A Dynamic Programming Model for Text Segmentation Based on Min-Max Similarity

Na Ye[1], Jingbo Zhu[1], Yan Zheng[1],
Matthew Y. Ma[2], Huizhen Wang[1], and Bin Zhang[3]

[1] Institute of Computer Software and Theory,
Northeastern University, Shenyang 110004, China
[2] IPVALUE Management Inc. 991 Rt. 22 West, Bridgewater, NJ 08807, USA
[3] Institute of Computer Applications, Northeastern University, Shenyang 110004, China
yn.yena@gmail.com,
{zhujingbo,wanghuizhen,zhangbin}@mail.neu.edu.cn,
mattma@ieee.org

Abstract. Text segmentation has a wide range of applications such as information retrieval, question answering and text summarization. In recent years, the use of semantics has been proven to be effective in improving the performance of text segmentation. Particularly, in finding the subtopic boundaries, there have been efforts in focusing on either maximizing the lexical similarity within a segment or minimizing the similarity between adjacent segments. However, no optimal solutions have been attempted to simultaneously achieve maximum within-segment similarity and minimum between-segment similarity. In this paper, a domain independent model based on min-max similarity (MMS) is proposed in order to fill the void. Dynamic programming is adopted to achieve global optimization of the segmentation criterion function. Comparative experimental results on real corpus have shown that MMS model outperforms previous segmentation approaches.

Keywords: text segmentation, within-segment similarity, between-segment similarity, segment lengths, similarity weighting, dynamic programming.

1 Introduction

A natural language discourse is usually composed of multiple subtopics, which in turn may convey only one main topic. In traditional text processing tasks such as information retrieval (IR), question answering (QA) and text summarization, if the subtopic structure of a text can be identified and consequently its semantic segments can be used in the basic processing unit, the performance of the system will be greatly improved [1][2]. In addition, the segment-based IR will provide users with answers of higher accuracy and less redundancy results. The core technology involved in the identification of subtopic structure and therefore semantic segments of a text is called text segmentation, which is the focus of this paper.

In text segmentation, it becomes critical how to design a good criterion to evaluate the subtopic coherence of a document. According to the definition of text segmentation

H. Li et al. (Eds.): AIRS 2008, LNCS 4993, pp. 141–152, 2008.

task, in an appropriate segmentation, sentences within a segment convey the same subtopic while sentences among different segments belonging to different subtopics. Therefore, in order to achieve good separation over all segments, both high within-segment similarity and low between-segment similarity should be achieved. However, in previous literature [2-10], no solutions have been given to simultaneously optimize both within-segment and between-segment similarities.

Another important issue in text segmentation is the strategy for finding the best segmentation. Some algorithms use local optimization approaches, such as sliding window [2-3]and divisive clustering [6-7], to detect subtopic changes. Some models use more global strategy by representing lexical distribution on a *dotplot* [4]. However this is still not a complete globalization strategy. A truly global optimization searching strategy is dynamic programming [5] [8] [10]

In this paper we present a global optimization model, MMS, for text segmentation. This model adopts a segmentation criterion function attempting to optimize both within-segment lexical similarity and between-segment lexical similarity. Segmentation with maximum within-segment and minimum between-segment similarity is selected as the optimal one. In MMS model, additional text structure factors, such as segment lengths and lexical similarity weighting strategy based on sentence distance, are also incorporated as part of lexical similarity weighting strategy. To achieve global optimization, we implemented our MMS model using the dynamic programming searching strategy, with which the number of segments can be determined automatically. Experimental results show that our MMS model outperforms other popular approaches in terms of P_k [11]and *WindowDiff* [12] measure.

The remainder of this paper is organized as follows. Literature research is briefly reviewed in Section 2. In Section 3, the proposed MMS model and a complete text segmentation algorithm are described in detail. In Section 4, experimental results are given to compare our approach with other popular systems. At last, we draw conclusion and address future work in Section 5.

2 Related Work

Existing text segmentation algorithms can be classified into two categories with respect to the segmentation criteria being employed. One is to make use of the property that lexical similarity within a segment is high. Lexical densities within segments are measured to find lexically homogeneous text fragments [6-10]. The second approach assumes that lexical similarity between different segments is low, and subtopic boundaries correspond to locations where adjacent text fragments have the lowest lexical similarity [2-5].

In contrast to previous work, the focus of our work is not only lexical relations within a segment or between different segments, but appropriate combination of the two factors. Fundamental structural factors of written texts, such as segment length and sentence distance are also taken into account in the design of the segmentation criterion. In analogy, our work is similar to [13], which measured homogeneity of a segment not only by the similarity of its words, but also by their relation to words in other segments of the text. However, their method is designed for spoken lecture segmentation and can not address the problems of written texts very well. Zhu [14]

used Multiple Discriminant Analysis (MDA) criterion function to find the best segmentation by means of the largest word similarity within a segment and the smallest word similarity between segments. However, the algorithm applied a full search to find the optimal segmentation, which is an NP problem with high computational complexity. In comparison, MMS model adopts dynamic programming strategy, which greatly reduces the time cost. Fragkou[10] also used dynamic programming in optimizing the segmentation cost function. But their method only considers within-segment similarity and needs prior information about segment length.

3 The Segmentation Algorithm

3.1 Problem Definition

Let's assume a text consists of K sentences, denoted by $S = \{1, 2, ..., K\}$, and has a vocabulary of T distinct words $V = \{w_1, w_2, ..., w_T\}$. Each sentence can be represented as a point in a T-dimensional data space. Assume that the topic boundaries occur at the ends of sentences and there are N segments in the text, then the task of text segmentation is to partition the sentences into N groups $G = \{1, 2, ..., p_1\}$, $\{p_1 + 1, p_1 + 2, ..., p_2\}, \ldots, \{p_{N-1} + 1, p_{N-1} + 2, ..., K\}$. Each group $G_i = \{p_{i-1} + 1, p_{i-1} + 2, ..., p_i\}$ is a segment that reflects an individual subtopic. A shorter representation of the segmentation can be given as $G = \{p_0, p_1, ..., p_N\}$, where $p_0, p_1, ..., p_N$ are segment boundaries with $p_0 = 0$ and $p_N = K$. Text segmentation aims at finding the best segmentation G^* among all possible segmentations.

In this paper we design a criterion function J to evaluate segmentations of a text. Thus the process of finding the best segmentation can be viewed as the process of finding the segmentation with the highest evaluation score as follows:

$$G^* = \arg\max_G J(S, G) . \tag{1}$$

In the following section we will introduce our motivation in designing the criterion.

3.2 Motivation

It will reasonably hold true that in an appropriately segmented text, sentences within a single segment are topically related and sentences that belong to adjacent segments are topically unrelated conveying different subtopics. In much of previous work[4] [6-10], the lexical similarity is a natural candidate in measuring the topical relation of sentences. If two sentences describe the same topic, words used in them tend to be related to one another. Thus, within a segment, vocabulary tends to be cohesive and repetitive, leading to significant within-segment lexical similarity; whereas between adjacent segments, the vocabulary tends to be distinct, leading to dismal between-segment similarity. We believe that the above lexical similarity property must exist for a good segmentation strategy.

3.3 Segmentation Criterion Function

Following the lexical similarity property stated above, we propose our MMS model to comprise of the segmentation evaluating criterion function as follows:

$$J = F(Sim_{Within}, Sim_{Between}) . \tag{2}$$

where Sim_{Within} refers to the within-segment similarity, $Sim_{Between}$ refers to the between-segment similarity. F is a function whose value increases as Sim_{Within} increases, and decreases as $Sim_{Between}$ increases. The best segmentation can be achieved by maximizing the value of F, which is expressed as:

$$F(Sim_{Within}, Sim_{Between}) = \alpha \cdot Sim_{Within} - (1-\alpha) \cdot Sim_{Between} . \tag{3}$$

where α and $1-\alpha$ are the relative weight of within-segment lexical similarity and between-segment lexical similarity, respectively.

Within-segment lexical similarity is:

$$Sim_{Within} = \sum_{i=1}^{N} \frac{\sum_{m=p_{i-1}+1}^{p_i} \sum_{n=p_{i-1}+1}^{p_i} D_{m,n}}{(p_i - p_{i-1})^2} . \tag{4}$$

where m and n are the m^{th} and n^{th} sentence in the text. $D_{m,n}$ is the lexical similarity between sentence m and sentence n. The value of $D_{m,n}$ equals to one if there exist one or more words in common between sentence m and n, and zero otherwise. Sim_{Within} represents the global density of word repetition within segments.

Similarly, between-segment lexical similarity is defined as:

$$Sim_{Between} = \sum_{i=1}^{N} \frac{\sum_{m=p_i+1}^{p_{i+1}} \sum_{n=p_{i-1}+1}^{p_i} D_{m,n}}{(p_{i+1} - p_i)(p_i - p_{i-1})} . \tag{5}$$

$Sim_{Between}$ represents the global lexical similarity between adjacent segments.

Combining Eq. 3 to Eq. 5, the segmentation evaluation function is computed:

$$J = \alpha \cdot \sum_{i=1}^{N} \frac{\sum_{m=p_{i-1}+1}^{p_i} \sum_{n=p_{i-1}+1}^{p_i} D_{m,n}}{(p_i - p_{i-1})^2} - (1-\alpha) \cdot \sum_{i=1}^{N} \frac{\sum_{m=p_i+1}^{p_{i+1}} \sum_{n=p_{i-1}+1}^{p_i} D_{m,n}}{(p_{i+1} - p_i)(p_i - p_{i-1})} . \tag{6}$$

3.4 Text Structure Weighting Factors

In addition to segment lexical similarity, there are other text structure factors that are weighted into the proposed text segmentation algorithm.

- *Segment Length Factor*

In text segmentation, text pieces that are too short do not adequately describe an independent subtopic. For example, if there is a sentence in a text, and is not closely related with its adjacent text, and then it is likely a parenthesis or a connecting link between its preceding and its successive segments to enhance coherence. To address this phenomenon, we should avoid introducing too many segments. Restriction on segment number is added into the segmentation criterion function. It penalizes

segmentation choices with too many segments by assigning a small evaluation function score to it.

For example, we have a segmentation $G=\{G_1, G_2, \ldots, G_N\}$, each segment G_i has length L_i, and the length of the whole text is L. Then the length factor can be defined as $\sum_{i=1}^{N}(\frac{L_i}{L})^2$, where $L=\sum_{i=1}^{N} L_i$. This factor achieves low value when too many segments are produced.

- *Distance-based Similarity Weighting*

If we randomly select two sentences from a discourse, the probability of them belonging to the same segment varies greatly with the distance between them. Two sentences far apart are unlikely to belong to the same segment, whereas two adjacent sentences are much more likely. Therefore, we add a distance-based weighting factor to the density function, thus the lexical similarity of two sentences fluctuate with the distance between them.

Having incorporated the above factors, the overall segmentation evaluating function for our proposed MMS model becomes:

$$J=\alpha\cdot\sum_{i=1}^{N}\frac{\sum_{m=p_{i-1}+1}^{p_i}\sum_{n=p_{i-1}+1}^{p_i}W_{m,n}D_{m,n}}{(p_i-p_{i-1})^2} -(1-\alpha)\cdot\sum_{i=1}^{N}\frac{\sum_{m=p_i+1}^{p_{i+1}}\sum_{n=p_{i-1}+1}^{p_i}W_{m,n}D_{m,n}}{(p_{i+1}-p_i)(p_i-p_{i-1})}+\beta\cdot\sum_{i=1}^{N}(\frac{L_i}{L})^2 \quad . \tag{7}$$

where $W_{m,n}$ is the weighting factor, and based on the distance between the sentence m and sentence n. The values of m and n represent the positions of each corresponding sentence. An exemplary definition of $W_{m.n}$ is as follows:

$$W_{m,n}=\begin{cases}1 & if\ |m-n|\le 2\\ \dfrac{1}{\sqrt{|m-n|-1}} & else\end{cases} \tag{8}$$

We see that in our MMS model, rich information such as the within-segment similarity and between-segment similarity, segment length and the distance between sentences, are considered to discover topical coherence.

Eq. 7 represents the final form of the evaluation function in our MMS model, in which N stands for the desired number of segments.

3.5 Text Segmentation Algorithm

To optimize the segmentation evaluating function (Eq. 7) globally, we provide an implementation using the dynamic programming searching strategy to find the best segmentation. Since there is a between-segment similarity item in the function, two-dimension dynamic programming is applied. The complete text segmentation algorithm is shown in Figure 1, followed by detailed explanation.

Input: The $K \times K$ sentence similarity matrix D; the parameter α , β

Initialization

For t = 1, 2, ..., K

  For s = 1, 2, ..., t

    Sum = 0;

    For k = s, ..., t

      For w = s, ..., t

        Sum = Sum + $W_{w,k} \cdot D_{w,k}$;

      End

    End

    $S_{s,t}$ = Sum;

  End

End

Maximization

For t = 1, 2, ..., K

  For s = 0, 1, ..., t-1

$C_{t,s} = 0$;

$C_{t,0} = \alpha \cdot \frac{S_{1,t}}{t^2}$;

    For w = 0, 1, ..., s-1

If $C_{s,w} + \alpha \cdot \frac{S_{s+1,t}}{(t-s)^2} - (1-\alpha) \cdot \frac{S_{w+1,t} - S_{w+1,s} - S_{s+1,t}}{(t-s)(s-w)} + \beta \cdot (\frac{t-s}{K})^2 \geq C_{t,s}$

$C_{t,s} = C_{s,w} + \alpha \cdot \frac{S_{s+1,t}}{(t-s)^2} - (1-\alpha) \cdot \frac{S_{w+1,t} - S_{w+1,s} - S_{s+1,t}}{(t-s)(s-w)} + \beta \cdot (\frac{t-s}{K})^2$;

$Z_{t,s} = w$;

      EndIf

    End

  End

End

Backtracking

$e = -\infty$; k = 0; N=1;

For t = 0, 1, ..., T-1

  If $C_{T,t} \geq e$

    k = t;

  EndIf

End

$S_{K-1} = T$;

$S_K = k$;

While $Z_{S_{N-1}, S_N} > 0$

$N = N + 1$;

$S_N = Z_{S_{N-2}, S_{N-1}}$;

End

$N = N+1$;

$\hat{t} = 0$;

For k =1, 2, ..., N

$\hat{t}_k = S_{N-k}$;

End

Output: The optimal segmentation vector $\hat{t} = (\hat{t}_1, \hat{t}_2, \ldots, \hat{t}_N)$

Fig. 1. MMS text segmentation algorithm using dynamic programming scheme

In the above procedure, maximization and backtracking are the basic parts. During maximization, we recursively compute $C_{t,s}$, which is the optimal (maximum) value of the segmentation criterion function of the subtext starting from sentence 1 and ending at sentence t (with the second to the last boundary of s). That is, $C_{t,s}$ is the maximum (with respect to s and w) value of $C_{s,w} + \alpha \cdot \frac{S_{s+1,t}}{(t-s)^2} - (1-\alpha) \cdot \frac{S_{w+1,t} - S_{w+1,s} - S_{s+1,t}}{(t-s)(s-w)} + \beta \cdot (\frac{t-s}{K})^2$, where w is the best boundary before t and s. This sum is the optimal evaluation score to segment sentences 1 to s (with the second to last boundary of w) plus the score of creating a segment including sentences $s+1$ until t. $Z_{t,s}$ is the segment boundary preceding s (with the next boundary of t) in the optimal segmentation. Upon completion of the maximization part of the algorithm we have computed the maximum segmentation criterion function value for sentences 1 until K. The backtracking part produces the optimal segmentation $\hat{t} = (\hat{t}_0, \hat{t}_1, \ldots, \hat{t}_N)$ in reverse order. In this process, the optimal number of segments N is computed automatically. The time complexity of the algorithm is $O(K^3)$ (K is the number of sentences in the text).

4 Evaluation

The evaluation has been conducted systematically under a strict guideline in order to compare our approach with other state of the art algorithms on a fair basis. The key requirements are: 1) Evaluation should be conducted using a sizable testing data in order to generate meaningful results; 2) The testing data should be publicly available; 3) In order to compare with other people's work, we attempt to use their own implementations or published results as these are likely optimized for taking maximum advantages of their merits.

4.1 Experiment Settings

In our experiments, we use two suites of corpus including English one and Chinese one to evaluate the proposed model. The English testing corpus is the publicly available book *Mars* written by Percival Lowell in 1895. There are 11 chapters in all and the body of the book (6 chapters) is extracted for testing. Each chapter is composed of 2-4 sections. Chinese testing corpus is the scientific exposition *Exploring Nine Planets*. There are 10 chapters in the corpus and each chapter consists of 2-6 sections. The boundaries of paragraphs in the sections are taken as the subtopic boundaries for reference. Sections with few paragraphs (less than 3) are excluded.

In fact, there is another testing corpus developed by Choi[1], which is widely used for the evaluation of text segmentation algorithms. This is a synthetic corpus in which each article is a concatenation of ten text segments. A segment is the first *n* sentences of a randomly selected document from the Brown corpus. The motivation of constructing corpus in this way is to avoid the difficulty of judging subtopic boundaries by human beings. However this strategy has introduced obvious limitations to the corpus. Namely the subtopic similarity in the article is artificially enhanced, making the boundaries more distinct. This is quite different from real corpus and cannot represent the performance on real corpus by reducing the difficulty of segmentation. Therefore we used the real corpus instead of the synthetic one in our experiments.

To evaluate text segmentation algorithms, using precision and recall is inadequate because inaccurately identified segment boundaries are penalized equally regardless of their distance from the correct segment boundaries. In 1997, Beeferman[11] proposed the P_k metric to overcome the shortcoming. P_k is the probability that a randomly chosen pair of words with a distance of k words apart is incorrectly segmented[2]. P_k metric is defined as:

$$P_k = \sum_{1 \le i \le j \le K} D_k(i,j)(\delta_{ref}(i,j) \oplus \delta_{hyp}(i,j)) \ . \tag{9}$$

where $\delta_{ref}(i,j)$ is an indicator function whose value is one if sentences i and j belong to the same segment and zero otherwise. Similarly, $\delta_{hyp}(i,j)$ is one if the two sentences are hypothesized as belonging to the same segment and zero otherwise. The $\oplus$ operator is the XOR operator. The function D_k is the distance probability distribution that uniformly concentrates all its mass on the sentences which have a distance of k. The value of k is usually selected as half the average segment length. Low P_k value indicates high segmentation accuracy.

This error metric was recently criticized by Pevzner [12] to have several biased flaws such as penalizing missed boundaries more than erroneous additional boundaries and a new metric called *WindowDiff* was proposed:

$$WindowDiff(ref, hyp) = \frac{1}{K-k} \sum_{i=1}^{K-k} (\mid b(ref_i, ref_{i+k}) - b(hyp_i, hyp_{i+k}) \mid > 0) \ . \tag{10}$$

[1] www.lingware.co.uk/homepage/freddy.choi/index.htm

[2] We use the implementation of P_k in Choi's software package. (www.lingware.co.uk/homepage/freddy.choi/index.htm).

where *ref* is the correct segmentation for reference, *hyp* is the segmentation produced by the model, *K* is the number of sentences in the text, *k* is the size of the sliding window and *b(i, j)* is the number of boundaries between sentence *i* and sentence *j*. Low *WindowDiff* value indicates high segmentation accuracy. We will make comparison under both metrics (P_k and *WindowDiff*) on the testing corpus.

In experiments, punctuation marks and stopwords are removed. The Porter[15] stemming algorithm is applied to the remaining words to obtain word stems for English experiments.

4.2 Experimental Results

In MMS model, there are two parameters α and β that affect the quality of segmentation over certain ranges. To obtain the best parameter we randomly selected 50% corpus for training. The algorithm is run on the training corpus with α and β varies (the variation is 0.1 each time). Appropriate combination of α and β value is selected as the one which yields the minimum *WindowDiff* value. The rest of the corpus is used as testing corpus.

We evaluate MMS model in comparison to the C99 [6] method and Dotplotting[4] method including minimization algorithm (D_Min) and maximization algorithm (D_Max). The experimental results of the two methods come from Choi's software package, and it is an exact implementation of the published method[3]. Table 1 and Table 2 summarize the experimental results of all the algorithms on English corpus and Chinese corpus, respectively.

In the above tables, C_mS_n refers to the n^{th} section of the m^{th} chapter in the corpus. From experimental results we can see that our MMS model performs better with more than 6% reduction on average error rate (*WindowDiff*) for min and max Dotplotting methods and up to 6.4% for C99 on English corpus. Similar results are obtained on

Table 1. Comparative Results on English Testing Corpus

Method	P_k				*WindowDiff*			
	MMS	C99	D_Min	D_Max	MMS	C99	D_Min	D_Max
C_1S_2	**0.3023**	0.5763	0.3709	0.3636	**0.4318**	0.5836	0.4562	0.4586
C_2S_1	**0.3664**	0.3990	0.4385	0.4604	0.4578	**0.4488**	0.5197	0.5401
C_3S_2	0.4609	0.4771	**0.4500**	0.4785	**0.4609**	0.4897	0.5349	0.5634
C_4S_2	**0.3095**	0.4691	0.4525	0.3580	0.4330	0.4947	**0.4259**	0.4938
C_5S_1	**0.4105**	0.4958	0.4562	0.4547	**0.4407**	0.5196	0.5351	0.6129
C_5S_2	**0.3293**	0.4127	0.4515	0.3985	**0.3822**	0.4468	0.5312	0.5029
C_6	0.4240	0.4356	**0.3619**	0.4555	**0.4240**	0.4857	0.4722	0.4918
Average	**0.3718**	0.4665	0.4259	0.4242	**0.4329**	0.4956	0.4965	0.5234

[3] For the Dotplotting method, Choi[6] developed a package that includes both the implementation of the original Dotplotting method and his own interpretation. In this paper we cite the experimental results of the original Dotplotting method as published in [4]. The implementation comes from the publicly available software package. (www.lingware.co.uk/homepage/freddy.choi/index.htm)

Table 2. Comparative Results on Chinese Testing Corpus

Method	P_k				*WindowDiff*			
	MMS	C99	D_Min	D_Max	MMS	C99	D_Min	D_Max
C_2S_1	**0.3922**	0.4122	0.3946	0.3673	**0.4079**	0.4696	0.4492	0.4468
C_2S_3	**0.1947**	0.3344	**0.1447**	0.6006	**0.2047**	0.3427	0.2129	0.6272
C_4S_4	**0.2900**	0.4342	0.4806	0.5269	**0.3583**	0.4831	0.5320	0.5564
C_4S_5	**0.2820**	0.3658	0.4899	0.3416	**0.3030**	0.4391	0.5705	0.4439
C_5S_2	**0.3934**	0.4812	0.4158	0.4019	**0.4341**	0.5064	0.4994	0.5101
C_6S_2	**0.2279**	0.5785	0.5247	0.4487	**0.3145**	0.6537	0.5353	0.4611
C_6S_4	0.4157	**0.4002**	0.5305	0.5610	0.4824	**0.4188**	0.6054	0.6287
C_9S_1	**0.2248**	0.3120	0.3736	0.4136	**0.2960**	0.3992	0.4568	0.4816
$C_{10}S_2$	**0.4224**	0.5136	0.4699	0.4589	**0.4308**	0.5668	0.5372	0.5106
$C_{10}S_3$	0.3872	**0.2673**	0.4895	0.4697	0.4343	**0.2772**	0.5566	0.5665
$C_{10}S_5$	0.4235	**0.3848**	0.5359	0.5523	0.4345	**0.3867**	0.5404	0.6114
Average	**0.3322**	0.4077	0.4409	0.4675	**0.3728**	0.4494	0.4996	0.5313

Chinese corpus. MMS model achieves more than 12% average error rate reduction from Dotplotting methods and up to 7.7% for C99. The tables also indicate that *WindowDiff* metric penalizes errors more heavily than P_k metric. However the overall rank of the algorithms remains approximately the same on both metrics.

On both corpora MMS achieves best performance on most chapters (5 out of 7 for English corpus and 8 out of 11 for Chinese corpus). This is because more weighting factors affecting the segmentation choices are considered in our model. As previously mentioned, either within-segment or between-segment lexical similarity is examined in Dotplotting and C99 while our MMS model examines both factors. In addition, using text structure factors such as segment lengths and sentence distances also leads to an improvement.

The dynamic programming searching strategy adopted in our model is a global optimization algorithm. Compared to the divisive clustering algorithm of C99 and *dotplot* algorithm of Dotplotting, our strategy is more accurate and effective due to the global perspective of dynamic programming. With this strategy, the number of segments can be determined automatically when the best segmentation is achieved. In contrast, the number of segments has to be given in advance for Dotplotting method because it cannot decide when to stop inserting boundaries. The same problem exists in C99 method. Although the author proposed an algorithm to determine the number of segments automatically, this algorithm has some negative influence on the performance of the method[6].

MMS model remedies two problems of Dotplotting. Ye [16] reported analysis of two problems in Dotplotting's segmentation evaluating function:

$$f_D = \sum_{i=2}^{N} \frac{V_{p_{i-1},p_i} \cdot V_{p_i,K}}{(p_i - p_{i-1})(K - p_i)} . \tag{11}$$

where K is the end of the whole text, and $V_{x,y}$ is a vector containing the word counts associated with word positions x through y in the article.

First, the above function is asymmetric. If the text is scanned from the end to the start, a "backward" function will be got in a form different from Eq. 11. This problem leads to the apparent illogical phenomenon that forward scan may result in different

segmentation with backward scan. In MMS model, the segmentation criterion function (Eq. 7) is symmetrized.

Secondly, while determining the next boundary, the evaluating strategy of Dotplotting does not adequately take the previously located boundaries into account. For each candidate boundary p_i being examined, only the segment boundary before it (p_{i-1}) is taken into consideration, and may work less effectively because it ignores the restriction of the segment boundary after it (p_{i+1}). In our MMS model, the restrictions of adjacent segment boundaries on both sides are considered. From the within-segment and between-segment similarity function (Eq. 4 and Eq. 5) we can see the segmentation evaluating function value after locating a boundary at p_i is determined by p_{i-1} and p_{i+1}. In this way the restriction from the previously located boundaries is strengthened. The optimization process of dynamic programming also helps to select boundaries globally.

5 Conclusion and Future Work

In this paper we presented a dynamic programming model for text segmentation. This model attempts to simultaneously maximizing within-segment and minimizing between-segment similarity. An analytical form of the segmentation evaluation function is given and a complete text segmentation algorithm using two-dimension dynamic programming searching scheme is described. In addition, other text structure factors, such as segment length and sentence distance, are also incorporated in the model to capture subtopic changes.

Experimental results on the public available real corpora are provided and compared with popular systems. The MMS model is shown to be promising and effective in text segmentation that it outperforms all other systems in most testing data sets. In comparing with the best comparable system (C99), the MMS model has achieved a reduction of more than 6% in average error rate (*WindowDiff* metric).

In the future we plan to optimize our algorithm by incorporating more features of the subtopic distribution and text structure. It is demonstrated in [17] that semantic information trained from background corpus can help improve text segmentation performance. We will also consider introducing semantic knowledge in the model. Besides, the length factor in our model is in a simple form and more adequate segment length factor needs to be investigated. Applying the algorithm to other text segmentation tasks such as news stream and conversation segmentation is also an important research topic.

Acknowledgments. This work was supported in part by the National Natural Science Foundation of China under Grant No.60473140, the National 863 High-tech Project No. 2006AA01Z154 ; the Program for New Century Excellent Talents in University No. NCET-05-0287; and the National 985 Project No.985-2-DB-C03 .

References

1. Boguraev, B.K., Neff, M.S.: Discourse segmentation in aid of document summarization. In: 33rd Hawaii International Conference on System Sciences, Hawaii (2000)
2. Hearst, M.A.: Multi-paragraph segmentation of expository text. In: 32nd Annual Meeting of the Association for Computational Linguistics, pp. 9--16, New Mexico (1994)

3. Brants, T., Chen, F., Tsochantarides, I.: Topic-based document segmentation with probabilistic latent semantic analysis. In: 11th International Conference on Information and Knowledge Management, pp.211--218, Virginia (2002)
4. Reynar, J.C.: An automatic method of finding topic boundaries. In: 32nd Annual Meeting of the Association for Computational Linguistics, pp. 331--333, New Mexico (1994)
5. Heinonen, O.: Optimal Multi-Paragraph Text Segmentation by Dynamic Programming. In: 17th International Conference on Computational Linguistics, pp. 1484--1486, Montreal (1998)
6. Choi, F.Y.Y.: Advances in domain independent linear text segmentation. In: 1st Meeting of the North American Chapter of the Association for Computational Linguistics, pp. 26-33, Washington (2000)
7. Choi, F.Y.Y., Wiemer-Hastings, P., Moore, J.: Latent Semantic Analysis for Text Segmentation. In: 6th Conference on Empirical Methods in Natural Language Processing, pp. 109--117, Pennsylvania (2001)
8. Utiyama, M., Isahara, H. : A Statistical Model for Domain-Independent Text Segmentation. In: 9th Conference of the European Chapter of the Association for Computational Linguistics, pp. 491--498, Bergen (2001)
9. Ji, X., Zha, H.: Domain-independent Text Segmentation Using Anisotropic Diffusion and Dynamic Programming. In: 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 322--329, Toronto (2003)
10. Fragkou, P., Petridis, V., Kehagias, A.: A Dynamic Programming Algorithm for Linear Text Segmentation. Journal of Intelligent Information Systems, Vol. 23, pp. 179--197 (2004)
11. Beeferman, D., Berger, A., Lafferty, J.: Text Segmentation Using Exponential Models. In : 2nd Conference on Empirical Methods in Natural Language Processing, pp. 35--46, Rhode Island (1997)
12. Pevzner, L., Hearst, M.: A Critique and Improvement of an Evaluation Metric for Text Segmentation, Computational Linguistics, Vol. 28, pp.19--36 (2002)
13. Malioutov, I., Barzilay, R.: Minimum Cut Model for Spoken Lecture Segmentation. In: 21st International Conference on Computational Linguistics, pp. 25--32, Sidney (2006)
14. Zhu, J.B., Ye, N., Chang, X.Z., Chen, W.L., Tsou, B.K.: Using Multiple Discriminant Analysis Approach for Linear Text Segmentation. In: 2nd International Joint Conference on Natural Language Processing, pp. 292--301, Jeju (2005)
15. Porter, M.F.: An Algorithm for Suffix Stripping. Program, Vol. 14, pp. 130--137 (1980)
16. Ye, N., Zhu, J.B., Luo, H.T., Wang H.Z., Zhang, B.: Improvement of the dotplotting method for linear text segmentation. In: 2005 IEEE International Conference on Natural Language Processing and Knowledge Engineering, pp. 636--641, Wuhan (2005)
17. Bestgen, Y.: Improving Text Segmentation Using Latent Semantic Analysis: A Reanalysis of Choi, Wiemer-Hastings, and Moore (2001). Computational Linguistics, Volume 32, pp. 5--12 (2006)