Topic Segmentation of Web Documents with Automatic Cue Phrase Identification and BLSTM-CNN

Liang Wang[1], Sujian Li[1,3], Xinyan Xiao[2], and Yajuan Lyu[2]

[1] Key Laboratory of Computational Linguistics, Peking University, MOE, China
[2] Baidu Inc., Beijing, China
[3] Collaborative Innovation Center for Language Ability, Xuzhou, Jiangsu, China
{intfloat,lisujian}@pku.edu.cn, {xiaoxinyan,lvyajuan}@baidu.com

Abstract. Topic segmentation plays an important role for discourse analysis and document understanding. Previous work mainly focus on unsupervised method for topic segmentation. In this paper, we propose to use bidirectional long short-term memory(BLSTM) model, along with convolutional neural network(CNN) for learning paragraph representation. Besides, we present a novel algorithm based on frequent subsequence mining to automatically discover high-quality cue phrases from documents. Experiments show that our proposed model is able to achieve much better performance than strong baselines, and our mined cue phrases are reasonable and effective. Also, this is the first work that investigates the task of topic segmentation for web documents.

Keywords: topic segmentation, neural network, web documents, sequence mining

1 Introduction

Topic segmentation is a natural language processing(NLP) task, which aims to segment a document into topically similar parts, it is also called text segmentation or discourse segmentation in various scenarios. This level of analysis provides a better understanding about document structure and topic shift, which are helpful information for many NLP tasks such as discourse parsing, dialogue generation etc.

There have been decades of research about topic segmentation, previous work mainly focus on unsupervised approach. TextTiling[12] is one of the most famous and earliest algorithms for topic segmentation. It is based on one simple intuition: lexical cohesion within each topic segment is high, while lexical cohesion between different topic segments is low. Therefore, the core part of TextTiling algorithm is to calculate lexical similarity of adjacent segments and then choose an appropriate threshold to determine topic boundaries. This algorithm is simple and computationally efficient, however, when annotated corpus is available, it fails to utilize training data and unable to learn an accurate model. Other unsupervised variants of TextTiling algorithm such as C99[6] and TopicTiling[18] also suffer from this issue.

On the contrary, supervised approach is able to learn more complex and accurate model. As long as training data is sufficient and feature set is good enough, its performance is much better than unsupervised approaches. In this paper, we conduct ex-

periments with conditional random field(CRF) and LSTM, both of them significantly outperform TextTiling algorithm.

For sequence modeling task such as topic segmentation, capturing long distance information is a key issue. Thanks to the gating mechanism in LSTM, it preserves useful information for a long time period. Bidirectional LSTM is a combination of forward LSTM and backward LSTM, therefore it is able to exploit useful features from both sides. Our experiments show that BLSTM consistently beats CRF and LSTM, and achieves best f1-score.

Different from previous work which focus on news or scientific documents, in our work, we conduct text segmentation for web documents. According to our observation, cue phrase is a strong indicator for topic boundary. For example, "第一", "第二" are frequently used to start a new topic. [9] presented a bayesian framework to identify cue phrases automatically. Different from their work, we treat cue phrase identification as a frequent subsequence mining problem, and come up with a variant of *Apriori* algorithm to mine cue phrases from annotated corpus. It turns out that our proposed algorithm is able to precisely locate those cue phrases and therefore boost system performance.

This paper makes three major contributions: 1. To the best of our knowledge, this is the first work that successfully applies neural network model for supervised topic segmentation and achieves promising results; 2. We present a novel algorithm based on frequent subsequence mining to identify cue phrases; 3. For the first time, the possibility of topic segmentation for web documents is examined.

2 Related Work

Due to the lack of large scale high-quality topic segmentation datasets, unsupervised approach is most widely adopted. Two of the early algorithms are TextTiling[12] and C99[6], both of which are based on the intuition that word distributions differ significantly if there is a transition in topic. TextTiling is more computationally efficient while C99 algorithm shows better performance. Vector space model is used to compute cosine similarity of sentences, but it fails to capture semantic similarity of different words. Later work used Latent Semantic Analysis(LSA)[7] and Latent Dirichlet Allocation(LDA)[18] [16] to compute sentence similarity more accurately.

Generative models were also presented to improve performance of topic segmentaion system. Similar to LDA, topics are seen as latent variables and words are seen as visible variables. Hidden Markov Model(HMM)[20] and several variants of LDA[8] [14] [17] were proposed. Carefully designed generative models outperform lexical similarity based models, however they are usually much more complicated and require efficient inference algorithms.

Supervised approach is also examined when large amount of training data is available. Therein, topic segmentation is formulated as a binary classification task. [10] trained decision tree classifier on a rich set of features such as cue phrases and lexical cohesion. Support Vector Machine(SVM)[11] was also tried on datasets from different domains. Experiment results showed its superiority over unsupervised models.

Evaluation of topic segmentation systems is non-trivial. Classification-based metrics such as precision, recall and f1-score are useful but sometimes too strict. P_k metric

was proposed by [2] to alleviate this problem. To calculate P_k, we need a sliding window of fixed length, and check whether the predicted segmentation is consistent with ground truth within this window. Now P_k is the most widely used metric within the literature of topic segmentation. However, P_k metric also has its own problems, WindowDiff(WD)[15] and word error rate based[3] metrics were proposed in later research.

LSTM[13] is a variant of vanilla recurrent neural network(RNN), which aims to solve gradient vanishing/exploding issue during training and allow useful information to flow over a long distance. It has been widely used for sequence modeling tasks such as word segmentation[4], named entity recognition[5] and Part-of-Speech tagging[19] etc.

3 Models

3.1 BLSTM(Bidirectional Long Short Term Memory)

LSTM is a recurrent network with gating mechanism. There are many variants of LSTM unit, here we adopt one widely used architecture with three types of gates: input gate $\mathbf{i}^t$, forget gate $\mathbf{f}^t$ and output gate $\mathbf{o}^t$, t denotes time step. The formula for calculating each gate and memory cell unit are as follows:

$$\mathbf{i}^t = tanh(\mathbf{W}_i\mathbf{x}^t + \mathbf{R}_i\mathbf{y}^{t-1} + \mathbf{b}_i) \tag{1}$$
$$\mathbf{f}^t = tanh(\mathbf{W}_f\mathbf{x}^t + \mathbf{R}_f\mathbf{y}^{t-1} + \mathbf{b}_f) \tag{2}$$
$$\mathbf{o}^t = tanh(\mathbf{W}_o\mathbf{x}^t + \mathbf{R}_o\mathbf{y}^{t-1} + \mathbf{b}_o) \tag{3}$$
$$\mathbf{z}^t = tanh(\mathbf{W}_z\mathbf{x}^t + \mathbf{R}_z\mathbf{y}^{t-1} + \mathbf{b}_z) \tag{4}$$
$$\mathbf{c}^t = \mathbf{i}^t \odot \mathbf{z}^t + \mathbf{f}^t \odot \mathbf{c}^{t-1} \tag{5}$$
$$\mathbf{y}^t = \mathbf{o}^t \odot tanh(\mathbf{c}^t) \tag{6}$$

Here $\mathbf{x}^t \in \mathbb{R}^d$, $\mathbf{y}^t \in \mathbb{R}^d$ are d dimensional input vectors and output vectors. $\mathbf{c}^t$ is the memory cell vector at time step t. $\mathbf{W}_z$, $\mathbf{W}_i$, $\mathbf{W}_f$, $\mathbf{W}_o$ are weight matrices for input. $\mathbf{R}_z$, $\mathbf{R}_i$, $\mathbf{R}_f$, $\mathbf{R}_o$ are weight matrices for output. $\mathbf{b}_z$, $\mathbf{b}_i$, $\mathbf{b}_f$, $\mathbf{b}_o$ are corresponding bias. $tanh$ is used as non-linear activation function.

Compared to LSTM, bidirectional LSTM captures information from both directions of a sequence. It is a combination of two independent LSTM with opposite directions: forward LSTM and backward LSTM. The final output vector $\mathbf{y}^t$ is concatenation of these two LSTM.

Below shows BLSTM's updating formula for memory cell $\mathbf{c}^t$, output gate $\mathbf{o}^t$, we omit other gates for simplicity as they are similar.

$$\mathbf{c}_k^t = \mathbf{i}_k^t \odot \mathbf{z}_k^t + \mathbf{f}_k^t \odot \mathbf{c}_k^{t-1},\ k \in \{f, b\} \tag{7}$$
$$\mathbf{o}_k^t = tanh(\mathbf{W}_o^k\mathbf{x}^t + \mathbf{R}_o^k\mathbf{y}_k^{t-1} + \mathbf{b}_o^k),\ k \in \{f, b\} \tag{8}$$
$$\mathbf{y}_k^t = \mathbf{o}_k^t \odot tanh(\mathbf{c}_k^t),\ k \in \{f, b\} \tag{9}$$
$$\mathbf{y}^t = [\mathbf{y}_f^t,\ \mathbf{y}_b^t] \tag{10}$$

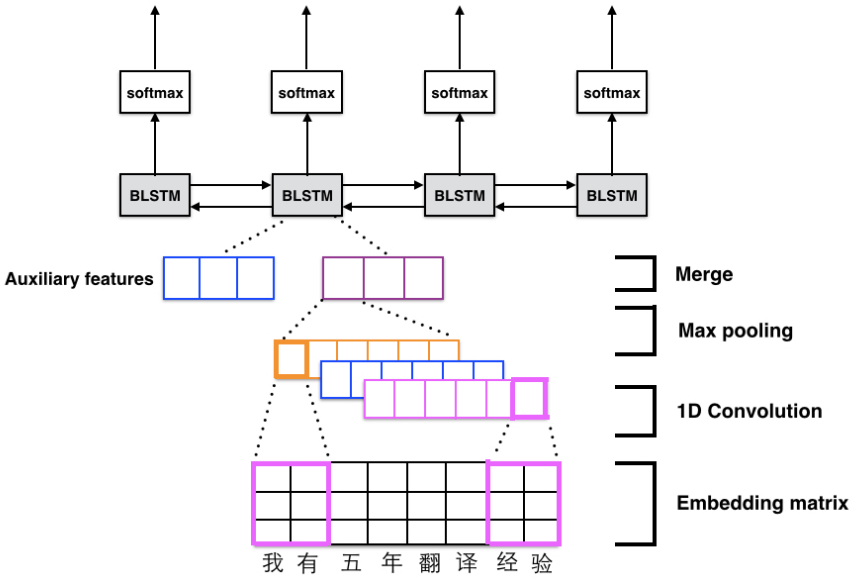

Fig. 1: BLSTM-CNN for topic segmentation
Example text: 我有五年翻译经验(I have five years of experience as a translator.)

Among those equations, f denotes forward pass layer, b denotes backward pass layer, $\mathbf{y}^t$ is a concatenation of $\mathbf{y}_f^t$, $\mathbf{y}_b^t \in \mathbb{R}^d$ and therefore $\mathbf{y}^t \in \mathbb{R}^{2d}$.

For our classification task, there is a softmax layer over output vectors:

$$\mathbf{P}_t(y|x) = softmax(\mathbf{W}_h \mathbf{y}^t + \mathbf{b}_h) \tag{11}$$

$$\mathbf{Y}_{pred} = argmax\ \mathbf{P}_t(y|x) \tag{12}$$

The final prediction is the label with highest probability $argmax\ \mathbf{P}_t(y|x)$.

3.2 CNN for paragraph representation

There are many ways to represent paragraph text, one-hot encoding representation would result extremely sparse feature vector. In this paper, we adopt a popular CNN architecture to represent entire paragraph as a low dimensional dense vector. In other words, CNN serves as a text feature extractor for BLSTM model.

The input to CNN is a word sequence, each word w is mapped to an embedding vector $\mathbf{x}_w$ by matrix-vector product:

$$\mathbf{x}_w = \mathbf{W}^{d \times |V|} \mathbf{v}^w \tag{13}$$

$\mathbf{W}^{d \times |V|}$ is a d dimensional word embedding matrix for entire vocabulary V, $\mathbf{v}^w$ is one-hot vector representation of word w.

The output of embedding layer is a word embedding sequence $\mathbf{x}_{1:n}$.

$$\mathbf{x}_{1:n} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n] \tag{14}$$

Every convolution operation involves applying a filter $\mathbf{v} \in \mathbb{R}^{hd}$ to a window of h words to produce a new feature. For example, a feature f_i is generated from a window of words $\mathbf{x}_{i:i+h-1}$ by

$$f_i = h(\mathbf{v} \cdot \mathbf{x}_{i:i+h-1} + b). \tag{15}$$

$b \in \mathbb{R}$ is a bias term, h is a non-linear transformation function such as $tanh$. This operation is applied to every possible window $\{\mathbf{x}_{1:h}, \mathbf{x}_{2:h+1}, \ldots, \mathbf{x}_{n-h+1:n}\}$ to produce a feature map:

$$\mathbf{f} = [f_1, f_2, \ldots, f_{n-h+1}], \tag{16}$$

with $\mathbf{f} \in \mathbb{R}^{n-h+1}$. Max-pooling operation applies to the entire feature map and chooses the maximum value as the feature.

$$\hat{f} = \max\{\mathbf{f}\} \tag{17}$$

The overall network architecture is shown in Figure 1. Parameters of CNN and BLSTM are jointly learned. The final output of CNN is a vector representation of given paragraph.

3.3 Model Learning

We formulate topic segmentation as a binary classification task, and use cross entropy loss function:

$$J = -\frac{1}{N}\sum_{i=1}^{N}(y^* \log y + (1 - y^*)\log(1 - y)) \tag{18}$$

where N represents the size of training set; y^* is the ground truth label, y is model's probability output.

To train our network, we use mini-batch stochastic gradient descent(SGD) with adaptive learning rate computed by *Adadelta*[21], which shows better performance and convergence property.

4 Features

4.1 Frequent subsequence mining based cue phrase identification

When writing articles, people often use cue phrase to start a new topic, such as "首先", "然后", "最后". They are strong indicators for detecting topic boundary. Collecting those cue phrases by hand would be time-consuming. Moreover, cue phrases are often dependent on corpus domain and language. If we transplant our system to a new domain, we have to manually summarize cue phrases all over again.

In this paper, we propose a novel algorithm to automatically discover cue phrases base on frequent subsequence mining. It is a variant of the famous *Apriori* algorithm[1]

Algorithm 1 Cue Phrase Sequence Mining Algorithm
Input: Corpus D = {d | d is a document},
$minsup$: minimum support to become a frequent subsequence,
$maxlen$: maximum length of cue phrase sequence
Output: a list of cue phrase sequence S = { p | p is a cue phrase sequence}

function MINE(D, $minsup$, $maxlen$)
  $C_1 \leftarrow$ count each word $w \in D$
  $P_1 \leftarrow$ {w | w.count $\geq$ *minsup*}
  S $\leftarrow$ {}
  for $i \leftarrow 2$ to *maxlen* **do**
    $C_i \leftarrow$ CANDIDATE-GEN(P_{i-1}, i - 1)
    for *candidate* in C_i **do**
      for each document $d \in D$ **do**
        if IS-SUBSEQUENCE(*d*, *candidate*) **then**
          candidate.count++
    $P_i \leftarrow$ {*candidate* | *candidate*.count $\geq$ *minsup* }
  S $\leftarrow \cup \{P_1, P_2 \ldots P_{maxlen}\}$
  return S

function CANDIDATE-GEN(C_i, len)
  candidates $\leftarrow$ {}
  for t_a in C_i **do**
    for t_b in C_i **do**
      if t_a[1:len] == t_b[0:(len - 1)] and CO-OCCURRENCE(t_a, t_b) $\geq$ *minsup* **then**
        candidates $\leftarrow$ *candidates* $\cup$ (t_a[1:len] + t_b[len - 1])
  return $candidates$

for frequent itemset mining, with one key difference that itemset is unordered while cue phrase sequence is ordered. The intuition behind our algorithm is that cue phrase sequence usually appear more often at topic boundary than other words.

Our proposed algorithm is summarized above.

We omit the implementation details of two functions: *IS-SUBSEQUENCE(d, candidate)* and *CO-OCCURRENCE(t_a, t_b)*. *IS-SUBSEQUENCE(d, candidate)* checks whether given *candidate* is a subsequence of given document *d*. We adopt two sequence matching strategies: prefix matching and suffix matching. In prefix matching strategy, a candidate cue phrase w matches a paragraph p when w is a prefix of p; Similarly, in suffix matching strategy, a candidate cue phrase w matches a paragraph p when w is a suffix of p. *CO-OCCURRENCE(t_a, t_b)* calculates the number of document d that both *IS-SUBSEQUENCE(d, t_a)* and *IS-SUBSEQUENCE(d, t_b)* evaluate to *true*.

This is an iterative algorithm, which involves two key steps at each iteration: candidate generation and candidate validation. In candidate generation step, for each pair of length len cue phrase sequence t_a and t_b, if the $(len-1)$-suffix of t_a is equal of $(len-1)$-prefix of t_b, and their co-occurrence count is no less than *minsup*, then t_a and t_b can be combined into a length $len+1$ candidate. The co-occurrence constraint is not necessary, but it can greatly reduce the number of candidates. In our experiments,

it reduces the number of length-2 candidates from over 20,000 down to less than 300. In candidate validation step, for each *candidate*, the algorithm calculates in how many documents this *candidate* sequence appears, then candidates whose frequency is no less than $minsup$ get into the final result set S.

The worst time complexity of our proposed algorithm is exponential, however, the number of cue phrases in real dataset is often limited. Our python implementation without any further optimization finishes within 10 seconds.

Cue Phrase Sequence	1.xxx 2.xxx 3.xxx	一、xxx 二、xxx 三、xxx	工具xxx 方法xxx 注意事项xxx	xxx介绍 xxx介绍 xxx介绍
English Explanation	first.xxx second.xxx third.xxx	first.xxx second.xxx third.xxx	tools xxx method xxx precautions xxx	xxx introduction xxx introduction xxx introduction

Table 1: Examples of cue phrase sequence, "xxx" denotes some other irrelevant words

Some of the cue phrase sequences are listed in Table 1. It is clear that our algorithm is able to find out high quality cue phrases. As expected, people often use number sequence to start a new topic. The last two are due to the large number of tutorial documents on web, and they are much less common in other domains. Handcrafted cue phrase set may very likely miss them. Compared to time-consuming and possibly incomplete manual cue phrase selection process, our algorithm is more accurate, efficient and can be easily adapted to other domains.

4.2 Other features

Besides cue phrase, here is a list of other features we used in experiments.

1. **Lexical feature.** Paragraph text is encoded as a dense feature vector via CNN model described above.
2. **Part-of-Speech(POS) feature.** The POS tags of words in current paragraph. We perform word segmentation and POS tagging with open source library *jieba*[4], the same is true for other features related to word segmentation.
3. **Length feature.** The number of characters and words in current paragraph, previous paragraph and next paragraph. It also includes the number of paragraph in current document.
4. **Position feature.** Whether current paragraph is the document's first paragraph or last paragraph.
5. **Hyperlink feature.** Whether current paragraph contains text with embedded hyperlink.

[4] https://github.com/fxsjy/jieba

6. **Text font feature.** Whether current paragraph contains text with bold or italic font. For web documents, the first paragraph of a new topic often contains such text, they can be useful information.

Many other types of features are also examined, including LDA features and syntactic features, but they show no performance gain, therefore we choose to not list them here.

5 Experiments

5.1 Data and Setup

Our dataset is provided by Baidu[5] and consists of 2951 web documents with human annotated ground truth labels. To the best of our knowledge, this is the largest human annotated topic segmentation dataset. Dataset used in previous research are either much smaller or constructed automatically with the help of some heuristics. Table 2 shows some statistics for our datasets.

Number of Documents	Average Number of Paragraphs	Average Number of Topics
2951	15.75	2.33

Table 2: Statistics for our topic segmentation dataset.

Dataset is randomly split into training set(70%), validation set(10%), test set(20%). Hyperparameters are chosen via grid search by maximizing f1-score on validation set. Once hyperparameters are fixed, we train on both training set and validation set, then report model's performance on test set.

Our implementation of BLSTM is based on open source library *keras*[6]. We use *Adadelta*[21] to compute learning rate. The dimension of memory cell is set to 50, mini-batch size is 16. To combat overfit, we add one dropout layer above BLSTM output, dropout probability is set to 0.5. For CNN, the number of filters is set to 150, the window size is set to 4 for 1D convolution, and the size of word embedding d is set to 32, word embedding matrix $\mathbf{W}^{d\times|V|}$ is initialized with uniform random values from $[-0.5, 0.5]$.

As comparison, we also implemented some other algorithms such as TextTiling, CRF[7]. Similar to BLSTM, hyperparameters are chosen according to validation set.

To have a comprehensive comparison of model's effectiveness, we evaluate on multiple metrics: precision, recall, f1-score and P_k.

5.2 Results

Table 3 shows model's performance with all features, except TextTiling algorithm is unsupervised and doesn't need any feature. Notice that for P_k metric, smaller value means better performance.

[5] Not publicly available for now.
[6] https://github.com/fchollet/keras
[7] https://github.com/tpeng/python-crfsuite

Model	Precision	Recall	F1-score	P_k
TextTiling	0.762	0.448	0.565	0.146
CRF	**0.859**	0.624	0.723	0.133
LSTM-CNN	0.786	0.716	0.750	0.092
BLSTM-CNN	0.829	**0.730**	**0.776**	**0.075**

Table 3: Performance comparison of different models.

We can clearly see that supervised models significantly outperform unsupervised TextTiling algorithm on every metric. CRF is a widely used model for sequence labeling, it gets highest precision, while performs much worse than our neural network models on other metrics. LSTM-CNN is a forward LSTM stacking with CNN, all other parameters are same with BLSTM-CNN. Its f1-score is 2.6% lower than BLSTM-CNN, and P_k metric 0.017 higher than BLSTM-CNN. This performance gap implies that being able to capture information from both left and right is helpful to do topic segmentation. BLSTM-CNN achieves the best overall performance, highest recall, highest f1-score and lowest P_k.

Feature Set	Precision	Recall	F1-score	P_k
1+2+3+4	0.831	0.666	0.739	0.095
1+2+3+4+5	**0.850**	0.669	0.749	0.085
1+2+3+4+5+6	0.835	0.708	0.767	0.079
1+2+3+4+5+6+Cue Phrase	0.829	**0.730**	**0.776**	**0.075**

Table 4: Comparison of BLSTM-CNN performance on different feature set.

To examine the effects of different features, we conduct a series of experiments with BLSTM-CNN model. The results are shown in Table 4. For the mapping relations between arab number and feature name, please refer to section 4.2.

Feature 1 to 4 are called basic features, in the sense that they are shared across documents in all domains, not just web documents. Experiments show that basic features are already enough to deliver a competitive result. Hyperlink(feature 5), bold and italic text(feature 6) are unique characteristics for web documents. Incorporating these two features results in better performance, f1-score goes up by 2.8% and P_k value goes down by 0.016. Table 4 also shows that our frequent subsequence mining based cue phrase identification algorithm is crucial to further boost system performance.

5.3 Error Analysis

By analyzing bad cases, we find there are two major types of document structure that our model performs poorly: the document with hierarchical topic structure or implicit topic structure.

– **Hierarchical topic structure.** When we formulate topic segmentation as binary classification task, we actually make an implicit assumption that document topics

have a linear structure. However, some documents have hierarchical topic structure. Such document contains several major topics, and each major topic contains many subtopics. For example, one document describes how to properly configure a computer, it involves hardware configuration topic and software installation topic. Within software installation topic, it contains many subtopics about how to install different softwares. Our model has trouble with determining the granularity of topics.

– **Implicit topic structure.** Cue phrase is a useful feature to identify topic boundary. However, sometimes people starts a new topic without using any cue phrase, and the lexical distribution between topics has no obvious difference. For example, one document contains two topics: one topic is about positive effects of NATO[8], the other one is about negative effects of NATO. There is a significant lexical overlap between these two topics, and our model fails to recognize the transition of underlying topic.

To handle document with hierarchical topic structure, our model need to have a better understanding of document's global structure, rather than merely focus on local structure; for document with implicit topic structure, more accurate semantic analysis algorithms are needed to detect topic boundary.

6 Conclusion and Future Work

In this paper, we propose to use BLSTM stacking with CNN to do topic segmentation of web documents. CNN enables efficient and effective paragraph representation learning, while BLSTM manages to capture and preserve useful information from both directions. Based on the characteristics of web documents, a frequent subsequence mining based cue phrase identification algorithm is presented to identify cue phrases automatically. Experiments show that our BLSTM-CNN model combined with cue phrase feature is able to achieve much better performance than strong baseline models.

For future work, we would like to verify our model's effectiveness on other domains and other languages. Also, other network architectures will be examined to further improve the performance of our topic segmentation system.

Acknowledgements

We thank all the anonymous reviewers for their insightful comments on this paper. This work was partially supported by Baidu-Peking University joint project, and National Natural Science Foundation of China (61273278 and 61572049). The correspondence author of this paper is Sujian Li.

References

1. Agrawal, R., Srikant, R., et al.: Fast algorithms for mining association rules. In: Proc. 20th int. conf. very large data bases, VLDB. vol. 1215, pp. 487–499 (1994)

[8] North Atlantic Treaty Organization

2. Beeferman, D., Berger, A., Lafferty, J.: Statistical models for text segmentation. Machine learning 34(1-3), 177–210 (1999)
3. Carroll, L.: Evaluating hierarchical discourse segmentation. In: Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics. pp. 993–1001. Association for Computational Linguistics (2010)
4. Chen, X., Qiu, X., Zhu, C., Liu, P., Huang, X.: Long short-term memory neural networks for chinese word segmentation. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing (2015)
5. Chiu, J.P., Nichols, E.: Named entity recognition with bidirectional lstm-cnns. arXiv preprint arXiv:1511.08308 (2015)
6. Choi, F.Y.: Advances in domain independent linear text segmentation. In: Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference. pp. 26–33. Association for Computational Linguistics (2000)
7. Choi, F.Y., Wiemer-Hastings, P., Moore, J.: Latent semantic analysis for text segmentation. In: In Proceedings of EMNLP. Citeseer (2001)
8. Du, L., Buntine, W.L., Johnson, M.: Topic segmentation with a structured topic model. In: HLT-NAACL. pp. 190–200 (2013)
9. Eisenstein, J., Barzilay, R.: Bayesian unsupervised topic segmentation. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing. pp. 334–343. Association for Computational Linguistics (2008)
10. Galley, M., McKeown, K., Fosler-Lussier, E., Jing, H.: Discourse segmentation of multi-party conversation. In: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1. pp. 562–569. Association for Computational Linguistics (2003)
11. Georgescul, M., Clark, A., Armstrong, S.: Word distributions for thematic segmentation in a support vector machine approach. In: Proceedings of the Tenth Conference on Computational Natural Language Learning. pp. 101–108. Association for Computational Linguistics (2006)
12. Hearst, M.A.: Texttiling: Segmenting text into multi-paragraph subtopic passages. Computational linguistics 23(1), 33–64 (1997)
13. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation 9(8), 1735–1780 (1997)
14. Jameel, S., Lam, W.: An unsupervised topic segmentation model incorporating word order. In: Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval. pp. 203–212. ACM (2013)
15. Pevzner, L., Hearst, M.A.: A critique and improvement of an evaluation metric for text segmentation. Computational Linguistics 28(1), 19–36 (2002)
16. Riedl, M., Biemann, C.: How text segmentation algorithms gain from topic models. In: Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. pp. 553–557. Association for Computational Linguistics (2012)
17. Riedl, M., Biemann, C.: Text segmentation with topic models. Journal for Language Technology and Computational Linguistics 27(1), 47–69 (2012)
18. Riedl, M., Biemann, C.: Topictiling: a text segmentation algorithm based on lda. In: Proceedings of ACL 2012 Student Research Workshop. pp. 37–42. Association for Computational Linguistics (2012)
19. Wang, P., Qian, Y., Soong, F.K., He, L., Zhao, H.: Part-of-speech tagging with bidirectional long short-term memory recurrent neural network. arXiv preprint arXiv:1510.06168 (2015)
20. Yamron, J.P., Carp, I., Gillick, L., Lowe, S., van Mulbregt, P.: A hidden markov model approach to text segmentation and event tracking. In: Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on. vol. 1, pp. 333–336. IEEE (1998)

21. Zeiler, M.D.: Adadelta: an adaptive learning rate method. arXiv preprint arXiv:1212.5701 (2012)