, and both query words and query phrases are utilized. NBM25-RR@WQS-WP means that the relevance scheme is *n*-gram BM25, the segmentation method is RR@WQS, and both query words and query phrases are utilized.

[10] To save space we only report the results of relevance ranking in terms of NDCG@5 and NDCG@10. The results in terms of NDCG@1, etc have similar trends.

Table 7 Datasets in relevance ranking

Data	Training (random)	Validation (random)	Test1 (head)	Test2 (tail)	Test3 (random)
# of queries	201,585	3,953	12,089	10,490	10,959
# of words per query	3.70	3.76	3.05	4.49	3.70
# of labeled pages	8,761,343	158,837	664,362	283,956	453,155
# of labeled pages per query	43.46	40.18	54.96	27.07	41.35
# of perfect per query	0.24	0.26	0.33	0.12	0.25
# of excellent per query	0.72	0.72	0.99	0.54	0.70
# of good per query	5.61	5.15	9.93	4.14	5.32
# of fair per query	12.71	12.59	20.89	9.05	12.65
# of bad per query	23.16	21.46	22.82	13.22	22.43

6.2.1 Analysis of improvements of our approach

The experimental results show that the performance of all three schemes improves in terms of all measures when our approach is employed, with most of improvements being statistically significant by two-sided *t* test $p < 0.01$, especially on Test1 and Test3. The results indicate that our approach of employing query segmentation is effective for relevance ranking.

We investigate the main reasons for the performance enhancement by our approach.

1. The longest *n*-gram in NBM25 and KN is trigram and that in DM is bigram. Thus, the models do not directly handle phrases longer than three words. For example, in the query “my heart will go on mp3 download”, “my heart will go on” is a phrase, and is not taken as a whole by the three baseline models. In contrast, there is no length constraint in query segmentation, and the query can be segmented into “my heart will go on/mp3/download”. Our approach based on query segmentation can properly use the segmentation in relevance ranking. This seems to be the main reason of the performance improvements by our approach.
2. The baseline models put equal weights on the *n*-grams of the same lengths. In fact, some *n*-grams should have higher weights because they are of more importance in queries. Query segmentation can help to reduce the impact of non-phrasal *n*-grams and enhance the impact of phrasal *n*-grams. For example, for query “beijing/seven eleven/store”, segmentation can filter out non-phrasal bigrams “beijing seven” and “seven stores”, which may have negative influence on relevance, and can retain phrasal bigrams such as “seven eleven”, which may have positive influence on relevance.
3. In DM, usually only the dependencies between word bigrams are considered, although in principle more complicated dependencies can be modeled. Thus, it is difficult for DM to handle dependencies between phrases. In contrast, our approach based on query segmentation can cope with the problem, when there exist dependencies between phrases. For example, the dependency between phrases “north korea” and “nuclear weapon” in the query of “north korea nuclear weapon in 2009” is useful for relevance ranking, and can be leveraged by our approach.

6.3 Analysis of improvements on different types of queries

We analyze the improvements on different types of queries from two perspectives: query frequency (head versus tail) and query length.

From Table 9, we can observe that our method has larger improvements on Test1 (the set of head queries) than on Test2 (the set of tail queries). One of the most important data sources for query segmentation is web *n*-gram, and therefore it is more feasible to conduct accurate segmentation on high frequency queries (head queries) than on low frequency queries (tail queries). Moreover, for tail queries, usually there are less relevant documents and the space for making improvements is also smaller.

In addition, we analyze the improvements on queries with different lengths in Test3 (the set of random queries). We take NBM25-RR@BWC-WP and NBM25 as an example. We first calculate the improvement of NBM25-RR@BWC-WP over NBM25 in terms of NDCG for each query. We then calculate average NDCG improvement for each query length. Figure 1 shows the result. From the figure we can see that shorter queries have larger improvements than longer queries, and queries of length three have the largest improvements. This result is in accordance with the result in Table 9 that improvements on

Table 8 Comparison between BM25 and BM25F based on unigrams on Test3 dataset

	NDCG@5	NDCG@10
BM25	0.4352	0.4602
BM25F	0.4340	0.4584

Table 9 The results on relevance ranking in three ranking schemes with six segmentation methods

	Test1 (head)		Test2 (tail)		Test3 (random)	
	NDCG@5	NDCG@10	NDCG@5	NDCG@10	NDCG@5	NDCG@10
NBM25 (*n*-gram BM25)	0.5427	0.5662	0.3842	0.4182	0.4466	0.4707
NBM25-NP@WQS-WP	0.5447	0.5670	0.3842	0.4183	0.4476	0.4712
NBM25-NP@BWC-WP	0.5460*	0.5700*	0.3862	0.4184	0.4486	0.4720
NBM25-WT-WP	0.5464*	0.5713*	0.3854	0.4203	0.4488	0.4721
NBM25-WBN-WP	0.5493*	0.5762*	0.3873	0.4213*	0.4506*	0.4748*
NBM25-RR@WQS-WP	0.5524*	0.5775*	0.3916*	0.4239*	0.4515*	0.4748*
NBM25-RR@BWC-WP	**0.5582***	**0.5820***	**0.3921***	**0.4245***	**0.4522***	**0.4750***
KN	0.5578	0.5838	0.4023	0.4390	0.4687	0.4920
KN-NP@WQS-WP	0.5582	0.5844	0.4025	0.4387	0.4707	0.4950*
KN-NP@BWC-WP	0.5600*	0.5868*	0.4030	0.4385	0.4712	0.4956*
KN-WT-WP	0.5709*	0.5974*	**0.4067***	**0.4407**	0.4804*	0.5029*
KN-WBN-WP	0.5615*	0.5695*	0.4033	0.4399	0.4774*	0.4992*
KN-RR@WQS-WP	0.5633*	0.5906*	0.4055*	0.4399	0.4786*	0.5016*
KN-RR@BWC-WP	**0.5771***	**0.6039***	0.4055*	**0.4407**	**0.4809***	**0.5037***
DM	0.5771	0.6016	0.3782	0.4125	0.4780	0.5007
DM-NP@WQS-WP	0.5801*	0.6056*	0.3784	0.4130	0.4791	0.5017
DM-NP@BWC-WP	0.5883*	0.6115*	0.3795	0.4145	0.4798	0.5031*
DM-WT-WP	0.5916*	0.6162*	0.3800	0.4139	0.4862*	0.5069*
DM-WBN-WP	0.5913*	0.6174*	0.3804	0.4135	0.4876*	0.5074*
DM-RR@WQS-WP	0.5969*	0.6201*	0.3816*	0.4141	0.4889*	0.5090*
DM-RR@BWC-WP	**0.6014***	**0.6254***	**0.3828***	**0.4173***	**0.4898***	**0.5104***

Bold: The highest performance for the scheme on the dataset

* Statistically significant improvement on baseline (two sided *t* test, $p < 0.01$)

head queries are larger than tail queries, because head queries tend to be short and tail queries tend to be long.

6.4 Comparison with using only segmented query phrases

Our method makes use of both query words and query phrases (i.e., it creates two query vectors). This is also one of the major differences between our method and existing methods.

In this section, we make comparisons between our method and the alternative method which only uses segmented query phrases, again in the three schemes.

Table 10 shows the results in terms of NDCG. We use the following notations. For example, NBM25-RR@BWC-P means that the relevance scheme is *n*-gram BM25, the segmentation method is RR@BWC, and only query phrases are utilized. NBM25-RR@WQS-P means that the relevance scheme is *n*-gram BM25, the segmentation method is RR@WQS, and only query phrases are utilized.

We find that most alternative methods perform worse and even statistically significantly worse than the baselines, except for several measures. Especially on Test2, all the alternative methods are worse than the baseline methods. All the alternative methods perform statistically significantly worse than the methods which utilize both query words and query phrases (our approach).

We find that there are several reasons.

1. It appears that simply replacing query words with query phrases after query segmentation is not always helpful, and sometimes it is even harmful. For example, there is an NDCG loss for query "cambridge university students count". The query segmentation result is "cambridge university/students/count". When "cambridge university" is combined together, it will not match "Cambridge" in a webpage, which also means Cambridge University.
2. Incorrect segmentation is inevitable. Incorrect segmentation includes incorrect splitting of phrases such as "my heart/will/go on/mp3/download" and incorrect merging of words such as "beijing seven eleven/stores". Both results can increase the number of incorrect phrase *n*-grams and reduce the number of correct phrase *n*-grams.
3. Test2 consists of tail queries. It is difficult to conduct segmentations on such queries, because less information is available for the queries. Therefore, solely using phrases would not produce good performance in such case.

The experiment results demonstrate that it is better to make use of both query words and query phrases when employing query segmentation in relevance ranking.

6.5 Comparison among segmentation methods

The results in Table 9 also show differences in performance by employing different query segmentation methods in our approach of relevance ranking on Test1, Test2 and Test3.

The six segmentation methods (i.e. NP, WT, WBN and RR on BWC and WQS) have different relevance results. Among the segmentation methods, our method of re-ranking trained with the BWC dataset (*-RR@BWC-WP) achieves the best performances on nearly all three ranking schemes with three test data sets. One exception is WT on Test2 Data which has the best performance.

There are two main differences among the six segmentation methods. The first is different segmentation accuracies. Some methods tend to generate more accurate segments which match human labels better. The second is different segment lengths. Hereafter, we conduct analysis to see how segmentation accuracy and average segment length affect relevance ranking performance.

6.5.1 Impact of segmentation accuracy

We investigate whether higher query segmentation accuracy can lead to higher relevance ranking performance. Figure 2 shows different relevance ranking results of four basic segmentation methods. Column 1 shows the impact of segmentation accuracy on the BWC data, column 2 shows the impact of segmentation accuracy on the WQS data. Here we only

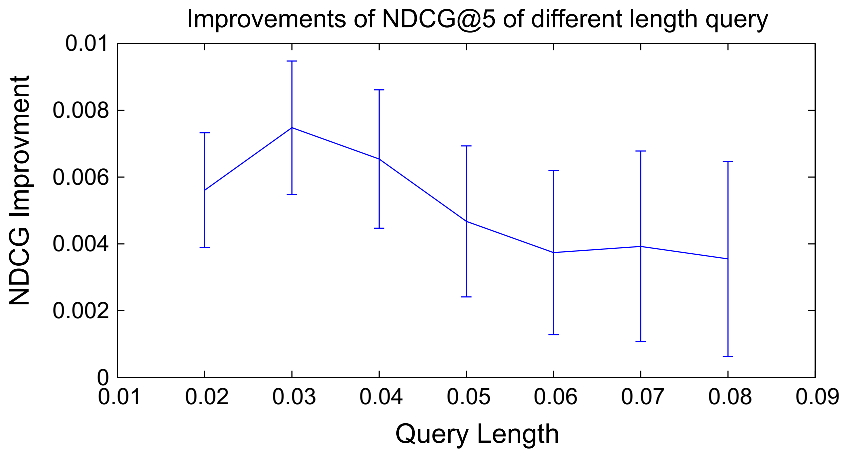

Fig. 1 Improvements of queries of different lengths from NBM25 to NBM25-RR@BWC-WP in terms of NDCG@5. We use standard *error bars* in this figure

use segmentation accuracy as measure because the other measures have similar trends. Note that in Fig. 2, the vertical axis represents deviation of NDCG scores, which is defined as NDCG@i-Avg(NDCG@i). We use it to highlight the gradual change of relevance ranking performance in terms of NDCG with respect to query segmentation accuracy.

We can see that our query segmentation methods (RR@BWC and RR@WQS), which have the highest accuracies, achieve the highest relevance ranking performances. On the other hand, relevance ranking performance does not always increase when more accurate segmentation methods are employed, which is consistent with the observations of Hagen et al. (2012), Roy et al. (2012). For example, NP is a more accurate segmentation method than WT on WQS, however, the relevance ranking performances of NP@WQS are worse than those of WT.

There are three reasons for the inconsistency between query segmentation accuracy and relevance ranking performance.

First, the queries in BWC and WQS are sampled from search log under certain conditions (Hagen et al. 2011). For example, in BWC, it is required that queries should only consist of determiners, adjectives, and nouns. These make the queries in BWC and WQS different from the queries in relevance ranking, which are randomly sampled. Therefore, more accurate segmentation on BWC and WQS does not necessarily mean better relevance ranking.

Second, segmentation methods may suffer from over fitting. For example, NP makes use of many carefully designed features, such as whether the token is “the” or “free”, because the authors find that “the” and “free” in the data are likely to be split into single segments. Although these features are helpful for query segmentation, some of them may not help relevance.

Third, in some special cases, segmentation accuracy is not the most important factor for relevance ranking. For example, from Table 9 we can see that WT has the best performance for KN on Test2 which contains only tail queries, although WT is not the best segmentation method on both BWC and WQS. We look into the extracted key *n*-grams of documents and find that 39.6% *n*-grams are Wikipedia titles. The key *n*-gram model makes use of extracted unigrams, bigrams, and trigrams. WT tends to create segmentations consisting of Wikipedia titles and one-word segments. As a result, it is likely that the query representations of WT and the key *n*-gram fields match well in relevance ranking.

Table 10 The results on relevance ranking when only segmented query phrases are used in representation

	Test1(head)		Test2(tail)		Test3(random)	
	NDCG@5	NDCG@10	NDCG@5	NDCG@10	NDCG@5	NDCG@10
NBM25 (*n*-gram BM25)	**0.5427**	**0.5662**	**0.3842***	**0.4182**	**0.4466**	**0.4707***
NBM25-NP@WQS-P	0.5353	0.5592	0.3739	0.4078	0.4386	0.4641
NBM25-NP@BWC-P	0.5257	0.5505	0.3568	0.3926	0.4271	0.4527
NBM25-WT-P	0.5362	0.5602	0.3615	0.3952	0.4295	0.4543
NBM25-WBN-P	0.5362	0.5629	0.3723	0.4044	0.4296	0.4570
NBM25-RR@WQS-P	0.5378	0.5629	0.3800	0.4160	0.4388	0.4631
NBM25-RR@BWC-P	0.5423	**0.5662**	0.3809	0.4162	0.4433	0.4665
KN	**0.5578**	**0.5838**	**0.4023***	**0.4390***	**0.4687***	**0.4920***
KN-NP@WQS-P	0.5338	0.5622	0.3905	0.4192	0.4235	0.4627
KN-NP@BWC-P	0.4774	0.5079	0.3595	0.4012	0.4077	0.4372
KN-WT-P	0.4964	0.5315	0.3822	0.4216	0.4155	0.4469
KN-WBN-P	0.4886	0.5240	0.3633	0.4029	0.4065	0.4378
KN-RR@WQS-P	0.4891	0.5242	0.3759	0.4145	0.4164	0.4458
KN-RR@BWC-P	0.5197	0.5483	0.3884	0.4271	0.4170	0.4478
DM	**0.5771***	0.6016	**0.3782**	**0.4125**	**0.4780***	**0.5007***
DM-NP@WQS-P	0.5697	0.5947	0.3578	0.3964	0.4665	0.4857
DM-NP@BWC-P	0.5580	0.5815	0.3485	0.3840	0.4644	0.4815
DM-WT-P	0.5605	0.5840	0.3497	0.3849	0.4674	0.4855
DM-WBN-P	0.5606	0.5848	0.3634	0.3980	0.4688	0.4898
DM-RR@WQS-P	0.5739	**0.6026**	0.3753	0.4095	0.4695	0.4908
DM-RR@BWC-P	0.5704	0.6004	0.3774	0.4110	0.4690	0.4903

Bold: The maximum value of the scheme for the dataset

* Baseline is statistically significantly better than alternative methods (two-sided t test, $p < 0.01$)

6.5.2 *Impact of average segment length*

We investigate the impact of segment lengths on the performances of relevance ranking. Figure 3 shows the distributions of segment lengths by the four methods. NP@BWC, NP@WQS and WT tend to create short segments, while RR@BWC, RR@WQS and WBN tend to create long segments. There are two reasons for the different segment distributions: different characteristics of segmentation methods and different training sets. First, it is likely for NP to generate shorter segments because it only considers the frequencies of two adjacent words (Bergsma and Wang 2007), without considering whether the two words is a part of Wikipedia title. The method tends to break entity names with low frequencies into single words, such as “xbox one”, “final fantasy 7”. For WT, it is likely to treat Wikipedia titles as segments and make the rest one-word segments, yielding fine-grained segmentations. In contrast, WBN and our re-ranking method are likely to merge Wikipedia titles as well as the consecutive words with high frequencies as segments, yielding coarse-grained segmentations. Second, the two training datasets, BWC and WQS, have different average segment lengths (see Table 4), BWC has longer segments, while WQS has shorter segments. Therefore, the lengths of segments created by the supervised segmentation methods

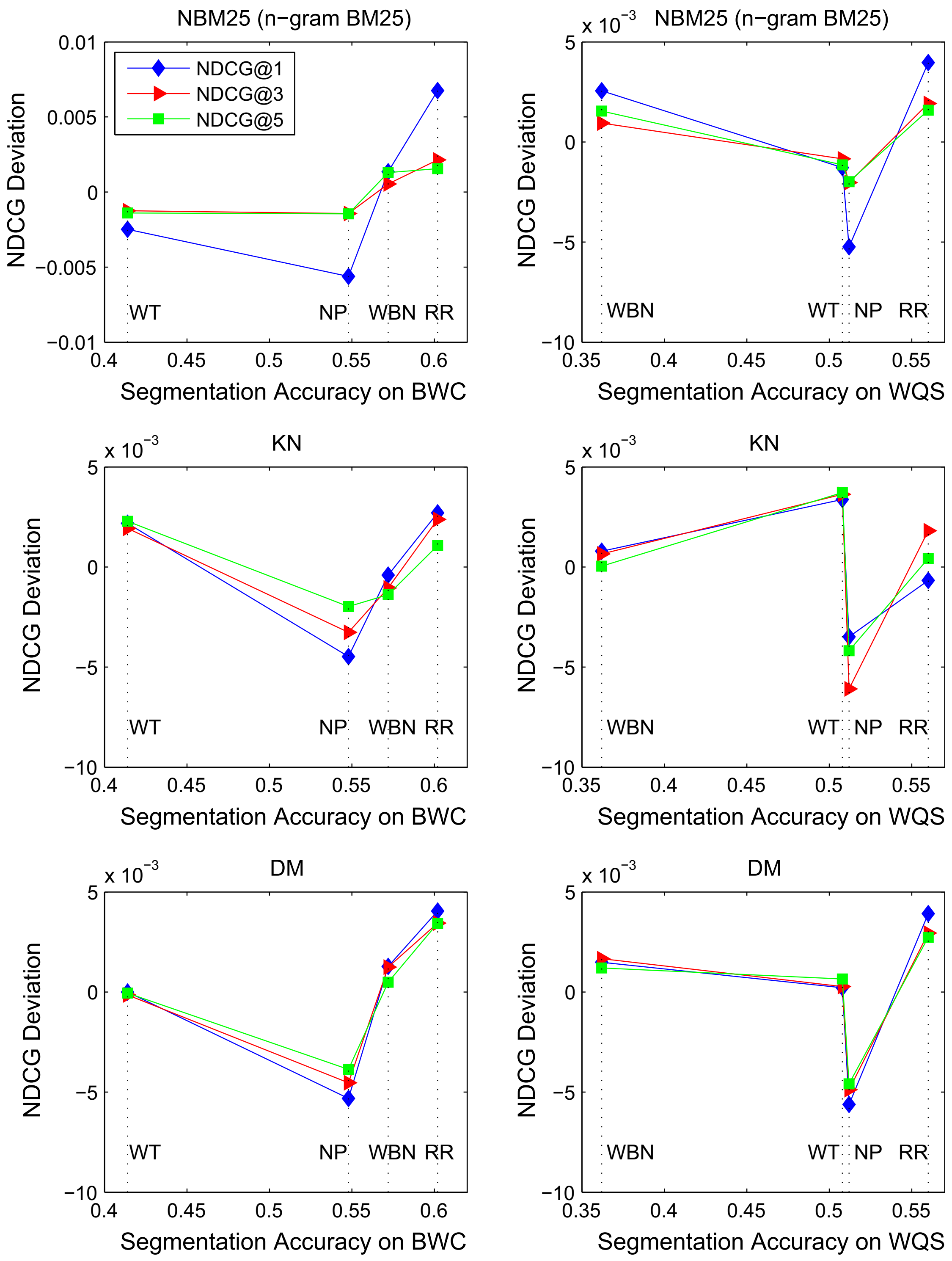

Fig. 2 The impact of query segmentation accuracy on relevance ranking performance

of NP and RR are quite different. As a result, NP@BWC and RR@BWC generate longer segments than NP@WQS and RR@WQS respectively.

We observe a tendency for coarse-grained segmentation to outperform fine-grained segmentation. It is easy to understand that using a fully segmented query is equivalent to using all query words. Therefore, the performance of using the former in relevance ranking will be the same as that of using the latter. That is why NP@WQS and NP@BWC do not

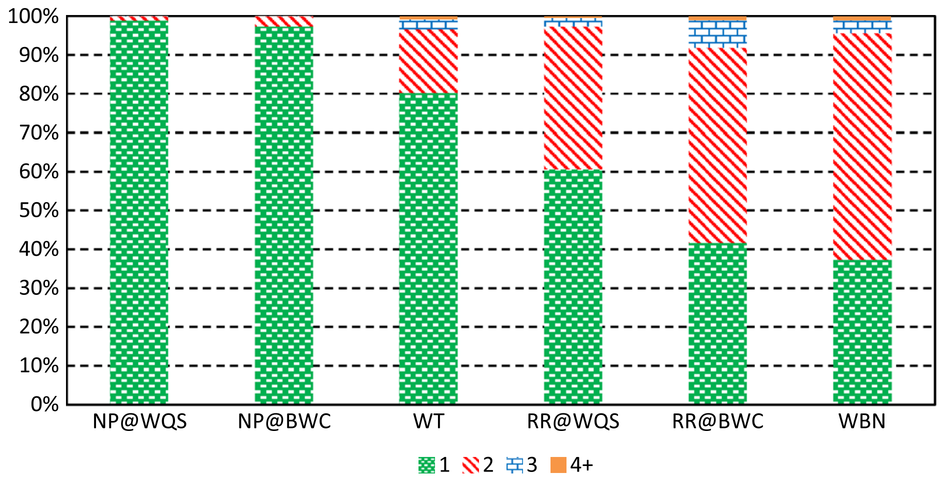

Fig. 3 Distributions of segment lengths by different segmentation methods

have much space to improve. In contrast, the coarse-grained segmentation methods, which combine those highly associated words into a segment, can add useful information for matching between query and document, and thus generate higher relevance ranking accuracy. However, if the segmentation method gives too coarse segmentations, such as WBN, it also hinders the improvements. Figure 4 shows the relation between deviation of NDCG scores and average segment length.

7 Conclusions and future work

In this paper, we have proposed a new approach to enhancing search relevance by query segmentation, including a method of query segmentation and a method of using query segmentation result in search. The former method first generates top *k* candidates for query segmentation with a generative model and then re-ranks the candidates with a discriminative model. The latter method takes both the original query words and the segmented query phrases as units of query representation. We have empirically studied the effectiveness of the proposed approach with the relevance schemes of *n*-gram BM25, key *n*-gram model, and term dependency model, using large datasets in web search. We have found that our approach can statistically significantly improve relevance ranking.

There are certainly a number of directions for future research with regard to query segmentation in search. (1) Current methods of query segmentation including our method do not heavily rely on natural language analysis. In theory, a sophisticated and robust natural language analysis on query can generate better results. How to perform such analysis is a challenging issue and needs further study. (2) It is still difficult to carry out accurate segmentation on tail queries. One possible solution is to leverage more web data (not only Wikipedia) to address the problem. How to realize the goal is certainly an interesting research topic. (3) Although the time complexity of the re-ranking approach is not high, it is definitely important to improve efficiency for online query processing. One question is whether it is possible to develop a special index to store the data (e.g., *n*-grams) necessary for processing. (4) Query segmentation is hard in a single search. However, if the previous searches in the same session are given, then more accurate segmentation may be conducted by leveraging the information. In other words, context aware segmentation may be more effective, which deserves some investigation.

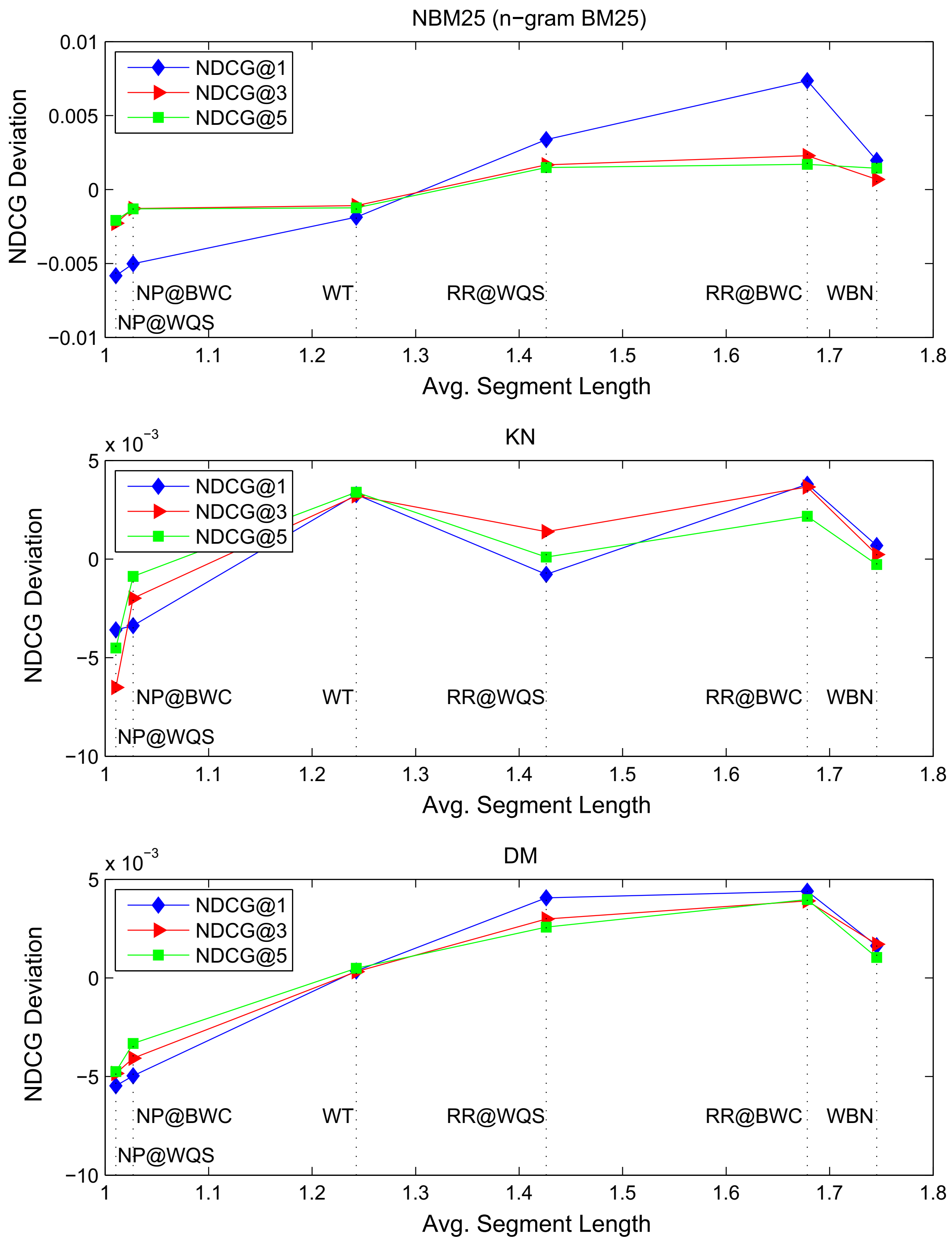

Fig. 4 The impact of average segment length on relevance ranking

References

Bendersky, M., Croft, W. B., & Smith, D. A. (2009). *Two-stage query segmentation for information retrieval*. In SIGIR, pp. 810–811.

Bendersky, M., Metzler, D., & Croft, W. B. (2010). *Learning concept importance using a weighted dependence model*. In WSDM, pp. 31–40.

Bendersky, M., Croft, W. B., & Smith, D. A. (2011a). *Joint annotation of search queries*. In ACL, pp. 102–111.

Bendersky, M., Metzler, D., & Croft, W. B. (2011b). *Parameterized concept weighting in verbose queries*. In SIGIR, pp. 605–614.

Bergsma, S., & Wang, Q. I. (2007). *Learning noun phrase query segmentation*. In EMNLP-CoNLL, pp. 819–826.

Brenes, D. J., Gayo-Avello, D., & Garcia, R. (2010). *On the fly query entity decomposition using snippets*. CoRR abs/1005.5516.

Burges, C. J. C. (2010). *From ranknet to lambdarank to lambdamart: An overview*. Microsoft Research Technical Report MSR-TR-2010-82.

Hagen, M., Potthast, M., Stein, B., & Bräutigam, C. (2011). *Query segmentation revisited*. In WWW, pp. 97–106.

Hagen, M., Potthast, M., Beyer, A., & Stein, B. (2012). *Towards optimum query segmentation: In doubt without*. In CIKM, pp. 1015–1024.

Huang, J., Gao, J., Miao, J., Li, X., Wang, K., Behr, F., & Giles, C. L. (2010). *Exploring web scale language models for search query processing*. In WWW, pp. 451–460.

Järvelin, K., & Kekäläinen, J. (2000). *Ir evaluation methods for retrieving highly relevant documents*. In SIGIR, pp. 41–48.

Joachims, T. (2002). *Learning to classify text using support vector machines—Methods, theory, and algorithms*. Berlin: Springer.

Jones, R., Rey, B., Madani, O., & Greiner, W. (2006). *Generating query substitutions*. In WWW, pp. 387–396.

Lafferty, J. D., & Zhai, C. (2001). *Document language models, query models, and risk minimization for information retrieval*. In SIGIR, pp. 111–119.

Li, H. (2011a). Learning to rank for information retrieval and natural language processing. *Synthesis Lectures on Human Language Technologies*. Morgan and Claypool Publishers.

Li, H. (2011b). A short introduction to learning to rank. *IEICE Transactions*, *94–D*(10), 1854–1862.

Li, Y., Hsu, B. J. P., Zhai, C., & Wang, K. (2011). *Unsupervised query segmentation using clickthrough for information retrieval*. In SIGIR, pp. 285–294.

Liu, T. Y. (2011). *Learning to rank for information retrieval*. Berlin: Springer.

Metzler, D., & Croft, W. B. (2005). *A markov random field model for term dependencies*. In SIGIR, pp. 472–479.

Ogilvie, P., & Callan, J. P. (2003) *Combining document representations for known-item search*. In SIGIR, pp. 143–150.

Pass, G., Chowdhury, A., & Torgeson, C. (2006). *A picture of search*. In Infoscale, p. 1.

Ponte, J. M., & Croft, W. B. (1998). *A language modeling approach to information retrieval*. In SIGIR, pp. 275–281.

Risvik, K. M., Mikolajewski, T., & Boros, P. (2003). *Query segmentation for web search*. In WWW (Posters).

Robertson, S. E., & Walker, S. (1994). *Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval*. In SIGIR, pp. 232–241.

Robertson, S. E., Zaragoza, H., & Taylor, M. J. (2004). *Simple bm25 extension to multiple weighted fields*. In CIKM, pp. 42–49.

Roy, R. S., Ganguly, N., Choudhury, M., & Laxman, S. (2012). *An ir-based evaluation framework for web search query segmentation*. In SIGIR, pp. 881–890.

Tan, B., & Peng, F. (2008). *Unsupervised query segmentation using generative language models and wikipedia*. In WWW, pp. 347–356.

Wang, C., Bi, K., Hu, Y., Li, H., & Cao, G. (2012). *Extracting search-focused key n-grams for relevance ranking in web search*. In WSDM, pp. 343–352.

White, R. W., & Morris, D. (2007). *Investigating the querying and browsing behavior of advanced search engine users*. In SIGIR, pp. 255–262.

Xu, J., Li, H., & Zhong, C. (2010). *Relevance ranking using kernels*. In AIRS, pp. 1–12.

Yu, X., & Shi, H. (2009). *Query segmentation using conditional random fields*. In KEYS, pp. 21–26.

Zhang, C., Sun, N., Hu, X., Huang, T., & Chua, T. S. (2009). *Query segmentation based on eigenspace similarity*. In ACL/IJCNLP (Short Papers), pp. 185–188.