Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

Personalized text snippet extraction using statistical language models

Qing Li[a,*], Yuanzhu Peter Chen[b]

[a]School of Economic Information Engineering, Southwestern University of Finance and Economics, China
[b]Department of Computer Science, Memorial University of Newfoundland, Canada

A R T I C L E   I N F O

Article history:
Received 17 September 2008
Received in revised form 14 May 2009
Accepted 10 June 2009

Keywords:
Text snippet extraction
Personalization
Language model
Information retrieval
Natural language processing
Pattern discovery
Hidden Markov Model

A B S T R A C T

In knowledge discovery in a text database, extracting and returning a subset of information highly relevant to a user's query is a critical task. In a broader sense, this is essentially identification of certain personalized patterns that drives such applications as Web search engine construction, customized text summarization and automated question answering. A related problem of text snippet extraction has been previously studied in information retrieval. In these studies, common strategies for extracting and presenting text snippets to meet user needs either process document fragments that have been delimitated a priori or use a sliding window of a fixed size to highlight the results. In this work, we argue that text snippet extraction can be generalized if the user's intention is better utilized. It overcomes the rigidness of existing approaches by dynamically returning more flexible start–end positions of text snippets, which are also semantically more coherent. This is achieved by constructing and using statistical language models which effectively capture the commonalities between a document and the user intention. Experiments indicate that our proposed solutions provide effective personalized information extraction services.

1. Introduction

Information systems are becoming an integral part of everyday life as we accumulate more and more knowledge that are often represented as texts. An important function of information processing in such systems is to facilitate search of useful information in textual corpus. This service is parameterized as a user query so that the search can be personalized. It is essentially discovering patterns in a knowledge base relevant to the query. Specifically, after a user issues a query, the system retrieves a set of text documents from the corpus which are deemed relevant to the query. Typically, there is also a rank associated with each returned document. In order for the user to better utilize these documents, they need to be "shortened" in a way that reflects the characteristics of the query. This step is referred to *personalized text snippet extraction* in this article. A straightforward approach would be to return one or multiple parts of a document that contain as much of the query as possible. This is adopted by most current Web search engines, such as Google, when presenting results to users. In this work, we focus on more intelligent methods to identify flexible document snippets.

Personalized text snippet extraction finds a wider scope of applications than Web search engine design. Its generality is due to the various potential origins of a user query. When a query is a string of text submitted by a Web search engine user, our method can be utilized by search engine construction. However, a query can carry other forms of information. For example, a digital library can present annotated documents to a user where parts of the documents are highlighted according to this particular user's preference encoded as a query. When used in automated document summarization, our method can provide more personalized document summaries tailored to the needs of different users. It is also useful to provide more accessible Web user interface to visually impaired users. Such a special group of users make extensive use of screen readers, such as JAWS and Window-Eyes, which convert texts to audible signals. Screen readers are not easy to follow without the capability of intelligently summarizing the documents being processed. Their comprehensibility can be significantly improved when personalized summaries are directed to them rather than the full documents themselves.[1] In the rest of this article, we use the term "query" equivalently to information that represents user preferences.

In our approach, we first use a document filter to retrieve a set of documents from the corpus that are relevant to a query. Since this step is orthogonal to the subsequent text snippet extraction, we can use any good such filter to retrieve documents. Typically, a

* Corresponding author.
E-mail addresses: kooliqing@gmail.com, liq_t@swufe.edu.cn (Q. Li).

[1] The solutions described in this article are in fact a central component of OASIS, which is a Web front end of a digital library for blind children of the State of Arizona (http://aria.asu.edu/oasis/).

0031-3203/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2009.06.003

document filter conducts a certain similarity comparison between the query and a subset of documents in the corpus which contain at least part of the query. Next, we construct a pair of statistical language models, one pertinent to the query and the other independent of the query. Using these two models, we can calculate two probabilities of each word in the relevant document being generated by either model. We treat a relevant document as a sequence of words, each associated with these two probabilities. Thus, from every relevant document, we are able to extract one or more snippets that are deemed to best satisfy the user's intention. In this work, we resort to two methods for snippet extraction. One directly utilizes the probability sequence associated with the document, and the other processes a Markov model hidden behind these observed word generation probabilities. When understood from the context, we use "documents" and "relevant documents" interchangeably.

In this article, we utilize relevance language models to extract text snippets in response to user needs. It improves existing approaches, which more or less formulate a query as a word frequency vector and compare it with the frequency vector of document fragments. In particular:

- Two language models are constructed for the relevant documents specifically for the given query. Using these models, we calculate the precise start–end positions of each snippet to return. This flexibility allows our method to produce more coherent text snippets personalized to different users.
- Furthermore, our approach looses the implicit assumption made in the existing work that all words in the query carry the same weight when compared to the frequency vector of a document fragment. This, however, is not reasonable because different words can have significantly different occurrence probabilities in any language model. Our method dispenses with such an assumption via heavier investigation of statistical language models.

In Section 2, we first provide some background information. The design details of our solutions are described in Section 3. We have conducted experimental evaluation, and comparisons to previous work indicate that our method yield higher user satisfaction (Section 4). In Section 5, we conclude our work with further discussion and outlook for future research.

2. Background

Text snippet extraction returns one or more fragments of a document that are considered most related to a given query. When a text corpus is constructed or updated, there can be a preprocessing step that partitions each document into segment [5,19,26]. In this case, when a query is issued and a set of relevant documents are returned, one or more of these segments generated a priori are returned as extraction. Thus, the start–end positions of a snippets are fixed [1,6]. Alternatively, if no such preprocessing is involved, each relevant document is extracted on the fly and a snippet can have arbitrary start–end positions [9,10,16,18].

During the preprocessing of a document, there can be a variety of options as of how to partition a document. For example, a document can be cut into non-overlapped segments. This leads to a potential problem that the best text snippet can cross the boundary of two adjacent segments. To address this, a partition can also create overlapped segments. When determining the length of a segment, be it overlapped or not, it can also have a fixed size as part of the system configuration, or a variable length using semantic text partitioning. These two aspects can be somehow combined to create, say, overlapped variable-length text snippets.

In contrast, text snippet extraction can also be conducted without document partitioning a priori. Specifically, the start–end positions of a snippet are determined in response to a user query. The Google search engine takes such an approach. When a set of documents have been identified as relevant by the engine and presented in the search result page, each document is represented by a few sliding windows containing (part of) the query. Google uses a set of heuristics, such as document and grammatical structures, to regulate the size of the sliding windows and where they start and end. In this work, we further investigate using more intelligent methods to generate variable-length text snippets that make heavy use of the characteristics of the user query.

In a broader context, a related problem is the creation of a summary for a given document [3,4,29]. In document summarization, while being significantly shorter than the input documents, the output needs to represent the content of the document well. Most existing summarization schemes [3,22,30,15] create summaries by combining several important words or sentences from the original documents. A large amount of work in this area take this even further by creating a single summary from multiple documents [22,15]. Instead of capturing a summary using several sentences within a document or across multiple documents, some researchers study composition of headlines, which are very short phrases, typically not more than 10 words, from news stories or e-mails [29,31]. While summaries and headlines are useful to help users skim through long documents to assess their usefulness, they are often insensitive to the needs of particular users. That is, while a single summary or headline will be a reasonable gist for a given document, this is not sufficient for snippet extraction, where the output should be tailored to the user needs. On the other hand, document summarization systems may be further informed by snippet extraction tools for personalized services. In particular, once document segments relevant to a user query have been identified, they can be fed to a summarizer to meet this user's specific needs. Another related domain is question answering (QA) [2,17,23]. A QA system looks for parts of documents as answer to a submitted question. Text snippet extraction can also be used to improve the performance of QA systems by narrowing down the scope of materials that the system uses to compose answers.

Our solutions to text snippet extraction are based on statistical language models [13,20,25]. A *statistical language model* is a probability distribution that captures the statistical regularities (e.g., word or phoneme distribution) of a natural language. If we confine the use of this language within a domain under consideration, such a model is called a *relevance language model* [13]. When using a statistical language model in text retrieval, a document can be represented as a bag of words and their distribution. That is, a given piece of text can be regarded as a sample drawn according to the underlying word probability distribution of the model. Thus, language models are also referred to as "generative" models. A common theme in this area is to estimate the likelihood that a query and a document could have been generated by the same language model. Recently, language models have been applied to various research problems in information retrieval, including topic detection and tracking [12], information recommendation [14] and question answering [32].

3. Design details

The design scheme is sketched in Fig. 1. It takes the entire set of documents and a user query as input and generates personalized text snippets as output. The user query is fed to a retriever for a subset of relevant documents. Using the relevant subset and the entire set, we generate a relevant and irrelevant language modes, respectively. With these models, we are able to output personalized text snippets using word sequence analysis or hidden Markov model.

To process the user query, we construct two language models, one relevant to the query, denoted by M_r, and the other irrelevant,

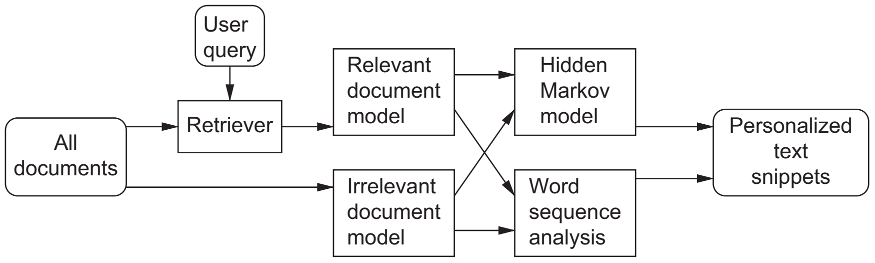

Fig. 1. Design scheme.

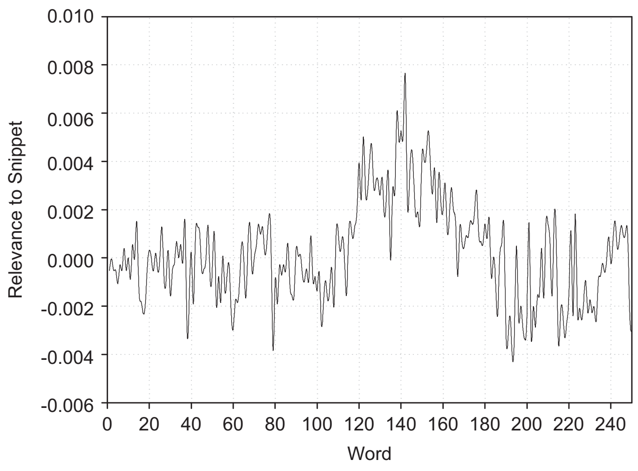

Fig. 2. Document plotted as a sequence of words.

```
Input : Document $d$, query $q$, relevance threshold, $\theta$
Output : Relevant snippet set $\mathcal{S}$
Step 1 : Plot $d$ as word sequence and store in a queue
1.1. $a = \emptyset$
1.2. for each word $w \in d$ do
        $P(w|M_r) = \text{getRMProb}(w, q)$
        $P(w|M_{\bar{r}}) = \text{getIRMProb}(w, q)$
        $c = P(w|M_r) - P(w|M_{\bar{r}})$
        $a$.add($c$)
Step 2: Identification of relevant snippets
2.1. $\mathcal{S} = \emptyset$, $S_{cur}$ = null, $S_{prev}$ = null
2.2. $a$.smooth()
2.3. $S_{cur} = a$.getMaxInterval() /* get the maximum contiguous interval */
2.4 $H_{cur} = S_{cur}$.sum()
2.5. do {
        $S_{prev} = S_{cur}$
        $S_{cur} = a$.getMaxInterval()
        $H_{prev} = H_{cur}$
        $H_{cur} = S_{cur}$.sum()
        if $\left(\frac{H_{cur}}{H_{prev}} < \theta\right)$
            break;
        $a$.maskSubqueue($S_{cur}$)
        $\mathcal{S}$.add($S_{cur}$)
}
2.6. Return $\mathcal{S}$
```

Fig. 3. Snippet extraction via word sequence analysis.

denoted by $M_{\bar{r}}$. Specifically, the relevance model M_r is a probability distribution function of each word being generated by the natural language *relevant to the query*. In contrast, the irrelevance model $M_{\bar{r}}$ is a probability distribution function of each word being generated by the complement of the above language, i.e., the part of the language considered irrelevant to the query. When written in the form of conditional probability as a convention in natural language processing [24,28], each word w can be generated by M_r with probability $P(w|M_r)$. In addition, it can also be generated by $M_{\bar{r}}$ with probability $P(w|M_{\bar{r}})$. We defer the description of how we obtain these probabilities to Section 3.3.

We represent each relevant document d as a sequence of words $\{w_1, w_2, \ldots, w_n\}$, where n is the number of words in d. There are two ways to treat such a sequence. On one hand, any word subsequence can resemble the query to a varying degree, which can be quantified by the probabilities of each word in the subsequence being generated by M_r and $M_{\bar{r}}$. On the other hand, the word sequence can also be considered as output of a generator that can be modeled by a two-state Markov Chain. At each point, the generator is either in a relevant or irrelevant state. Essentially, the Markov chain and the observed sequence of words form a hidden Markov model (HMM), whose output probabilities are regulated by M_r and $M_{\bar{r}}$. These two approaches are described in the sequel. In both cases, we assume that each document contains at least one relevant subsequence.

3.1. Snippet extraction via word sequence analysis

We can plot the word sequence along the x-axis and represent each word's relevance to the query as a value on the y-axis (Fig. 2). Thus, a relevant text snippet corresponds to a contiguous portion of the plotting with a relatively large value. Here, for each word w in document d, we use the value

$$P(w|M_r) - P(w|M_{\bar{r}})$$

to quantify the degree that w is relevant to the query. The subtraction of the probability of w being generated by the irrelevance model is a type of normalization to offset its "background likelihood" in the language. Note that there can be other forms of similar normalization, but our experiments (Section 4) show that this simple adjustment is very effective. This probability difference falls in $[-1, 1]$. When w is considered relevant to the query, this quantity tends to be positive; while w is irrelevant, it tends to be negative. Therefore, we identify a contiguous interval $[s, t]$ on the x-axis that achieves the following maximum:

$$\max_{1 \le s \le t \le n} \sum_{i=s}^{t} (P(w|M_r) - P(w|M_{\bar{r}})).$$

That is, the relevant text snippet is $\{w_s, w_{s+1}, \ldots, w_t\}$.

We describe the entire algorithm as the pseudocode in Fig. 3. There are two notes that we would like to make. First, there is a smoothing process before the actual calculation of the maximum in order to remove the very-short-term fluctuation in the plotting. Second, the algorithm has been extended to extract multiple relevant text snippets from a single document. To do that, iteratively extract and mask the returned snippet from the word sequence. For each snippet, we record the sum of the y values of all words in it. This is repeated until the ratio of such values between two consecutive iterations falls below a threshold θ.

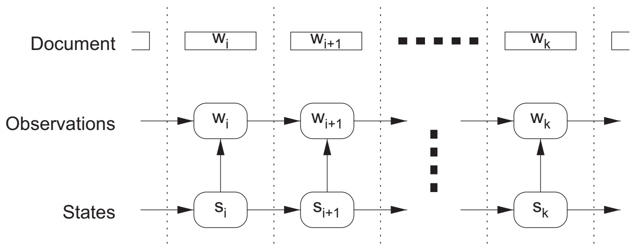

Fig. 4. Representation of a document as HMM.

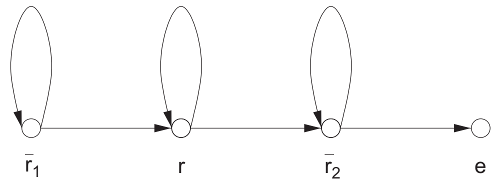

Fig. 5. Hidden state transitions in HMM.

3.2. Snippet extraction using HMM

In contrast, snippet extraction via HMM models each document as an observable symbol chain, behind which there is a hidden state chain revealing whether the corresponding word is relevant to the query or not (Fig. 4). That is, a given document can be represented as a sequence of words, each of which is generated by either the relevance or irrelevance model. The hidden Markov chain has two states associated with these two language models. Therefore, our goal is to determine how the chain transits between these two states and how each state generates a word in the given document. In essence, this is the well-known HMM evaluation problem [21].

In order to extract a relevant text snippet from the document, we expect that the Markov chain starts with the irrelevant state, transits to the relevant state to generate the snippet and then returns to the irrelevant state before it terminates. Since the irrelevant states before and after the snippet must be differentiated to avoid loops, we extend the two-state HMM to a three-state HMM by breaking the irrelevant state into two, with one preceding the relevant state and the other succeeding it. The state transition of the Markov chain is depicted in Fig. 5. In the figure, $\bar{r}_1$ and $\bar{r}_2$ are the two types of irrelevant states and r is the relevant state. Here, we focus on the scenario of extracting one text snippet from the document. This scheme is extended to produce multiple snippets towards the end of this subsection.

Next, the snippet boundary detection task is to locate the two transitions that leads the Markov chain into and out of r. That is, to find the subsequence of the hidden states that are most likely to have generated the observed sequence of output symbols (i.e., words). We denote all possible state transitions that follow the pattern in Fig. 5 by $\mathscr{T}$. In addition, for any $T \in \mathscr{T}$, we use $P(T|d)$ to denote the probability that the transition is in fact T given the observation of document d. Thus, we are interested in the transition in $\mathscr{T}$ that maximizes such a probability, i.e.,

$$T^* = \arg\max_{T\in\mathscr{T}} p(T|d).$$

Using Bayes' rule, this can be rewritten as

$$T^* = \arg\max_{T\in\mathscr{T}} \frac{P(d|T) \times P(T)}{P(d)},$$

where $P(d)$ is the probability of observing d and $P(T)$ is that of the hidden state transition T. Since we build HMM for the same given document d, the probability $p(d)$ of generating it is a constant. Thus, the equation can also be written as

$$T^* = \arg\max_{T\in\mathscr{T}} P(d|T) \times P(T).$$

Here, we assume that document d can be represented as a first-order Markov chain and that the words in the sequence are independent of each other (i.e., uni-gram). We can further rewrite the above equation as

$$\begin{aligned} T^* &= \arg\max_{T\in\mathscr{T}} P(T) \times \prod_{i=1}^{n} P(w_i|s_i) \\ &= \arg\max_{T\in\mathscr{T}} P(s_1) \times \prod_{i=1}^{n-1} P(s_{i+1}|s_i) \times \prod_{i=1}^{n} P(w_i|s_i), \end{aligned}$$

where $\{s_1, s_2, \ldots, s_n\}$ is a sequence of states that the hidden Markov chain has visited when generating d. In the above derivation, $P(s_{i+1}|s_i)$ is the state transition probability, with $P(s_1)$ being the probability of the initial state s_1, and $P(w_i|s_i)$ is output probability.

While this maximization problem can be solved using the Viterbi algorithm [21], in order to apply it, we first need to obtain the parameters of the HMM, i.e., the above output and transition probabilities. There can be two possible approaches to calculate them. We could apply the Baum–Welch algorithm [21] iteratively to estimate them simultaneously. However, since the output probabilities can be computed using the statistical language models as a separate source (Section 3.3), we modify the Baum–Welch algorithm to only estimate the state transition probabilities and hold the output probabilities constant. This decision has been backed by our preliminary experiments. The similar idea to limit the parameter estimation in HMM for better performance have also been adopted in part-of-speech tagging for natural language processing in [8,11].

Extracting n ($n > 1$) snippets from the same document would require either extending the HMM to accommodate n relevant states or running the above single-snippet procedure multiple times. We experimented with both and found that the former tends to break the a long relevant snippet into several fragments and return them separately. Therefore, to produce multiple relevant snippets iteratively, we first extract and mask a relevant snippet using the above single-snippet procedure. The rest of the document is applied with the same procedure again. This is repeated until the difference of the word lengths between two snippets falls below a threshold θ. The details of our algorithm are described in Fig. 6.

3.3. Language model parameter estimation

Here, we investigate the word generation probabilities of the relevance language model M_r and the irrelevance language mode $M_{\bar{r}}$. For the irrelevance model $M_{\bar{r}}$, the probability $P(w|M_{\bar{r}})$ of observing word w can be closely approximated by that regulated by a background language model M, which combines M_r and $M_{\bar{r}}$. This is true because the irrelevant part of the language dominates the vast majority of the background language given the small number of relevant documents relative to the entire corpus. Thus, for given word w and the entire text corpus, we denote the number of times that w occurs in the corpus by f_w, and the total number of tokens (i.e., words with repetition) in the corpus by F. Therefore, we estimate the word generation probability of w as

$$P(w|M_{\bar{r}}) = P(w|M) = \frac{f_w}{F}.$$

Estimation of the parameters of the relevance language model M_r would require a good sample of the text snippets relevant to the query. However, this information is not available at this point.

```
Input : Document $d$, query $q$
Output : Relevant snippet set $\mathcal{S}$
Step 1 : Initialization
1.1. $words = \emptyset$ /* word map of probabilities
1.2. $HMM = \emptyset$ /* Hidden Markov Model */
1.3. $tMat = \emptyset$ /* transition probability matrix */
1.4. $oMat = \emptyset$ /* output probability matrix*/
/* compute output probability for each word using language models */
1.5. for each word $w \in d$ do
        $P(w|M_r)$ = getRMProb($w, q$)
        $P(w|M_{\bar{r}})$ = getIRMProb($w, q$)
        $words$.add($w$, $P(w|M_r)$, $P(w|M_{\bar{r}})$)
1.6. $tMat$.init() /* create transition matrix based on Figure 5 */
1.7. $oMat$.set($d$, $words$) /* create output probability matrix language models */
Step 2: Identification of relevant snippets
2.1. $\mathcal{S} = \emptyset$, $S_{cur}$ = null, $S_{prev}$ = null
2.2. $HMM$.baumwelch($d$, $tMat$, $oMat$) /* estimate HMM parameters */
2.3. $S_{cur} = HMM$.viterbi($d$) /* extract relevant snippet */
2.4 $H_{cur} = S_{cur}$.length()
2.5. do {
        $S_{prev} = S_{cur}$
        $HMM$.baumwelch($d$, $tMat$, $oMat$)
        $S_{cur} = HMM$.viterbi($d$)
        $H_{prev} = H_{cur}$
        $H_{cur} = S_{cur}$.length()
        if $\left(\frac{H_{cur}}{H_{prev}} < \theta\right)$
            break;
        $d$.maskSubqueue($S_{cur}$)
        $\mathcal{S}$.add($S_{cur}$)
}
2.6. Return $\mathcal{S}$
```

Fig. 6. Snippet extraction using HMM.

Therefore, to address this issue, we take the approach of Lavrenko and Croft [13] by assuming that M_r can be built equally well on the set of documents relevant to the query. Specifically, we conduct the following operations.

- Use the query q to retrieve a set of highly ranked documents D_r using any good text filter, e.g., Apache Lucene. This step yields a set of documents that contain most or all of the words in q. In addition, each document d in D_r has a probability $P(d|q)$ that d should be retrieved based on q.
- Calculate the probability $p(w|d)$ of word w being generated by a given document d using a maximum likelihood estimate, smoothed by the background language model M:

$$P(w|d) = \lambda \frac{f_{w,d}}{|d|} + (1-\lambda)P(w|M),$$

  where $f_{w,d}$ is the number of times that w occurs in d and $|d|$ is the number of tokens in d. Note that we use a parameter λ to control the contribution of the term frequency to this probability. This is a common technique in natural language modeling. We have investigated the effect of $|D_r|$ and λ and the results are reported in Section 4.7.
- Calculate $P(w|D_r)$ to approximate $P(w|M_r)$, the probability that given word w is generated by the relevance language model:

$$P(w|M_r) \approx P(w|D_r) = \sum_{d \in D_r} P(w|d)P(d|q).$$

  This summation can be taken over only those in D_r that have non-trivial $P(d|q)$ to accelerate execution.

With the word generation probabilities of M_r and $M_{\bar{r}}$ in place, we can extract text snippets relevant to the query q using the algorithms in the previous subsections (Sections 3.1 and 3.2).

Table 1
Data sets used for evaluation.

	Single	Multiple	Noisy
Number of documents in collection		226,087	
Average length of relevant documents	521	1452	527
Number of queries for evaluation	80	54	80

4. Experimental evaluation

In this section, we present the experiments to evaluate the two proposed methods for extracting personalized text snippets. Both methods utilize language models to generate a pair of relevant/irrelevant probabilities of each word with regard to a query in the relevant document. Recall that one method is based on word sequence analysis by directly utilizing these probabilities, and the other treats the document as a observed chain of the hidden Markov model parameterized with these probabilities. In this section, we call these methods SE-WSA and SE-HMM, respectively.

4.1. Experimental settings

To gauge how well the proposed snippet identification approach performs, we carry out our experiments using data set synthesized from the TREC DOE (U.S. Department of Energy) collection (http://trec.nist.gov). This collection contains 226,087 documents along with 80 benchmark queries. Each document is an abstract from some longer DOE publication, and their average length is 149 words. Among these 226,087 document abstracts, 2352 are considered relevant to any of the 80 queries.

We build our test documents using these abstracts. Since each abstract is a self-containing snippet of average length 149, we can synthesize longer documents with different properties by compounding a number of relevant abstracts and irrelevant ones. Using these synthesized documents, we can test our algorithms in the following the aspects of (a) extraction of the most relevant snippet, (b) identification multiple relevant snippets and (c) extraction of relevant snippets under noisy conditions. Specifically, we generate three documents sets as follows.

- *Single-snippet data set* (S_1)—Each document in this set is created by concatenating a relevant abstract with a number of randomly selected irrelevant abstracts.
- *Multiple-snippet data set* (S_2)—In this set, each document is generated by concatenating 2–5 relevant abstracts with a number of randomly selected irrelevant abstracts. In addition, the relevant abstracts included in the document is always relevant with regard to the same query. Since 54 queries out of the 80 have at least two relevant abstracts in the document collection, only these 54 queries can be used to test this data set.
- *Noisy data set with single snippet* (S_3)—This date set is an enhancement of the first data set by including noise to the generated documents. For each query and a relevant abstract, we generate a document by concatenating the relevant abstract with a number of randomly selected irrelevant ones as when generating the first data set. Next, we insert words in the query to at least three loci around the irrelevant-abstract parts of the document. This data set represents a much more realistic scenario for testing.

Details of our synthesized data sets are summarized in Table 1.

4.2. Baseline methods

In addition to the proposed methods, we also implemented two well-known snippet extraction methods as the baseline.

The first method, FL-Win, is commonly applied when presenting snippets in the search result pages [18]. Given a window size k, it examines all k-word-long text segments and selects that with the most occurrences of the query words to be the most relevant. In our experiments, the window size k is set to 149 words, i.e., the average word length of the ground truth snippets for the data sets (Section 4.1). Without a priori knowledge about the appropriate length of the relevant snippet in each document, the best that an extract method can do is to adopt one that can perform well in the average case. We believe the true average relevant snippet length is an optimal value, and is the best setting that the system can choose for FL-Win method. As a matter of fact, this gives it an unrealistic advantage and it can be regarded as a strong baseline with over-estimated performance.

The second is the Cos-Win method, which extracts variable-length text segments as relevant snippets in response to a query submission [10]. In particular, the text segments of a set of predefined lengths are examined for each document, and the one with the highest cosine similarity with the query is selected as the most relevant snippet.

In both baseline methods, we follow the example of [10] by using text segments starting at every 25th word instead of every word in a document to reduce the computational complexity. As reported by Kaszkiel and Zobel [9,10], starting at 25-word intervals is as effective as starting at every word. In case of Cos-Win baseline method, we followed the example of [10] by first selecting one representative segment at each starting word among several candidate text segments starting at the same word with size ranging from 50 to 600 words, with increments of 25. The relevant snippets are then identified by the one with the highest cosine similarity with the query among these representatives. We applied the same cosine similarity between a text segment and a query as defined by [9] in the experiments.

4.3. Metrics

Given the identified snippets, we compute precision P and recall R as follows. Let L_s be the length of the ground truth snippet. Let L_e be the word length of the extracted snippet and L_o be the overlap between the ground truth and the extracted snippet. Then,

$$P = \frac{L_o}{L_e}$$

and

$$R = \frac{L_o}{L_s}.$$

In addition, we also use the harmonic mean (i.e., the F-measure), which assumes a high value only when both recall and precision are high

$$F = \frac{2}{\frac{1}{P} + \frac{1}{R}}.$$

For all of the three data sets in Section 4.1, we apply our proposed methods (SE-WSA and SE-HMM) and the two baseline methods (FL-Win and Cos-Win) to investigate their performance in text snippet extraction. For each method, we record its P, R and F measures along with time cost (Table 2).

4.4. Statistical significance

Before reporting our experimental results using precision, recall and F-measure, we provide a significance test to show that the observed differences are not incidental. The Wilcoxon signed-rank test and t-test are commonly used for the significance test in information retrieval experiments [27]. In a nutshell, both take a pair of equal-sized sets of per-query effectiveness values, and assign a confidence value to the null hypothesis that the values are drawn from the same distribution. If confidence in the hypothesis (reported as a p-value) is less than ($\leq 5\%$), it is typically rejected. Although such tests only consider the null hypothesis, it is common to assume rejection implies that values are most likely drawn from different distributions, which means that the results of experiments are statistically significant.

Among the assumptions of the Wilcoxon signed-rank test and the t-test are that the values being tested—in our case, per-query effectiveness—are distributed symmetrically and normally, respectively [27]. However, effectiveness rarely follows either distribution. Countering such caution, Hull [7] points out that the t-test can be reliable even when data being tested are not distributed normally. Therefore, we applied paired t-test to find out whether the observed differences is identical. Our t-test showed that using F-measure as the performance measure, the proposed methods (SE-WSA and SE-HMM) performed better than the baseline methods (FL-Win and Cos-Win) at $p = 0.001$. This is much less than the critical confidence value (0.05).

Table 2
Performance of alternative methods.

Method	P	R	F	Time cost (ms)
Cos-Win	0.685	0.415	0.517	28.9
FL-Win	0.781	0.573	0.661	17.6
SE-WSA	0.749	**0.856**	0.799	33.2
SE-HMM	**0.879**	0.793	**0.834**	31.5

4.5. Overall performance

We compare our methods to the two baseline methods using the single-snippet data set (S_1 of Section 4.1). Their measures are in Table 2. As indicated in the table, SE-HMM is the best in terms of precision. Based on the recall, on the other hand, SE-WSA outperforms the other methods. Further analysis of the results (Section 4.6) shows that, in general, SE-WSA tends to extract longer snippets, which correctly include the ground-truth snippets to achieve high recall, but reducing its precision. Although SE-WSA and SE-HMM win over each other in terms of recall and precision, respectively, the combined F-measure asserts that SE-HMM is the most effective among all (0.834) with SE-WSA trailing closely (0.799). While in most categories our proposed methods prove to superior, there is one exception when comparing SE-WSA to FL-Win in terms of precision. This is because the design of SE-WSA generates the longest snippets, thus, sacrificing its precision; but note that its recall is much higher even compared to our other proposed method SE-HMM. This provides a tradeoff between the precision and recall measures. The last column of the table summarizes the time cost of these four methods in terms of execution time per document per query.

4.6. Effect of snippet length

We focus on the effect of snippet length on the performance of these methods. To do that, we classify the single-snippet documents into different groups according to the length of the contained text snippet. Specifically, the groups are documents of snippet lengths up to 50, 51–100, 101–200, 201–300, 301–400 and 401–500. We plot the values of P, R and F-measure for these methods in different snippet length groups in Figs. 7(a), (b) and (c), respectively. The x-axes are labeled with the upper limits of these snippet length groups.

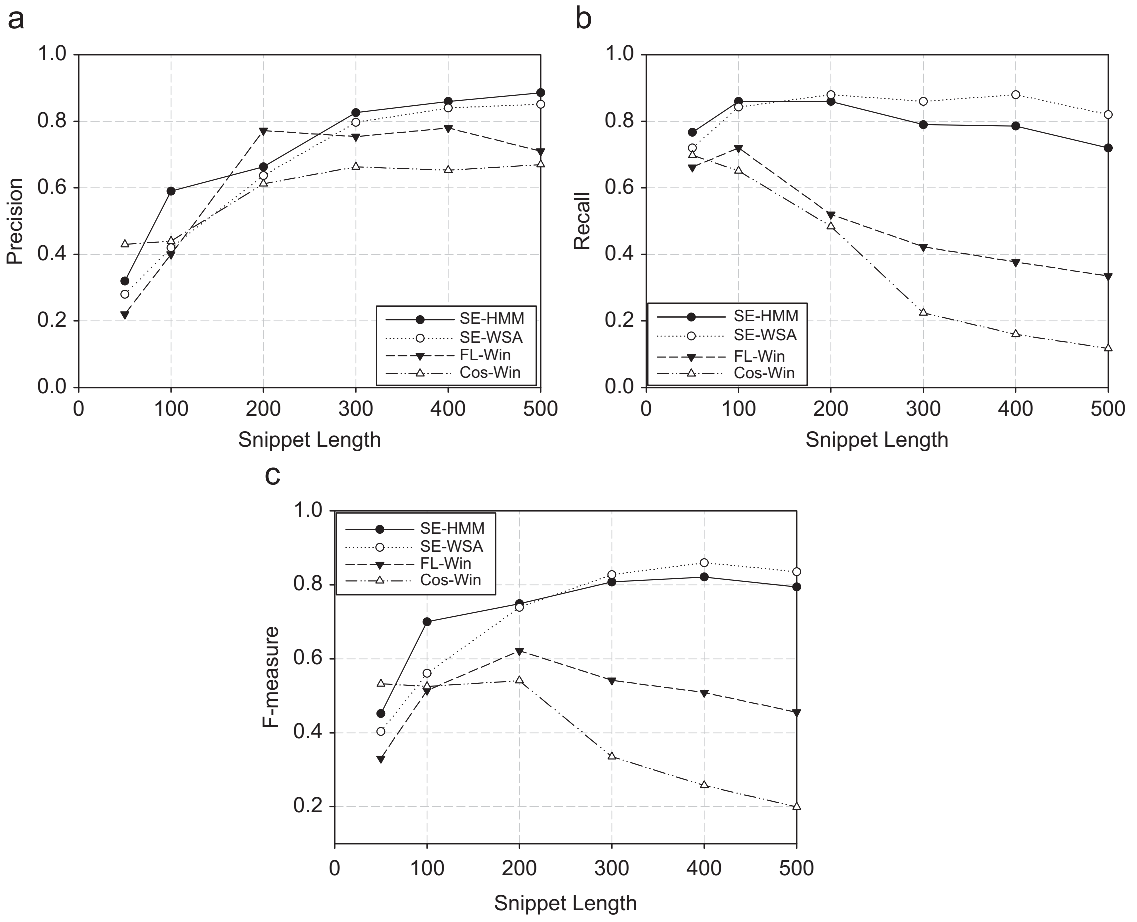

Fig. 7. Effect of snippet length on (a) precision, (b) recall and (c) *F*-measure.

The general trend of extraction precision is increasing as the ground-true length increases (Fig. 7(a)). The only exception is the FL-Win reaches its highest precision for ground-true length of 101–200. This is an artifact of setting the fixed window length to 149, the average snippet length, in the algorithm design (Section 4.2). We observe that, for FL-Win, its precision is affected adversely by the increase of ground-true length. When comparing across different methods, we see that SE-HMM and SE-WSA are more precise than the baseline methods for most groups except for FL-Win at 101–200. In terms of recall (Fig. 7(b)), SE-HMM and SE-WSA stabilize as the ground-true length increases while FL-Win and Cos-Win quickly deteriorate. This is attributed by the much stronger capability of SE-HMM and SE-WSA in adapting to different ground-true lengths whereas the baseline methods are only suitable for short ones. The *F*-measure (Fig. 7(c)) possesses a similar trend to recall.

4.7. Language model parameters

The relevance language model that forms the basis of SE-WSA and SE-HMM have two main parameters: (1) the size of retrieved relevant document set, $|D_r|$ and (2) the smoothing parameter, λ (Section 3.3).

We first take the *F*-measure for different document set size $|D_r|$ (Fig. 8). The figure indicates that the performance is stable when $|D_r|$ is set to anywhere between 5 and 40 documents. Using fewer than five documents does not provide sufficient retrieval background, but using more than 50 documents introduces excessive noises. In our experiments, unless other specified, we always use the top-15 documents in estimating the language model.

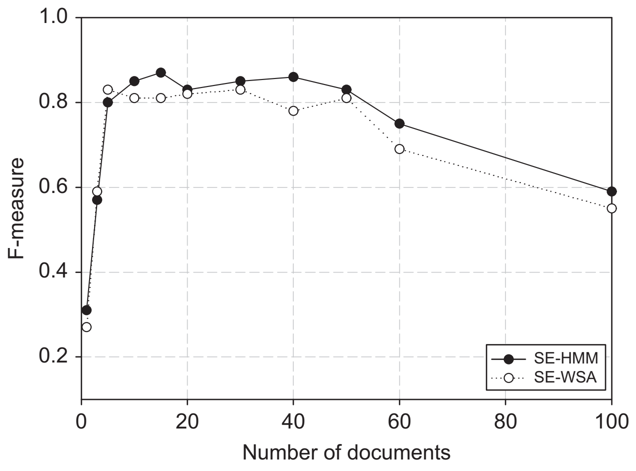

Fig. 8. Effect of the number ($|D_r|$) of documents used in modeling.

Usually, smoothing would have a very strong impact on performance of language model parameter estimation [20]. An important role of smoothing in the language models is to ensure non-zero probabilities for every word under the document model and to act as a variance-reduction technique. In our experiments, however, we observe that setting λ to 0.9 yields the best *F*-measure. In other words, with almost no smoothing of the document models, SE-WSA and SE-HMM achieve their best performance. This is because we incorporate

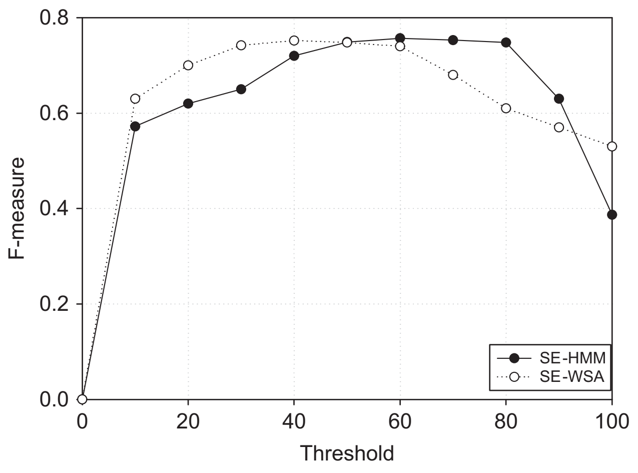

Fig. 9. Threshold (θ).

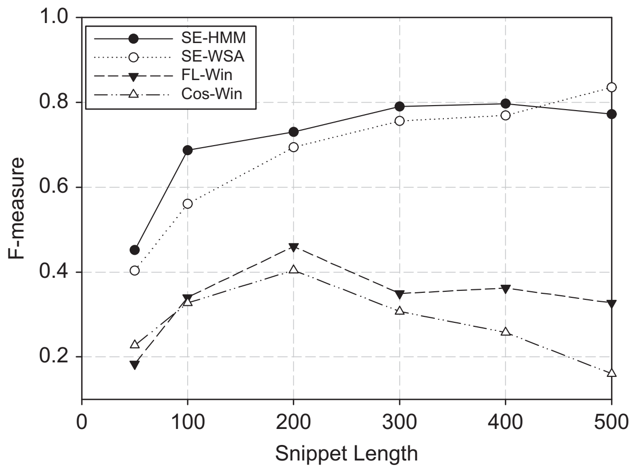

Fig. 10. Documents with noise.

Table 3
Multiple relevant snippets.

Method	*P*	*R*	*F*
Cos-Win	0.601	0.374	0.461
FL-Win	0.701	0.498	0.582
SE-WSA	0.73	0.776	0.752
SE-HMM	0.752	0.763	0.757

a number of the relevant documents when estimating the relevance language model for a given query. This results in non-zero probabilities for many words that do not occur in the original query, thus, little need for smoothing to avoid zeros.

4.8. Performance of multiple relevant snippets

Both SE-WSA and SE-HMM utilize thresholds to automatically decide the number of relevant snippets to be selected as described in Sections 3.1 and 3.2. We plot the *F*-measures of these methods against the changing values of threshold θ in Fig. 9 using the multiple-snippet data set S_2. Note that the threshold in SE-WSA determines "loss of returned area". As shown in the figure, the *F*-measure is high when set anywhere between 50% and 80%. In contrast, the threshold in SE-HMM controls the "loss of word sequence length". In the figure, the performance peaks when θ is 30%–60%. Therefore, in our experiments, unless explicitly stated, we set θ in SE-WSA to 40% and that in SE-HMM to 60%, respectively.

The baseline methods Cos-Win and FL-Win can also be extended to extract multiple relevant text snippets. To that, we also employ a similarity threshold to control the number of iterations executed. We compare our methods with them and record the precision, recall and *F*-measures in Table 3. From the table, we notice that SE-WSA and SE-HMM are capable of generating more relevant text snippets when multiple are extracted.

4.9. Immunity to noise

Up to this point, we have experimented with data with clean delineation of relevant and irrelevant parts. To study the affects of more realistic situations such as in presence of noise, we evaluate all the methods using the noisy data set S_3. In S_3, a document may have several text segments that contain the query words, but only one segment corresponds to the ground truth; others are noises. We plot the *F*-measure of all four methods tested using documents in S_3 (Fig. 10). These documents are grouped in length as previously. In the figure, SE-WSA and SE-HMM perform almost the same as in the noiseless situations (Figs. 7(a) and (b), Section 4.6). Oppositely, the performance of FL-Win and Cos-Win drop significantly when facing noisy data. In essence, for a method based on the relevance language model, it is able to select query-relevant snippets by not only considering the exact words in a query but also by incorporating other words closely related to these words. Thus, when there is noise in a document, SE-WSA and SE-HMM can eliminate irrelevant segments which have superficial connections to the query.

5. Concluding remarks

Previous work on text snippet extraction either outputs document fragments segmented prior to a user query or uses a sliding window to highlight some words of the query. While these methods may be light-weight, they have fundamental limitations. In the former approach, pre-processed document segments have rigid boundaries. Even if overlaps are allowed, the rigidness of document segmentation is not flexible enough to support accurate personalization according to user queries. For the latter approach, the start–end positions of the sliding window are usually defined so that a large number of words in the query appear in a small window. That is, to maximize the "number of query words per window length". Typically, less important words are given smaller weights when calculating the above ratio. However, such a weighting scheme is necessarily heuristic in nature. In this work, we have shown that personalized text snippet can be improved by utilizing statistical language models to effectively capture the commonalities between a document and the user intention. Taking the advantage of language models, we can lose the implicit assumption made in existing work that all words in the query contributes equally to personalizing the relevant snippet. This flexibility allows our method to produce more coherent text snippets personalized to different users. In future research, we plan to work on as smart but more efficient methods for personalized text snippet extraction.

Acknowledgements

This research is supported by National Natural Science Foundation of China Grant no. 60803106. We would also like to thank Chen Zhang and Jia Wang for their help in various phases of the implementation.

References

[1] J.P. Callan, Passage-level evidence in document retrieval, in: Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Springer, New York, 1994, pp. 302–310.

[2] C. Clarke, G. Cormack, D. Kisman, T. Lynam, Question answering by passage selection (multitext experiments for trec-9), in: Proceedings of the 9th Text Retrieval Conference (TREC), 2000, pp. 673–684.

[3] J. Conroy, D.P. O'leary, Text summarization via hidden Markov models and pivoted QR matrix decomposition, Technical Report CS-TR-4221, University of Maryland, College Park, Maryland, USA, 2001.

[4] P. Fung, G. Ngai, C.-S. Cheung, Combining optimal clustering and hidden Markov models for extractive summarization, in: ACL Workshop on Multilingual Summarization and Question Answering, Association for Computational Linguistics, Morristown, NJ, USA, 2003, pp. 21–28.

[5] M.A. Hearst, Texttiling: segmenting text into multi-paragraph subtopic passages, Computational Linguistics 23 (1) (1997) 33–64.

[6] M.A. Hearst, C. Plaunt, Subtopic structuring for full-length document access, Technical Report, Berkeley, CA, USA, 1993.

[7] D. Hull, Using statistical testing in the evaluation of retrieval experiments, in: Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, New York, NY, USA, 1993, pp. 329–338.

[8] F. Jelinek, R. Mercer, Probability distribution estimation from sparse data, IBM Technical Disclosure Bulletin 28 (6) (1985) 2591–2594.

[9] M. Kaszkiel, J. Zobel, Passage retrieval revisited, in: Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, New York, NY, USA, 1997, pp. 178–185.

[10] M. Kaszkiel, J. Zobel, Effective ranking with arbitrary passages, Journal of the American Society for Information Science and Technology 52 (4) (2001) 344–364.

[11] J. Kupiec, Robust part-of-speech tagging using a hidden Markov model, Computer Speech & Language 6 (3) (1992) 225–242.

[12] V. Lavrenko, J. Allan, E. DeGuzman, D. LaFlamme, V. Pollard, S. Thomas, Relevance models for topic detection and tracking, in: Proceedings of the 2nd International Conference on Human Language Technology Research, Morgan Kaufmann, San Francisco, CA, USA, 2002, pp. 115–121.

[13] V. Lavrenko, W.B. Croft, Relevance based language models, in: Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, New York, NY, USA, 2001, pp. 120–127.

[14] V. Lavrenko, M. Schmill, D. Lawrie, P. Ogilvie, D. Jensen, J. Allan, Language models for financial news recommendation, in: Proceedings of the 9th International Conference on Information and Knowledge Management (CIKM), ACM, New York, NY, USA, 2000, pp. 389–396.

[15] C.Y. Lin, E. Hovy, From single to multi-document summarization: a prototype system and its evaluation, in: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, 2002, pp. 457–464.

[16] X. Liu, W.B. Croft, Passage retrieval based on language models, in: Proceedings of the 7th International Conference on Information and Knowledge Management (CIKM), 2002, pp. 375–382.

[17] F. Llopis, J.L. Vicedo, A. Ferrández, Passage selection to improve question answering, in: Proceedings of the Conference on Multilingual Summarization and Question Answering (COLING), Association for Computational Linguistics, Morristown, NJ, USA, 2002, pp. 1–6.

[18] R. Nobles, Pleased with your google description? Technical Report, 2004.

[19] J.M. Ponte, W.B. Croft, Text segmentation by topic, in: Proceedings of the 1st European Conference on Research and Advanced Technology for Digital Libraries (ECDL), Springer, London, UK, 1997, pp. 113–125.

[20] J.M. Ponte, W.B. Croft, A language modeling approach to information retrieval, in: Proceedings of the 21st annual international ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, New York, NY, USA, 1998, pp. 275–281.

[21] L.R. Rabiner, A tutorial on hidden markov models and selected applications in speech recognition, Proceedings of the IEEE 77 (2) (1989) 257–286.

[22] D.R. Radev, H. Jing, M. Budzikowska, Centroid-based summarization of multiple documents: sentence extraction, utility-based evaluation, and user studies, in: Proceedings of ANLP/NAACL Workshop on Summarization, 2000, pp. 21–29.

[23] I. Roberts, R. Gaizauskas, Evaluating passage retrieval approaches for question answering, in: Proceedings of 26th European Conference on Information Retrieval, 2004, pp. 72–84.

[24] S.E. Robertson, The probability ranking principle in IR, Journal of Documentation 33 (4) (1977) 294–304.

[25] R. Rosenfeld, Two decades of statistical language modeling: where do we go from here, in: Proceedings of the IEEE, 2000.

[26] G. Salton, J. Allan, A. Singhal, Automatic text decomposition and structuring, Information Processing and Management 32 (2) (1996) 127–138.

[27] M. Sanderson, J. Zobel, Information retrieval system evaluation: effort, sensitivity, and reliability, in: Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, New York, NY, USA, 2005, pp. 162–169.

[28] F. Song, W.B. Croft, A general language model for information retrieval, in: Proceedings of the 8th International Conference on Information and Knowledge Management (CIKM), ACM, New York, NY, USA, 1999, pp. 316–321.

[29] M. Witbrock, V.O. Mittal, Headline generation: A framework for generating highly-condensed non-extractive summaries, in: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 1999, pp. 315–316.

[30] M. Witbrock, V.O. Mittal, Summarizing text documents: sentence selection and evaluation metrics, in: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 1999, pp. 121–128.

[31] D. Zajic, B. Dorr, R. Schwartz, Automatic headline generation for newspaper stories, in: Proceedings of the ACL Workshop on Automatic Summarization and Document Understanding Conference (DUC), 2002, pp. 78–85.

[32] D. Zhang, W.S. Lee, A language modeling approach to passage question answering, in: Proceedings of the Text Retrieval Conference (TREC), 2003.

About the Author—QING LI is an associate professor of Southwestern University of Finance and Economics, China. Currently, he is also a visiting professor at the Department of Computer Science and Engineering at the Arizona State University. Prior to that he was a post-doctoral Researcher with ASU and Information & Communications University of Korea separately. Li's research interests lie primarily in intelligent information access and text data mining. He has published over 20 articles in respected journals and conferences in related areas. He served in the organization and program committees of various international conferences including PC member of SIGIR 2008, CIKM 2007 and AIRS2005. He is also the winner of Chinese Government Award for Outstanding Self-financed Students Abroad in 2004. He received his Ph.D. from Kumoh National Institute of Technology in February 2005, and his M.S. and B.S. degrees from Harbin Engineering University, China.

About the Author—YUANZHU PETER CHEN is an assistant professor in the Department of Computer Science at Memorial University of Newfoundland in St. John's, Newfoundland. He received his Ph.D. from Simon Fraser University in 2004 and B.Sc. from Peking University in 1999. Between 2004 and 2005, he was a post-doctoral researcher at Simon Fraser University. His research interests include mobile ad hoc networking, wireless sensor networking, distributed computing, combinatorial optimization, graph theory and information retrieval.