SEGBOT: A Generic Neural Text Segmentation Model with Pointer Network

Jing Li, Aixin Sun and **Shafiq Joty**
School of Computer Science and Engineering, Nanyang Technological University, Singapore
jli030@e.ntu.edu.sg, {axsun,srjoty}@ntu.edu.sg

Abstract

Text segmentation is a fundamental task in natural language processing that comes in two levels of granularity: (i) segmenting a document into a sequence of topical segments (topic segmentation), and (ii) segmenting a sentence into a sequence of elementary discourse units (EDU segmentation). Traditional solutions to the two tasks heavily rely on carefully designed features. The recently proposed neural models do not need manual feature engineering, but they either suffer from sparse boundary tags or they cannot well handle the issue of variable size output vocabulary. We propose a generic end-to-end segmentation model called SEGBOT. SEGBOT uses a bidirectional recurrent neural network to encode input text sequence. The model then uses another recurrent neural network together with a pointer network to select text boundaries in the input sequence. In this way, SEGBOT does not require hand-crafted features. More importantly, our model inherently handles the issue of variable size output vocabulary and the issue of sparse boundary tags. In our experiments, SEGBOT outperforms state-of-the-art models on both topic and EDU segmentation tasks.

1 Introduction

Text segmentation has been a fundamental task in natural language processing (NLP) that has been addressed at different levels of granularity. At a coarser level, text segmentation generally refers to breaking a document into a sequence of topically coherent segments, often known as **topic segmentation** [Hearst, 1997; Choi, 2000]. Topic segmentation is often considered as a pre-requisite for other higher level discourse analysis tasks like discourse parsing [Joty *et al.*, 2015], and has been shown to support a number of downstream NLP applications including text summarization and passage retrieval [Riedl and Biemann, 2012]. At a finer level, text segmentation refers to breaking each sentence into a sequence of elementary discourse units (EDUs), often known as **EDU segmentation** [Marcu, 2000]. As exemplified in Figure 1, EDUs are clause-like units that serve as building blocks for discourse parsing in Rhetorical Structure Theory [William and Thompson, 1988].

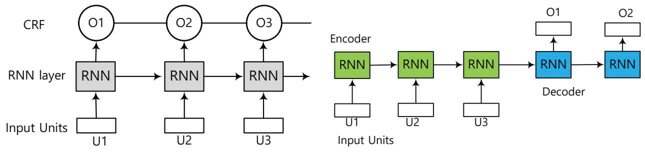
[A person]$_{\text{EDU}}$ [who never made a mistake]$_{\text{EDU}}$ [never tried anything new]$_{\text{EDU}}$

Figure 1: A sentence with three elementary discourse units (EDUs).

Both topic and EDU segmentation tasks have received a lot of attention in the past due to their utility in many NLP tasks. Although related, these two tasks have been addressed separately with different sets of approaches. Both supervised and unsupervised methods have been proposed for topic segmentation. Unsupervised topic segmentation models exploit the strong correlation between topic and lexical usage, and can be broadly categorized into two classes: *similarity-based* models and probabilistic *generative* models. The similarity-based models are based on the key intuition that sentences in a segment are more similar to each other than to sentences in the preceding or the following segment. Examples of this category are TextTiling [Hearst, 1997], C99 [Choi, 2000], and LCSeg [Galley *et al.*, 2003]. Probabilistic generative models are based on the intuition that a discourse is a hidden sequence of topics, each of which has its own characteristic word distribution. Variants of Hidden Markov Models (HMMs) and Latent Dirichlet Allocations (LDAs) fall into this class. Supervised topic segmentation models are more flexible in using more features (*e.g.,* cue phrases, length and similarity scores) and generally perform better than unsupervised models, however, come with the price of efforts to manually design informative features and to annotate large amount of data. Successful methods for EDU segmentation are mostly supervised, and they use lexical and syntactic features [Soricut and Marcu, 2003].

While most existing topic segmentation methods use lexical similarity based on surface terms (*i.e.,* words), it is now generally admitted that lexical semantics are better captured with distributed representations [Goldberg, 2017]. Furthermore, existing supervised models, for both topic and EDU segmentation, require a large set of features manually designed for each task and domain, which demands task and domain expertise. We would envision for a system that is based on distributed representation, and that can learn informative features for each task and domain by itself without requiring human effort. In this paper, we propose a generic neural architecture that can achieve this goal.

Both topic and EDU segmentation can be treated as a se-



(a) RNN-CRF (b) Sequence-to-sequence

Figure 2: Two classical models for sequence labeling.

quence labeling problem, where the task is to predict a sequence of 'yes/no' boundary tags at the level of sentences in a document (for topic segmentation) or words in a sentence (for EDU segmentation). Conditional random fields (CRFs) have been the traditional models for such sequence labeling tasks in NLP. More recently, recurrent neural networks with CRF output layer (RNN-CRF) as shown in Figure 2(a) have provided state-of-art results in many sequence tagging tasks in NLP [Lample *et al.*, 2016]. However, due to the sparsity of 'yes' boundary tags in EDU and topic segmentation tasks, CRFs did not provide any additional gain over simple classifiers like MaxEnt [Fisher and Roark, 2007; Joty *et al.*, 2015].

Instead, we cast our segmentation problems as sequence prediction tasks with seq2seq encoder-decoder models, which have been quite successful in machine translation and summarization. Figure 2(b) shows a toy seq2seq model, which uses an RNN to encode an input sequence $(U1, U_2, U_3)$, and then uses another RNN as a language model to generate the output sequence $(O1, O_2)$. However, one limitation of this basic model is that the output vocabulary (*i.e.,* from which $O1$ and $O2$ are drawn) is fixed so that it needs to train different models with respect to different vocabularies. Whereas in our tasks, the segmentation positions depend on the input sequence. For example, in the input sequence in Figure 3, there are three segment boundaries – units $U3$, $U6$, and $U8$.

To alleviate these issues, we propose SegBot, a generic end-to-end neural model for text segmentation at various levels of granularity. SegBot uses distributed representations to better capture lexical semantics, and employs a bidirectional RNN to model sequential dependencies while encoding a text. The decoder, which is an unidirectional RNN, uses a *pointer* mechanism [Vinyals *et al.*, 2015] to infer the segment boundaries. In addition, SegBot can effectively handle variable size vocabulary in the output to produce segment boundaries depending on the input sequence.

In summary, we make the following contributions:

- We propose SegBot– a generic end-to-end model for text segmentation at various levels of granularity. SegBot learns informative features automatically while alleviating the problem of tag sparsity in output sequence and the problem of variable size output vocabulary.
- We conduct experiments at two levels of granularity: document-level topic segmentation, and sentence-level EDU segmentation. Our results show that SegBot achieves new state-of-the-art results on both tasks.

2 Related Work

Text Segmentation. Existing methods for text segmentation fall into two categories: unsupervised and supervised. One branch of unsupervised methods is based on the idea of lexical cohesion, which states that similar vocabulary tends to be in a coherent segment. Hearst *et al.* [1997] introduced TextTiling, based on the fact that high vocabulary intersection between two adjacent blocks means high coherence and vice versa. C99 [Choi, 2000] is an algorithm based on divisive clustering with a matrix-ranking schema. LCSeg [Galley *et al.*, 2003] uses a lexical chain to identify and weight word repetitions. U00 [Utiyama and Isahara, 2001] is a probabilistic approach using dynamic programming to find a segmentation with minimum cost. Many unsupervised methods are based on topic modeling, including TopSeg [Brants *et al.*, 2002] and LDA based models [Misra *et al.*, 2009; Riedl and Biemann, 2012]. The idea is to induce semantic relationship between words, and use topics assigned by topic models to build sentence vector.

Supervised methods have also been proposed for text segmentation. Hsueh *et al.* [2006] integrate lexical and conversation-based features for topic and sub-topic segmentation. Hernault *et al.* [2010] use CRF to train a discourse segmenter with a set of lexical and syntactic features. Joty *et al.* [2015] train a binary classifier to decide whether to place an EDU boundary by using lexico-syntactic, shallow syntactic and contextual features. Different from existing studies where features have to be carefully hand-crafted, our approach does not need feature engineering.

Sequence Labeling. Sequence labeling is a fundamental task in NLP. Traditional methods employ machine learning models like hidden markov models [Qiao *et al.*, 2015] and CRFs [Liu *et al.*, 2011], and have achieved relatively high performance. The drawback of these methods is that they require domain-specific knowledge in the form of hand-crafted features and data pre-processing.

Recently, neural network models have been successfully applied to sequence labeling. For instance, Long Short-Term Memory (LSTM) network has been used to encode the input sequence, and a CRF layer is used to decode tag sequence [Lample *et al.*, 2016]. However, in text segmentation tasks, segment boundaries are sparse making CRFs less effective [Fisher and Roark, 2007; Joty *et al.*, 2015]. Vaswani *et al.* [2016] use seq2seq model to output tag sequence through a LSTM layer. The drawback of these approaches is that the output dictionary is fixed and is not dependent on the input sequence. By using a pointer-based neural network, our model overcomes both shortcomings simultaneously.

3 SegBot: Neural Text Segmentation Model

To address the problem of sparse boundary tags and variable output vocabularies in text segmentation, we propose a generic segmentation model SegBot that can perform segmentation at various levels of granularity, *e.g.,* document-level topic segmentation and sentence-level EDU segmentation. In the following, we describe our model in detail.

3.1 Model Architecture

Figure 3 shows the model architecture of SegBot, consisting of three components: encoding phase, decoding phase and pointing phase. Depending on the granularity of the task,

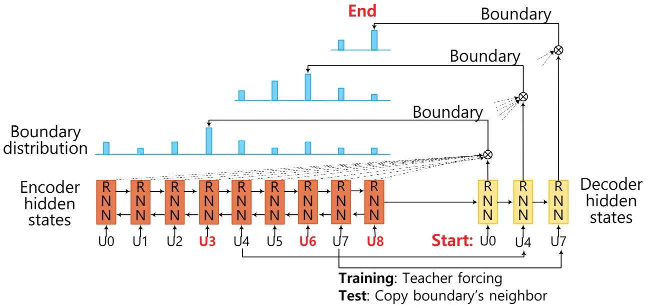


Figure 3: The model architecture of SEGBOT. Input sequence: $U0, U1, ..., U8$. Identified boundaries: $U3, U6, U8$.

the units in the input ($U0$ to $U8$) can be either sentences in a document (for topic segmentation) or words in a sentence (for EDU segmentation). We first represent each input unit with a distributed representation. For words, we use GloVe [Pennington *et al.*, 2014], which provides good representations that are validated on various NLP tasks including text classification and reading comprehension. For sentences, we use embeddings from [Arora *et al.*, 2017], which were shown to outperform many sophisticated supervised methods on various textual similarity tasks.

Formally, given an input sequence $\mathbf{U} = (U_1, U_2, \ldots, U_N)$ of length N, we get its distributed representations $\mathbf{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N)$ by looking up the corresponding embedding matrix, where $\boldsymbol{x}_n \in \mathbb{R}^K$ is the representation for the unit U_n with K being the dimensions. Our ultimate goal is to split the input sequence into contiguous segments by identifying the boundaries (*e.g.,* $U3$, $U6$ and $U8$ in Figure 3).

Encoding Phase. We encode the input sequence $\mathbf{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N)$ using a RNN. RNNs capture sequential dependencies, and with hidden cells like long short-term memory (LSTM) [Hochreiter and Schmidhuber, 1997] and gated recurrent unit (GRU) [Cho *et al.*, 2014], it can capture long distance dependencies without running into the problems of gradient vanishing or explosion. In our model, we use GRU to encode input sequences, which is similar to LSTM but is computationally cheaper. The GRU activations at time step n are computed as follows:

$$\boldsymbol{z}_n = \sigma(\boldsymbol{W}_z \boldsymbol{x}_n + \boldsymbol{R}_z \boldsymbol{h}_{n-1} + \boldsymbol{b}_z) \quad (1)$$

$$\boldsymbol{r}_n = \sigma(\boldsymbol{W}_r \boldsymbol{x}_n + \boldsymbol{R}_r \boldsymbol{h}_{n-1} + \boldsymbol{b}_r) \quad (2)$$

$$\boldsymbol{n}_n = \tanh(\boldsymbol{W}_h \boldsymbol{x}_n + \boldsymbol{R}_h(\boldsymbol{r}_n \odot \boldsymbol{h}_{n-1}) + \boldsymbol{b}_h) \quad (3)$$

$$\boldsymbol{h}_n = \boldsymbol{z}_n \odot \boldsymbol{h}_{n-1} + (1 - \boldsymbol{z}_n) \odot \boldsymbol{y}_n \quad (4)$$

where $\sigma()$ is the sigmoid function, $\tanh()$ is the hyperbolic tangent function, $\odot$ is the element-wise multiplication, $\boldsymbol{z}_n$ is update gate vector, $\boldsymbol{r}_n$ is reset gate vector, $\boldsymbol{n}_n$ is the new gate vector, and $\boldsymbol{h}_n$ is the hidden state at time step n. $\boldsymbol{W}$, $\boldsymbol{R}$, $\boldsymbol{b}$ are the parameters of the encoder that we need to learn.

We use a bi-directional GRU (BiGRU) network to memorize past and future information in the input sequence. Specifically, each hidden state of BiGRU is formalized as:

$$\boldsymbol{h}_n = \overrightarrow{\boldsymbol{h}}_n \oplus \overleftarrow{\boldsymbol{h}}_n \quad (5)$$

where $\oplus$ indicates concatenation operation, $\overrightarrow{\boldsymbol{h}}_n$ and $\overleftarrow{\boldsymbol{h}}_n$ are hidden states of forward (left-to-right) and backward (right-to-left) GRUs, respectively. Assuming the size of the GRU layer is H, the encoder yields hidden states in $\boldsymbol{h} \in \mathbb{R}^{N \times 2H}$.

Decoding Phase. Since the number of boundaries in the output vary with the input, it is natural to use RNN-based models to decode the output. At each step, the decoder takes a start unit (*i.e.,* start of a segment) U_m in the input sequence as input and transforms it to its distributed representation $\boldsymbol{x}_m$ by looking up the corresponding embedding matrix. It then passes $\boldsymbol{x}_m$ through a GRU-based (unidirectional) hidden layer. Formally, the decoder hidden state at a time step is computed by:

$$\boldsymbol{d}_m = GRU(\boldsymbol{x}_m, \boldsymbol{\theta}) \quad (6)$$

where $\boldsymbol{\theta}$ are the parameters in the hidden layer of the decoder, which has the same form as described by Equations (1) – (4). If the input sequence contains M boundaries, the decoder produces hidden states in $\boldsymbol{d} \in \mathbb{R}^{M \times H}$ with H being the dimensions of the hidden layer.

Pointing Phase. At each step, the output layer of our decoder computes a distribution over the possible positions in the input sequence for a possible segment boundary. For example, considering Figure 3, as the decoder starts with input $U0$, it computes an output distribution over all positions ($U0$ to $U8$) in the input sequence. Then, for $U4$ as input, it computes an output distribution over positions $U4$ to $U8$, and finally for $U7$ as input, it computes a distribution over $U7$ to $U8$. Note that unlike traditional seq2seq models (*e.g.,* the ones used in neural machine translation), where the output vocabulary is fixed, in our case the number of possible positions in the input sequence changes at each decoding step. To deal with this, we use a *pointing* mechanism [Vinyals *et al.*, 2015] in our decoder.

Recall that $\boldsymbol{h} \in \mathbb{R}^{N \times 2H}$ and $\boldsymbol{d} \in \mathbb{R}^{M \times H}$ are the hidden states in the encoder and the decoder, respectively. We use an attention mechanism to compute the distribution over the possible positions in the input sequence for decoding with input symbol U_m:

$$u_j^m = \boldsymbol{v}^T \tanh(\boldsymbol{W}_1 \boldsymbol{h}_j + \boldsymbol{W}_2 \boldsymbol{d}_m), \text{ for } j \in (m, \ldots, M) \quad (7)$$

$$p(y_m | \boldsymbol{x}_m) = \text{softmax}(\boldsymbol{u}^m) \quad (8)$$

where $j \in [m, M]$ indicates a possible position in the input sequence, and *softmax* normalizes u_j^m indicating the probability that the unit U_j is a boundary given the start unit U_m.

3.2 Model Training

We use "*teacher forcing*" [Lamb *et al.*, 2016] to train our model by supplying the ground-truth start units to the decoder RNN. This mechanism forces the RNNs to stay close to the ground-truth start units and segment boundaries. The loss function $\mathcal{L}$ is the negative log likelihood of boundary distribution over the whole training set $\mathcal{D}$, and can be written as:

$$\mathcal{L}(\boldsymbol{\omega}) = \sum_{\mathcal{D}} \sum_{m=1}^{M} - \log p(y_m | \boldsymbol{x}_m; \boldsymbol{\omega}) + \frac{\lambda}{2} ||\boldsymbol{\omega}||_2^2 \quad (9)$$

where $\boldsymbol{\omega}$ are the trainable parameters of the model (encoder and decoder), and λ is the strength of L_2 regularization.

When using the RNN decoder for prediction on test examples, the ground-truth boundaries are not available. Similar to traditional seq2seq decoders, we feed the input symbols based on the decoded symbol at the previous step, *e.g.,* we feed $U4$ after predicting a boundary at $U3$ in Figure 3.

4 Experiments

We conduct two sets of experiments to evaluate the effectiveness of SEGBOT: segmenting a document into topically coherent segments, and segmenting a sentence into EDUs.

4.1 Data and Metrics

Choi Dataset. For evaluating topic segmentation models, we use the commonly used Choi dataset [Choi, 2000]. It consists of 700 documents, each being a concatenation of 10 segments. A segment of a document is the first n (*s.t.* $3 \leq n \leq 11$, totally 4 subsets) sentences of a randomly selected document from the Brown corpus.

We use the error metric P_k [Beeferman *et al.*, 1999], which is the commonest metric to evaluate topic segmentation models. Using a sliding window of size k, P_k compares the inferred segmentation with gold-standard by:

$$P_k = \sum_{1 \leq s \leq t \leq T} \mathbf{1}(\delta_{tru}(s,t) \neq \delta_{hyp}(s,t)) \tag{10}$$

where a document consists of T sentences ($s, t = 1, 2, ..., T$), $\delta()$ is equal to 1 when sentences s and t belong to the same segment in the true/hypothetical segmentation and to 0 otherwise. $\mathbf{1}(a \neq b)$ is the indicator function (equal to 1 when $a = b$ and to 0 otherwise), k is equal to half of the document length divided by the number of gold segments. Note that lower P_k means higher accuracy.

RST-DT dataset. The Rhetorical Structure Theory Discourse Treebank (RST-DT) [Carlson *et al.*, 2002] is a publicly available corpus manually annotated with EDU segmentation and discourse relations according to Rhetorical Structure Theory. The RST-DT corpus is partitioned into a training set of 347 articles (6,132 sentences) and a test set of 38 articles (991 sentences), both from the Wall Street Journal.

Following previous work [Hernault *et al.*, 2010; Joty *et al.*, 2015], we measure segmentation accuracy with respect to the sentence-internal segmentation boundaries. That is, if a sentence has 3 EDUs, which correspond to 2 inside-sentence discourse boundaries and the end of the sentence. We measure the ability of our model to correctly identify these 2 boundaries within the sentence. Let g be the total number of sentence-internal boundaries in the human annotation, h be the total number of sentence-internal boundaries in the model output, and c be the total number of correct boundaries in the model output. Then, we measure Precision, Recall, and F-score for segmentation performance as follows:

$$\text{Precision} = \frac{c}{h}, \text{ Recall} = \frac{c}{g}, \text{ and F–score} = \frac{2c}{g+h} \tag{11}$$

4.2 Training Details

For Choi dataset, we split it into training and test sets with the same proportion as used in previous studies (see Section 4.3). For RST-DT dataset, training/test partition is provided. We use the first 10% data of shuffled training set as development set for both Choi dataset and RST-DT dataset.

Parameters	Choi	RST-DT
Learning rate	0.001	0.01
Regularization	$1e^{-4}$	$1e^{-4}$
Dropout	0.5	0.2
GRU dimensionality	128	64
GRU depth	3	6
Batch size	20	80

Table 1: Hyper-parameter settings.

Group	Method	P_k (%)
A	TextTiling [Hearst, 1997]	45.25
	C99 [Choi, 2000]	10.50
	U00 [Utiyama and Isahara, 2001]	7.75
	ADDP [Ji and Zha, 2003]	5.68
	TSM [Du *et al.*, 2013]	0.92
	GraphSeg [Glavaš *et al.*, 2016]	6.64
B	TopSeg [Brants *et al.*, 2002]	8.22
	F04 [Fragkou *et al.*, 2004]	4.20
	M09 [Misra *et al.*, 2009]	2.72
	SEGBOT (our model)	**0.33**
C	TopicTiling [Riedl and Biemann, 2012]	0.88
	BiLSTM-CRF [Lample *et al.*, 2016]	0.67
	SEGBOT (our model)	**0.11***

Table 2: Segmentation results on Choi dataset. Significant improvement over BiLSTM-CRF is marked with * (p-value < 0.01).

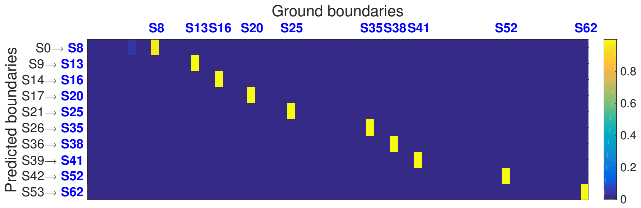
We use GloVe 300-dimensional pre-trained word embeddings released by Stanford[1], and the word vectors are fixed without fine-tuning during training. We use Adam optimizer to update model parameters. In addition, we use gradient clipping by a max norm of 5 and l_2 -regularization during training. Table 1 gives the details of other hyper-parameter settings. SEGBOT[2] is implemented with PyTorch framework and evaluated on NVIDIA Tesla P100 GPU.

4.3 Results

Topic Segmentation. In Table 2, we report the performance of SEGBOT and prominent methods on Choi dataset. Note that the methods in Group A involve no training set, but still require specifying some hyper-parameters. In Group B, the full dataset is split into 500 documents for training and 200 documents for test. In Group C, the full dataset is split into 630 for training and 70 for test. For fair comparison, the data partition of SEGBOT is consistent with these existing methods. Especially, we reimplement BiLSTM-CRF model, which is the state-of-the-art neural model for sequence labeling. Except BiLSTM-CRF, the P_k of baselines are from published results by averaging the 4 subsets.

[1] http://nlp.stanford.edu/projects/glove/

[2] An online version of SEGBOT (EDU segmentation) is available at http://138.197.118.157:8000/segbot/.



⇓ For the sake of space, we show three segments from S14∼ S25 for illustration: S14 ⟶ S16; S17 ⟶ S20; S21 ⟶ S25

S14 Some who have written on Utopia have treated it as “a learned diversion of a learned world”, “a phantasy with which More amused himself”, “a holiday work , a spontaneous overflow of intellectual high spirits , a revel of debate , paradox , comedy and invention”. S15 With respect to this view, two points are worth making. S16 First , it appears to be based on the fact that on its title page Utopia is described as “festivus”, “ gay”. S17 The Midwest , oxidation ponds are used extensively for the treatment of domestic sewage from suburban areas. S18 The high cost of land and a few operational problems resulting from excessive loadings have created the need for a wastewater treatment system with the operational characteristics of the oxidation pond but with the ability to treat more organic matter per unit volume. S19 Research at Fayette, Missouri on oxidation ponds has shown that the BOD in the treated effluent varied from 30 to 53 mg with loadings from 8 to 120 lb. S20 Since experience indicates that effluents from oxidation ponds do not create major problems at these BOD concentrations, the goal for the effluent quality of the accelerated treatment system was the same as from conventional oxidation ponds. S21 A proton magnetic resonance study of polycrystalline Afj as a function of magnetic field and temperature is presented. S22 Afj is paramagnetic, and electron paramagnetic dipole as well as nuclear dipole effects lead to line broadening. S23 The lines are asymmetric and over the range of field Afj gauss and temperature Afj the asymmetry increases with increasing Afj and decreasing T. S24 An isotropic resonance shift of Afj to lower applied fields indicates a weak isotropic hyperfine contact interaction. S25 The general theory of resonance shifts is used to derive a general expression for the second moment Afj of a polycrystalline paramagnetic sample and is specialized to Afj.

Figure 4: Visualization of topic segmentation. The boundaries of topics are in yellow. This document consists of 63 sentences. The upper part illustrates the boundary distribution and the lower part shows segmentation of $S14$-$S25$, which consisting of three topics: “*Utopia*” , “*Oxidation ponds*”, and “*Polycrystalline Afj*".

From the results, we make the following observations: (1) The performance of SegBot significantly outperforms all existing methods to date. (2) The performance of unsupervised methods (in Group A) is relatively poor as these methods do not utilize training data to learn accurate models. TSM achieves relatively high accuracy because it estimates parameters from the whole corpus, not only the test data. (3) SegBot outperforms all topic-modeling-based approaches (TSM, TopSeg, M09, and TopicTiling). Specifically, SegBot achieves an absolute P_k reduction of 87.5% over the state-of-the-art topic modeling method, *i.e.,* TopicTiling. One reason is that SegBot is based on the distributed representations of words which capture more semantic information over the topic modeling methods. (4) SegBot significantly outperforms the state-of-the-art neural model (BiLSTM-CRF) with an absolute P_k reduction of 83.6% (p-value < 0.01). This result shows that SegBot can effectively capture the dependencies of input sentence when the boundaries are sparse.

Figure 4 shows an example topic segmentation using SegBot. The upper part of this figure illustrates the boundary distribution. The x-axis shows gold boundaries while the y-axis shows predicted boundaries by SegBot. The heat map indicates that SegBot can effectively partition the document into ten topically coherent segments. We only show details of $S14$-$S25$ in the lower part of the figure. The predicted boundaries are highlighted in yellow. Observe that SegBot successfully identifies topic shifts (from “*Utopia*” to “*Oxidation ponds*”, to “*Polycrystalline Afj*”).

EDU Segmentation. We compare SegBot with six baselines. HILDA [Hernault *et al.*, 2010] and SPADE [Soricut and Marcu, 2003] are two publicly available discourse segmenters, which are based on syntactic and lexical features. F&R [Fisher and Roark, 2007] is a classification approach using features derived from finite-state and context-free annotations. DS [Joty *et al.*, 2015] is a binary Logistic Regression classifier. BiLSTM-CRF is a neural model for sequence labeling. We ran HILDA with its default settings. For SPADE, we applied the same modifications to its default settings as described in [Fisher and Roark, 2007], which delivers significant improvement over its original version. We reimplement DS and BiLSTM-CRF in our experiments. F&R [Fisher and Roark, 2007] segmenter is not available, so its performance is taken from the published results.

Method	Precision	Recall	F-score
HILDA [Hernault *et al.*, 2010]	77.9	70.6	74.1
SPADE [Soricut and Marcu, 2003]	83.8	86.8	85.2
F&R [Fisher and Roark, 2007]	91.3	89.7	90.5
DS [Joty *et al.*, 2015]	88.0	92.3	90.1
BiLSTM-CRF [Lample *et al.*, 2016]	89.1	87.8	88.5
SegBot (our model)	**91.6***	**92.8***	**92.2***

Table 3: Segmentation results on RST-DT Dataset. Significant improvements over BiLSTM-CRF is marked with * (p-value < 0.01).

Word Embeddings	Precision	Recall	F-score
GloVe vectors with fine-tuning	87.5	88.4	87.9
GloVe vectors without fine-tuning	91.6	92.8	92.2

Table 4: Performance w.r.t. fine-tuning and fixed word embeddings.

Table 3 reports precision, recall and F-score, of SegBot and six baseline systems. We make the following observations from the results. (1) SegBot outperforms all baselines on all measures. The improvements against baselines are from 0.3% to 17.6% on precision, 0.5%-31.4% on recall, and 1.8%-24.4% on F-scores, respectively. (2) It is worth mentioning that SegBot does not require any tedious feature engineering. Taking pre-trained word embeddings as input, SegBot outperforms all models that require carefully designed features including HILDA, SPADE, F&R and DS. Since SegBot does not need any syntactic parser or tagger, it can easily be transfered to other resource poor languages and domains. (3) BiLSTM-CRF takes the same input as our model, *i.e.,* pre-trained word embeddings. SegBot beats BiLSTM-CRF with an absolute F-score improvement of 4.2% (p-value < 0.01).

Figure 5 gives an example sentence segmentation by SegBot. It segments the sentence “*Sheraton and Pan Am **said** they are assured under the Soviet joint-venture **law** that they can repatriate profits from their hotel venture**.***” into three EDUs, with boundaries “**said**”, “**law**” and “**.**”, respectively. We observe that the identified boundaries have dominant attention weights, which implies that SegBot can successfully learn sentence structure and syntax.

4.4 Fixed Word Embeddings vs Fine-Tuning

Recall that SegBot takes the pre-trained GloVe vectors as input. During the training process, the word embeddings can also be fine-turned if we make them as learnable parameters. Accordingly, only those words appear in our training data will have embeddings. Table 4 reports the performance on RST-DT dataset, with and without fine-tuning the GloVe vectors. Observe that the performance of using off-the-shelf GloVe vectors, *i.e.,* without fine-tuning, is much better than the results of using fine-tuned embeddings.

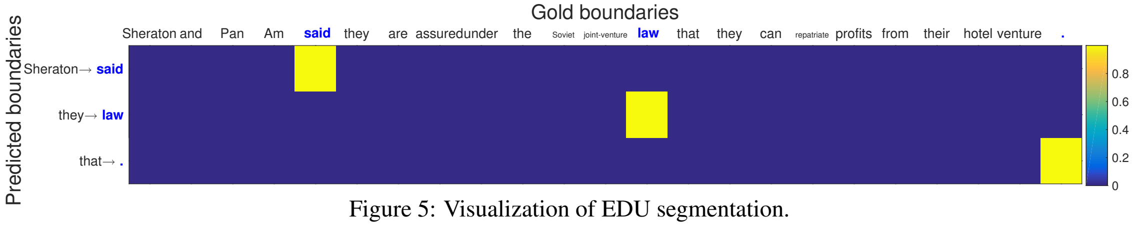


Figure 5: Visualization of EDU segmentation.

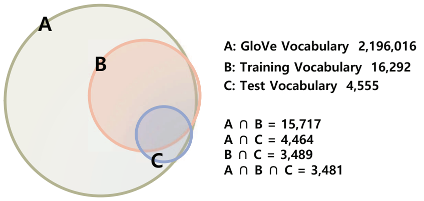


Figure 6: The venn diagram of three vocabularies.

The main reason is that fine-tuning results in more out-of-vocabularies on test data. Figure 6 illustrates the relationships among GloVe vocabulary, training data vocabulary, and test data vocabulary. It shows GloVe vocabulary covers 96.5% of training vocabulary and 98% of test vocabulary. With fine-tuning, only 76.6% of test vocabulary appear in training data ($B \cap C = 3,489$), resulting 1,066 (*i.e.,* $4,555 - 3,489$) words as "unknown". In short, GloVe vocabulary covers more words in test data than the fine-tuned embeddings. On the other hand, the higher performance of SEGBOT obtained on GloVe vocabulary shows that our model well handles those words that only appear in test data, *i.e.,* out-of-training-vocabulary words.

4.5 Effect of Pre-trained Word Embeddings

To test the impact of pre-trained word embeddings, we conducted experiments with two other sets of publicly available word embeddings, namely Google embeddings[3] trained on 100 billion words from Google News, and FastText embeddings[4] trained on 600 billion words from Common Crawl. We also include random initialization as a reference. Table 5 reports the performance of SEGBOT with the different word embeddings as input, on RST-DT dataset for EDU segmentation. Note that the word embeddings of Google, FastText and GloVe are fixed without fine-tuning. The embeddings with random initialization are learned during training.

There results show that Google embedding delivers the weakest performance. One possible reason is vocabulary mismatch. Google embedding excludes punctuation marks, digits, and stopwords, which are extremely important for EDU segmentation. FastText embedding obtains similar performance with GloVe embedding, as both were trained in case-sensitive manner, including common symbols such as punctuation marks, digits, and stopwords.

4.6 Error Analysis

In Figure 7, we show two error examples, one for topic segmentation and the other for EDU segmentation. When the decoder RNNs reach sentence $S27$, SEGBOT predicts boundary at $S32$ and misses the gold boundary $S29$. However, missing this gold boundary does not affect SEGBOT to correctly detect the subsequent boundaries ($S32$ and $S35$).

Word Embeddings	Precision	Recall	F-score
Random initialization (300d)	85.8	85.5	85.6
Google embeddings (300d)	84.5	84.6	84.5
FastText embeddings (300d)	91.1	93.0	92.0
GloVe embeddings (300d)	91.6	92.8	92.2

Table 5: Results of SEGBOT with different word embeddings on EDU segmentation.

For EDU segmentation shown in Figure 7, there are 3 gold boundaries in bold font: "*Until Mr. Luzon took the helm last November, Banco Exterior was run by* ***politicians*** *who lacked either the skills or the* ***will*** *to introduce innovative changes***.**". SEGBOT wrongly predicts "*skills*" as a boundary. Again, this wrongly predicted boundary does not affect the correct detection of "." to be the next boundary.

5 Conclusion

In this paper, we propose SEGBOT, an end-to-end neural model for text segmentation at different levels of granularity. SEGBOT does not need hand-crafted features or any prior knowledge of the given texts. It effectively addresses the sparsity of boundary tags in text segmentation tasks. More importantly, compared with existing neural models, SEGBOT has the key advantage of inherently handling variable size output vocabulary. Through two sets of experiments, on document-level topic segmentation and sentence-level EDU segmentation tasks respectively, we have demonstrated the effectiveness of SEGBOT against state-of-the-art solutions.

References

[Arora *et al.*, 2017] Sanjeev Arora, Yingyu Liang, and Tengyu Ma. A simple but tough-to-beat baseline for sentence embeddings. In *ICLR*, 2017.

[Beeferman *et al.*, 1999] Doug Beeferman, Adam Berger, and John Lafferty. Statistical models for text segmentation. *Machine learning*, 34(1):177–210, 1999.

[Brants *et al.*, 2002] Thorsten Brants, Francine Chen, and Ioannis Tsochantaridis. Topic-based document segmentation with probabilistic latent semantic analysis. In *CIKM*, pages 211–218, 2002.

[Carlson *et al.*, 2002] Lynn Carlson, Mary Ellen Okurowski, and Daniel Marcu. *RST discourse treebank*. Linguistic Data Consortium, University of Pennsylvania, 2002.

[Cho *et al.*, 2014] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural ma-

[3] https://code.google.com/archive/p/word2vec/

[4] https://fasttext.cc/docs/en/english-vectors.html

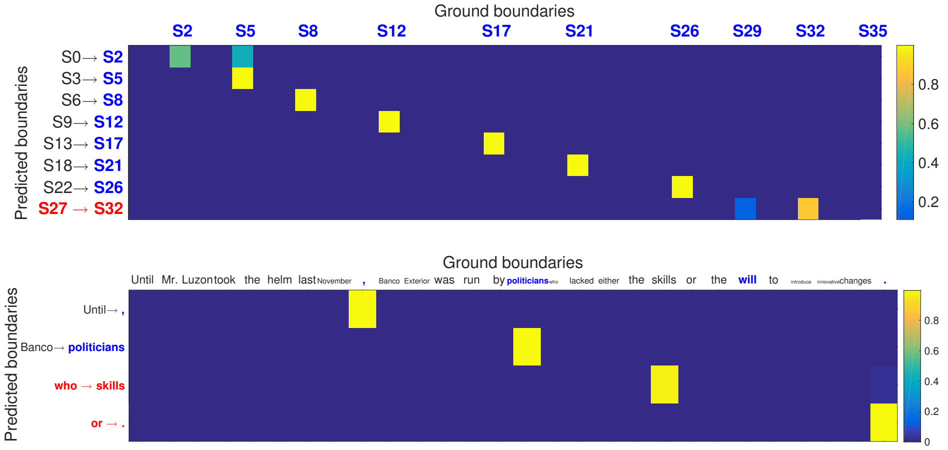


Figure 7: Two error examples. The gold boundaries are in blue, and the error boundaries are in red.

chine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[Choi, 2000] Freddy YY Choi. Advances in domain independent linear text segmentation. In *NAACL*, pages 26–33, 2000.

[Du *et al.*, 2013] Lan Du, Wray L Buntine, and Mark Johnson. Topic segmentation with a structured topic model. In *HLT-NAACL*, pages 190–200, 2013.

[Fisher and Roark, 2007] Seeger Fisher and Brian Roark. The utility of parse-derived features for automatic discourse segmentation. In *ACL*, volume 45, page 488, 2007.

[Fragkou *et al.*, 2004] Pavlina Fragkou, Vassilios Petridis, and Ath Kehagias. A dynamic programming algorithm for linear text segmentation. *JIIS*, 23(2):179–197, 2004.

[Galley *et al.*, 2003] Michel Galley, Kathleen McKeown, Eric Fosler-Lussier, and Hongyan Jing. Discourse segmentation of multi-party conversation. In *ACL*, pages 562–569, 2003.

[Glavaš *et al.*, 2016] Goran Glavaš, Federico Nanni, and Simone Paolo Ponzetto. Unsupervised text segmentation using semantic relatedness graphs. In *SEM@ACL*, 2016.

[Goldberg, 2017] Yoav Goldberg. *Neural Network Methods in Natural Language Processing*. Morgan & Claypool Publishers, 2017.

[Hearst, 1997] Marti A Hearst. Texttiling: Segmenting text into multi-paragraph subtopic passages. *Computational linguistics*, 23(1):33–64, 1997.

[Hernault *et al.*, 2010] Hugo Hernault, Danushka Bollegala, and Mitsuru Ishizuka. A sequential model for discourse segmentation. In *CICLing*, pages 315–326, 2010.

[Hochreiter and Schmidhuber, 1997] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[Hsueh *et al.*, 2006] Pei-Yun Hsueh, Johanna D Moore, and Steve Renals. Automatic segmentation of multiparty dialogue. In *EACL*, 2006.

[Ji and Zha, 2003] Xiang Ji and Hongyuan Zha. Domain-independent text segmentation using anisotropic diffusion and dynamic programming. In *SIGIR*, pages 322–329, 2003.

[Joty *et al.*, 2015] Shafiq Joty, Giuseppe Carenini, and Raymond T Ng. Codra: A novel discriminative framework for rhetorical analysis. *Computational Linguistics*, 41:385–435, 2015.

[Lamb *et al.*, 2016] Alex M Lamb, Anirudh GOYAL, Goyal, Ying Zhang, Saizheng Zhang, Aaron C Courville, and Yoshua Bengio. Professor forcing: A new algorithm for training recurrent networks. In *NIPS*, pages 4601–4609, 2016.

[Lample *et al.*, 2016] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*, 2016.

[Liu *et al.*, 2011] Xiaohua Liu, Shaodian Zhang, Furu Wei, and Ming Zhou. Recognizing named entities in tweets. In *ACL*, pages 359–367, 2011.

[Marcu, 2000] Daniel Marcu. *The Theory and Practice of Discourse Parsing and Summarization*. MIT Press, 2000.

[Misra *et al.*, 2009] Hemant Misra, François Yvon, Joemon M Jose, and Olivier Cappe. Text segmentation via topic modeling: an analytical study. In *CIKM*, pages 1553–1556, 2009.

[Pennington *et al.*, 2014] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543, 2014.

[Qiao *et al.*, 2015] Maoying Qiao, Wei Bian, Richard Xu, and Dacheng Tao. Diversified hidden markov models for sequential labeling. *TKDE*, 27(11):2947–2960, 2015.

[Riedl and Biemann, 2012] Martin Riedl and Chris Biemann. Topictiling: a text segmentation algorithm based on lda. In *ACL*, pages 37–42, 2012.

[Soricut and Marcu, 2003] Radu Soricut and Daniel Marcu. Sentence level discourse parsing using syntactic and lexical information. In *NAACL*, pages 149–156, 2003.

[Utiyama and Isahara, 2001] Masao Utiyama and Hitoshi Isahara. A statistical model for domain-independent text segmentation. In *ACL*, pages 499–506, 2001.

[Vaswani *et al.*, 2016] Ashish Vaswani, Yonatan Bisk, Kenji Sagae, and Ryan Musa. Supertagging with lstms. In *HLT-NAACL*, pages 232–237, 2016.

[Vinyals *et al.*, 2015] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In *NIPS*, pages 2692–2700, 2015.

[William and Thompson, 1988] Mann William and Sandra Thompson. Rhetorical structure theory: Toward a functional theory of text organization. *Text*, 8(3):243–281, 1988.