Neural Text Segmentation and Its Application to Sentiment Analysis

Jing Li, Billy Chiu, Shuo Shang and Ling Shao *Senior Member, IEEE*

Abstract—Text segmentation is a fundamental task in natural language processing. Depending on the levels of granularity, the task can be defined as segmenting a document into topical segments, or segmenting a sentence into elementary discourse units (EDUs). 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 cannot efficiently handle the issue of variable size output vocabulary. In light of such limitations, we propose a generic end-to-end segmentation model, namely SEGBOT, which first uses a bidirectional recurrent neural network to encode an input text sequence. SEGBOT then uses another recurrent neural networks, together with a pointer network, to select text boundaries in the input sequence. In this way, SEGBOT does not require any hand-crafted features. More importantly, SEGBOT 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 two tasks: document-level topic segmentation and sentence-level EDU segmentation. As a downstream application, we further propose a hierarchical attention model for sentence-level sentiment analysis based on the outcomes of SEGBOT. The hierarchical model can make full use of both word-level and EDU-level information simultaneously for sentence-level sentiment analysis. In particular, it can effectively exploit EDU-level information, such as the inner properties of EDUs, which cannot be fully encoded in word-level features. Experimental results show that our hierarchical model achieves new state-of-the-art results on the Movie Review and Stanford Sentiment Treebank benchmarks.

Index Terms—Natural Language Processing, Text Segmentation, Sentiment Analysis, Pointer Networks, Hierarchical Attention

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 coarse level, text segmentation generally refers to breaking a document into a sequence of topically coherent segments, often known as **topic segmentation** [1], [2]. Topic segmentation is typically considered as a pre-requisite for other higher level discourse analysis tasks (e.g., discourse parsing [3]), and has been shown to support a number of downstream NLP applications including text summarization [4] and passage retrieval [5], [6]. At a finer level, text segmentation refers to breaking each sentence into a sequence of elementary discourse units (EDUs), often known as **EDU segmentation** [7]. As exemplified in Figure 1, EDUs are clause-like units that serve as building blocks for discourse parsing in Rhetorical Structure Theory [8]. EDU segmentation is also useful for text compression [9].

Both topic and EDU segmentation have received a lot of attention in the past due to their utility in many NLP tasks. Although related, these two tasks are typically addressed separately with different sets of approaches; see [10] for an overview. Both supervised and unsupervised methods have been proposed for topic segmentation. Unsupervised segmentation models exploit the strong correlation between

[A rather average action film]$_{EDU1}$ [that benefits from several funny moments]$_{EDU2}$ [supplied by Epps.]$_{EDU3}$

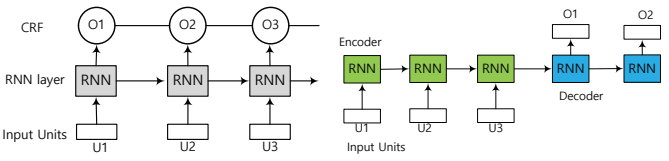
Fig. 1. A sentence with three elementary discourse units (EDUs).

topic and lexical usages, 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 the same segment are more similar to each other than to sentences in the preceding or the following segment. Examples of this category are TextTiling [1], C99 [2], and LCSeg [11]. 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 [12]–[14]. 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, they come with the price of requiring extensive efforts to manually design informative features and annotate large amounts of data. For EDU segmentation, HILDA [15], SPADE [16], F&R [17] and DS [3] are successful systems that use lexical and syntactic features in a supervised manner.

While most existing text 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 [18], [19]. 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 [10]. We envision a system that is based on distributed representation, and that can learn informative

- *J. Li and L. Shao are with the Inception Institute of Artificial Intelligence, Abu Dhabi, United Arab Emirates. E-mail: jingli.phd@hotmail.com; ling.shao@ieee.org.*
- *B. Chiu is with the Language Technology Laboratory, University of Cambridge, 9 West Road, Cambridge, CB39DB, UK. E-mail: hwc25@cam.ac.uk.*
- *S. Shang is with the University of Electronic Science and Technology of China, Chengdu, China. E-mail: jedi.shang@gmail.com.*

Manuscript received xx, xxxx; revised xx , xxxx.



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

Fig. 2. Two classical models for sequence labeling.

features for each task and domain by itself without requiring human effort. In this paper, we propose a neural architecture that can achieve this goal.

Both topic and EDU segmentation can be treated as sequence labeling problems, 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 classical models for such sequence labeling tasks in NLP [20]. More recently, recurrent neural networks with a CRF output layer (RNN-CRF), as shown in Figure 2(a), have provided state-of-art results in many sequence tagging tasks in NLP [21], [22]. However, due to the sparsity of 'yes' boundary tags in EDU and topic segmentation tasks, CRFs do not provide any additional gain over simple classifiers like MaxEnt [3], [17].

Instead, we cast our segmentation problems as sequence prediction tasks with encoder-decoder (known as seq2seq) models. Figure 2(b) shows a toy encoder-decoder model, which uses an RNN to encode an input sequence and then uses another RNN as a language model to generate/predict the output sequence. However, one limitation of this basic model is that the output vocabulary (i.e., from which $O1$ and $O2$ are drawn) is fixed, so different models must be retrained with respect to different vocabularies. Whereas in our tasks, the segmentation positions depend on the input sequence. For example, there are three segment boundaries – units $U3$, $U6$, and $U8$ in Figure 4. To alleviate these issues, we propose SEGBOT, a generic end-to-end neural model for text segmentation at various levels of granularity [23]. 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 [24], [25] 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.

Furthermore, we choose sentence-level sentiment analysis as a downstream application of SEGBOT. Sentiment analysis, also known as opinion mining, is the computational study of people's opinions, sentiments, emotions, appraisals, and attitudes from written languages [26]. Due to the increasing number of opinions and reviews on the internet, sentiment analysis has become a hot topic in knowledge discovery. As shown in Figure 3, we observe that 1) The overall sentiment of a sentence can be derived from different EDUs, each of which conveys a certain degree of sentiment polarity. For example, the overall sentiment is "negative", which is derived from EDU1, even though the EDU2 is "positive". 2) The sentiment of an EDU can be derived from different words, each of which has a different contribution.

[A rather average action film]$_{\text{EDU1: negative}}$ [that benefits from several funny moments]$_{\text{EDU2: positive}}$ [supplied by Epps.]$_{\text{EDU3: neutral}}$

Fig. 3. The sentiment polarity of EDUs.

For example, "funny" should be paid more attention to in EDU2. Based on these two intuitions, we propose a hierarchical attention model that takes advantage of both word-level and EDU-level information simultaneously for sentiment analysis. Different from existing sentiment analysis approaches [27], [28] which require expensive phrase-level annotations, our hierarchical attention model can exploit hierarchical structures to understand sentence sentiment in a lightweight manner (i.e., only requiring the output of SEGBOT). More importantly, our hierarchical model allows us to make full use of EDU-level information, such as the inner properties of EDUs, which cannot be fully encoded in word-level features.

In summary, we make the following contributions:

- We proposed 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 sequences and the problem of variable size output vocabulary.
- We conducted experiments to evaluate the effectiveness of SEGBOT at two levels of granularity: document-level topic segmentation, and sentence-level EDU segmentation. The results show that SEGBOT achieves new state-of-the-art results on both tasks.
- We implemented SEGBOT with a web application and provided users with Application Programming Interface (API): http://138.197.118.157:8000/segbot/. So far, 2300 users have visited our tool, from 30 countries.
- Based on the outcomes of SEGBOT, we proposed a hierarchical attention model to make full use of both word-level and EDU-level information simultaneously for sentence-level sentiment analysis.
- We conducted experiments to evaluate the effectiveness of our hierarchical attention model. Experimental results show that our hierarchical model achieves new state-of-the-art results on the Movie Review and Stanford Sentiment Treebank benchmarks.

2 RELATED WORK

2.1 Text Segmentation

The existing approaches for text segmentation fall into two categories: unsupervised methods and supervised methods. One branch of unsupervised methods is based on lexical cohesion, which states that similar vocabulary tends to be in a part of a coherent topic segment. Hearst et al. [1] introduced TextTiling, which is the most famous and earliest algorithm for text segmentation. TextTiling is based on the fact that high vocabulary intersection between two adjacent blocks is taken to mean high coherence and vice versa. C99 [2] is an algorithm based on divisive clustering with a matrix-ranking schema. LSeg [11] uses a lexical chain to identify and weight word repetitions. U00 [29] is a probalistic approach using dynamic programming to find a segment with minimum cost.

The other branch of unsupervised methods is based on topic modeling. The idea is to induce the semantic relationship between words and to use the frequency of a topic assigned to words by Latent Dirichlet Allocation to build a sentence vector. For example, TopSeg [30] was the first work to use the probabilistic latent semantic analysis to derive latent representations of segments. More recent models [31]–[33] employ the LDA to compute the latent topics and achieve superior performance to previous models.

Several approaches have investigated supervised learning in text segmentation. Fisher et al. [17] trained a classifier using a general machine learning approach and a range of finite-state and context-free derived features. Hernault et al. [15] used conditional random fields to train a discourse segmenter with a set of lexical and syntactic features. Joty et al. [3] trained a binary classifier to decide for each whether to place an EDU boundary using lexico-syntactic, shallow syntactic and contextual features. Different from these existing studies that use many hand-crafted features, our approach does not need manual feature engineering for text segmentation. To the best of our knowledge, our generic model is the first one using a neural architecture in text segmentation.

2.2 Sequence Labeling

Sequence labeling is a fundamental task in NLP, which includes part-of-speech tagging [34], chunking [35], named entity recognition (NER) [36] and so on. Classical methods employ machine learning models, like the hidden markov model [37] and conditional random fields (CRFs) [38], and have achieved relatively high performance. The drawback of these approaches is that they require large amounts of task-specific knowledge in the form of hand-crafted features and data pre-processing.

Recently, many neural network models have been successfully applied to sequence labeling. The use of neural models for NER was pioneered by [39], where an architecture based on temporal convolutional neural networks (CNNs) over word sequence, was proposed. Based on the recent taxonomy [40], a neural model for sequence labeling is composed of *distributed representations for input*, *a context encoder*, and *a tag decoder*. Some typical approaches for distributed representations for input include continuous bag-of-words (CBOW) and continuous skip-gram models [41]. In addition, there are two widely-used architectures for extracting character-level representation: CNN-based models [22], [42], [43] and RNN-based models [44]–[46].

A context encoder is used to capture the context dependencies for sequence labeling. Some studies [39], [47] have employed a CNN as the context encoder. For example, Strubell et al. [47] proposed Iterated Dilated Convolutional Neural Networks (ID-CNNs), which have better capacity than traditional CNNs for large context and structured prediction. The work by Huang et al. [48] is among the first to utilize a bidirectional Long Short-Term Memory (LSTM) CRF architecture for sequence tagging tasks (POS, chunking and NER). Following [48], a body of works [21], [22], [49]–[51] applied a bidirectional LSTM (BiLSTM) as the basic architecture to encode sequential context information.

A tag decoder takes context-dependent representations as input and produces a corresponding sequence of tags. For example, these work [21], [22] applied a LSTM network to encode the input sequence, and used a CRF layer to decode the tag sequence. Prior work along these lines is limited in text segmentation because Markov dependencies between tags cannot be effectively captured due to the sparsity of output sequence tags [3]. Some approaches [52], [53] apply a LSTM layer to produce the tag sequence. The drawback of these approaches is that the output dictionary is fixed and is not dependent on the input sequence. Compared with existing architectures, the main difference in our proposed SEGBOT is that our tag decoder is a pointer network, not CRF. Our model effectively captures sequential dependencies when boundary tags are sparse, while alleviating variable size vocabulary in the output to produce entity boundaries depending on the input sequence. In addition, Zhai et al. [54] proposed a model for sequence chunking based on pointer networks, which is most related to our work. Different from Zhai's model, our proposed model directly infers the start and end boundaries of an entity using a same network, which leads to a simpler architecture with fewer parameters.

2.3 Sentiment Analysis

Sentiment analysis has been growing to one of the most active research fields in NLP [26]. It has been widely used in various domains, including finance [55], marketing [56] and health [57]. While early approaches made use of handcrafted rule-based algorithms, modern ones most often resort to deep learning techniques. We organize them along two different axes: *traditional sentiment analysis* and *deep-learning based sentiment analysis*.

Traditional sentiment analysis has produced many techniques in different use cases, including both unsupervised and supervised methods. In the unsupervised setting, it is not necessary to acquire annotated training data. Instead, these approaches resort to sentiment lexicons, grammatical analysis, and syntactic patterns. For example, Kamps et al. [58] investigated a graph-thoretic model based on WordNet, and proposed measures that determine the semantic orientation of adjectives for three factors of subjective meaning. In the supervised setting, sentiment analysis can be framed as the problem of supervised classification. Classical supervised algorithms like Naive Bayes [59], [60], K-nearest neighbor [61], and Support Vector Machine [62], have been widely used in sentiment analysis. Despite their effectiveness, feature-based supervised approaches are labor intensive and require considerable amounts of engineering skills and domain expertise.

Deep learning has proven greatly successful in many natural language processing tasks [40], [63], [64]. It allows a machine to be fed raw data, which can be used to automatically discover latent representations and processing needed for classification or detection. Various neural models have been proposed to detect text sentiment at different levels of granularity: document level [65], sentence level [27], [66], and aspect level [67]. Here, we summarize existing studies of sentence-level sentiment analysis because our work belongs to this category. Socher et al. [27] first introduced a semi-supervise recursive autoencoder network for sentence level sentiment analysis. Dong et al. [28] proposed an adaptive recursive neural network for target-dependent Twitter

sentiment classification. Such recursive models learn vector space representations for multi-word phrases recursively, instead of through a bag-of-words model. However, annotating all subphrases can be expensive. Recently, CNN-based [68], [69] and RNN-based [66], [66], [70]–[72] models have become more popular in sentiment analysis, because they do not require parse trees to extract tree-structured features from sentences. For example, Qian et al. [66] proposed a linguistically regularized LSTM for sentiment classification, aiming to model the linguistic role of sentiment lexicons, negation words, and intensity words. Wang et al. [72] developed an RNN-Capsule network based on recurrent neural networks for sentence-level sentiment analysis. Tai et al. [73] introduced a generalization of the standard LSTM architecture to tree-structured network topologies. Zhang et al. [74] proposed a tree communication model using graph convolutional neural network and graph recurrent neural network, which allows rich information exchange between phrases constituent tree.

3 SegBot: Neural Text Segmentation

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

3.1 Model Architecture

Figure 4 illustrates the model architecture of SegBot, which consists of three components: a context encoder, a boundary decoder and a boundary pointer. It is worth emphasizing that SegBot is a generic model. Depending on the granularity of the task, the units in the input (i.e., $U0$ to $U8$) can be either sentences in a document (for topic segmentation) or words in a sentence (for EDU segmentation).

For input unit representation, distributed representation is an efficient method to capture a large number of precise syntactic and semantic unit relationships [75]. In SegBot, we first represent each input unit with a distributed representation. For words, we use GloVe [76], which provides good representations that are validated on various NLP tasks, including text classification and reading comprehension. For sentences, we use the embeddings from [77], 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 4).

3.2 Context Encoder

We encode the input sequence $\mathbf{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N)$ using an RNN. RNNs capture sequential dependencies, and with hidden cells like LSTMs [70] and gated recurrent units (GRUs) [78], it can capture long distance dependencies without running into the problems of gradient vanishing or explosion. In our model, we use a GRU to encode input sequences, which is similar to LSTM but is computationally cheaper.

Recall that $\boldsymbol{x}_n \in \mathbb{R}^K$ is the representation of U_n. The GRU activations at time step n are computed as follows:

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

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

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

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

where $\sigma()$ is the sigmoid function, $\tanh()$ is the hyperbolic tangent function, $\odot$ is an element-wise multiplication, $\boldsymbol{z}_n$ is the update gate vector, $\boldsymbol{r}_n$ is the 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. Each hidden state of the BiGRU is formalized as:

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

where $\oplus$ indicates a concatenation operation, and $\overrightarrow{\boldsymbol{h}}_n$ 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 $\boldsymbol{h} \in \mathbb{R}^{N \times 2H}$.

3.3 Boundary Decoder

Since the number of boundaries in the output vary with the input, it is natural to use RNN-based models to decode the output due to their ability to deal with variable lengths of sequences (note that *not variable output vocabulary*). At each step, the decoder takes a start unit (i.e., the 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) layer. Formally, the decoder's hidden state at a given 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 have 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.

3.4 Boundary Pointer

Unlike traditional seq2seq models, the output dictionary is fixed in each time step of decoder RNNs. In our case, it heavily depends on the input sequence. 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 4, 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

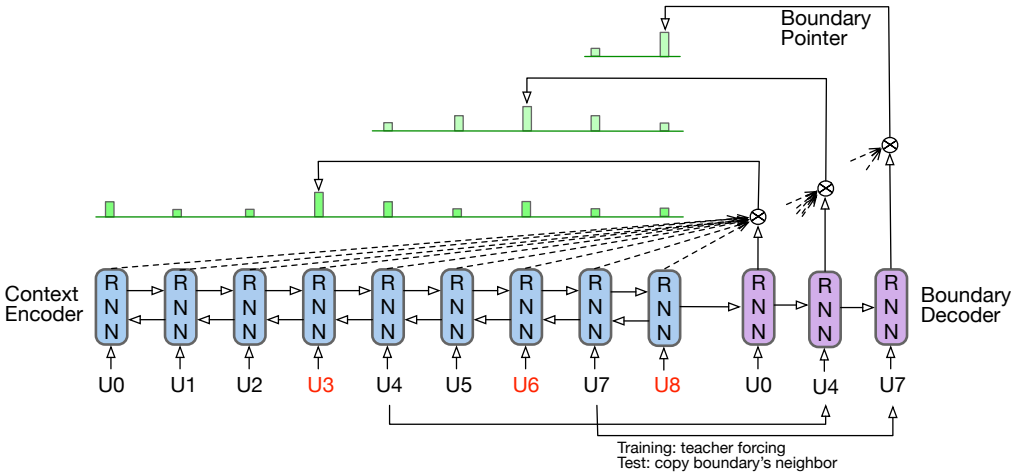


Fig. 4. The model architecture of SEGBOT. Input sequence: $\{U0, U1, ..., U8\}$. Identified boundaries (red color): $\{U3, U6, U8\}$. SEGBOT consists of three components: a context encoder, a boundary decoder and a boundary pointer. Given $U0$ as the decoder input, SEGBOT computes an output distribution over positions $U0$ to $U8$. Then with $U4$ as the decoder input, SEGBOT computes a distribution over $U4$ to $U8$, and $U6$ is identified as the boundary. Finally, with $U7$ as the decoder input, SEGBOT computes a distribution over $U7$ to $U8$.

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 [24] 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 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.

3.5 Model Training

We use "*teacher forcing*" [79] 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, decoder and pointer), and λ is the strength of L_2 regularization.

When using the RNN decoder for prediction on test samples, the ground-truth boundaries are not available. Similar to traditional seq2seq decoders in language models [18], we feed in the input units based on the decoded symbol at the previous step, e.g., we feed $U4$ after predicting a boundary at $U3$ in Figure 4. Algorithm 1 summarizes the learning process of SEGBOT.

Algorithm 1: Training the SEGBOT model.

Input: A set of sequences $\mathcal{D}$ with ground-truth segmentation labels $\{y\}$.
Output: SEGBOT model: $\boldsymbol{\omega}$

```
Initialize all parameters ω randomly;
foreach epoch in epoch_max do // iterate epoches
    Sample a mini-batch from D;
    Clear gradients dω ← 0;
    Compute context encoder states by Eq. (1) – (4);
    foreach m in M do // iterate decoder steps
        Compute current decoder state by Eq. (6);
        Compute current boundary distribution by Eq. (7) – (8);
        Retrieve input for next step m + 1 ; // train: teaching forcing, test: copy boundary's neighbor.
    Compute batch L based on Eq. (9);
    Update ω ← ω − (∂L/∂ω) · lr;
    return model ω
```

4 SEGBOT-BASED SENTIMENT ANALYSIS

In this paper, we choose text sentiment analysis as a downstream application to demonstrate the benefit of the EDUs produced by SEGBOT. In this Section, we introduce SEGBOT-based sentiment analysis in detail.

4.1 Model Architecture

Figure 5 illustrates the architecture of the proposed model for sentiment analysis. Our key idea is based on the three intuitions: 1) The overall sentiment of a sentence is derived from different EDUs, each of which conveys a certain degree of sentiment polarity. 2) The sentiment of an EDU is derived from different words, each of which has a different contribution. 3) We can employ both word-level and EDU-level information simultaneously for sentiment polarity score calculation. We resort to the attention mechanism [80] to capture the most important clues for sentiment analysis. As a result, our model can pay more attention to sentiment-carrying words and EDUs. One merit of our model is that

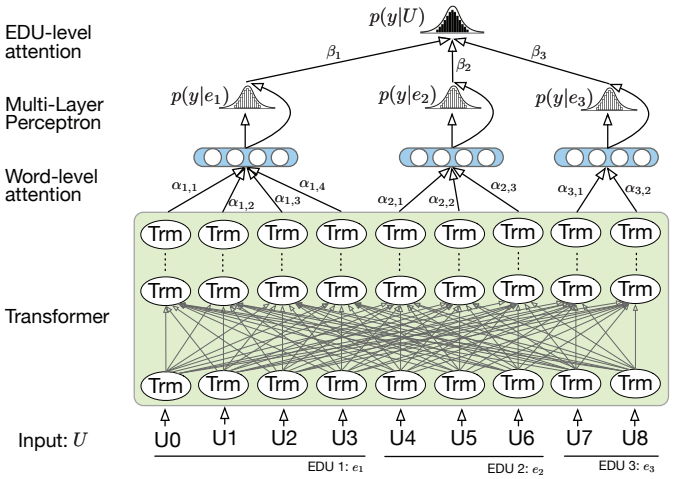


Fig. 5. The proposed hierarchical attention model for sentiment analysis. Input sequence: $\{U0, U1, ..., U8\}$, consisting of three EDUs $\{e_1, e_2, e_3\}$. The transformer layer maps the input representation to contextual embeddings for each word. The word-level attention layer captures the most important clues from words and forms EDU representations. The EDU-level attention layer captures the most indicative EDUs and forms the final sentiment polarity of $\boldsymbol{U}$.

it can take advantage of both word-level and EDU-level information simultaneously for sentiment analysis.

Our model is composed of four modules: the input representation, transformer encoder, word-level attention and EDU-level attention. Since it has two levels of attention, therefore, we call it a *hierarchical attention model*. The input representation module first represents the input sequence $\boldsymbol{U}$ as a sequence of embedding vectors. The transformer encoder module uses a multi-layer bidirectional Transformer [81] to capture the contextual information for each word. Subsequently, the word-level attention module is used to aggregate the representation of informative words to form an EDU vector. Finally, the EDU-level attention module uses an attention mechanism at the EDU level to capture the most indicative sentiment EDUs and form the final sentiment polarity of a sentence.

Input Representation. As shown in Figure 5, the input $\mathbf{U} = (U_1, U_2, \ldots, U_N)$ is a sequence of words of length N. Following [81], the input representation is constructed by summing the corresponding token and position embeddings. In particular, the token embeddings are from WordPiece embddings [82]. The position embeddings support a sequence length up to 512 tokens.

Transformer Encoder. The Transformer [83] is a new simple network based solely on an attention mechanism, dispensing with complex recurrence and convolutions entirely. It is superior in quality, while being more parallelizable and requiring significantly less time to train. We use a multi-layer bidirectional Transformer encoder to map the input representation to the contextual embedding for each word. Formally, let $\mathbf{P} = (\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_N)$ denote the sequence of word representations derived by the transformer encoder, where $\boldsymbol{p}_n \in \mathbb{R}^V$, V is the hidden size of the transformer encoder.

Word-level Attention. With SEGBOT, we had detected EDUs in the input sequence. Thus, $\mathbf{U} = (e_1, e_2, \ldots, e_L)$ is a sequence of EDUs of length L. For each EDU, we argue that not all words are equally important to the degree of sentiment polarity. We employ an attention mechanism to capture the most important clues for each EDU. Given an EDU e_l consisting of Q words, we compute word-level attention weights $(\alpha_{l,1}, ..., \alpha_{l,q}, ..., \alpha_{l,Q})$ and EDU embeddings $\boldsymbol{f}_l^j$ as follows:

$$\alpha_{l,q} = \frac{\exp(\boldsymbol{G}^T \cdot \boldsymbol{p}_{l,q})}{\sum_{i=1}^{Q} \exp(\boldsymbol{G}^T \cdot \boldsymbol{p}_{l,i})} \tag{10}$$

$$\boldsymbol{f}_l^j = \sum_{j=1}^{V} \alpha_{l,q} \cdot \boldsymbol{p}_{l,q}^j \tag{11}$$

where $\boldsymbol{G}$ are the learnable parameters of the word-level attention layer. $\boldsymbol{f}_l^j$ is the j-th slot element of the EDU embedding $\boldsymbol{f}_l$.

EDU-level Attention. After obtaining a representation $\boldsymbol{f}_l$ for every EDU, we can get individual EDU sentiment predictions via a softmax layer with a Multi-Layer Perceptron:

$$p(y|e_l) = \text{softmax}(\text{MLP}(\boldsymbol{f}_l)) \tag{12}$$

where the parameters of $\text{softmax}(\text{MLP}(\cdot))$ are shared across all EDUs. To form the final sentiment polarity of $\boldsymbol{U}$, a naive method is to average $p(y|e_l)$ across all EDUs. However, we claim that not all EDUs are equally important. Thus, we employ an EDU-level attention to capture the most indicative EDUs, which are more important to the final sentiment polarity. We compute EDU-level attention weights $(\beta_1, ..., \beta_l, ..., \beta_L)$ as follows:

$$\beta_l = \frac{\exp(\boldsymbol{D}^T \cdot \boldsymbol{f}_l)}{\sum_{i=1}^{L} \exp(\boldsymbol{D}^T \cdot \boldsymbol{f}_i)} \tag{13}$$

where $\boldsymbol{D}$ are the learnable parameters of the EDU-level attention layer. Finally, we obtain the sentence-level predictions over sentiment labels by

$$p(y|\boldsymbol{U}) = \sum_{l=1}^{L} \beta_l \cdot p(y|e_l) \tag{14}$$

4.2 Model Training

The hierarchical attention model is trained in an end-to-end way using sentence-level sentiment labels. The loss function $\mathcal{L}$ is the negative log likelihood of the boundary distribution over the whole training set $\mathcal{D}$, and can be written as:

$$\mathcal{L}(\boldsymbol{\xi}) = \sum_{\mathcal{D}} -\log p(y|\boldsymbol{U}; \boldsymbol{\xi}) + \frac{\lambda}{2}||\boldsymbol{\xi}||_2^2 \tag{15}$$

where $\boldsymbol{\xi}$ are the trainable parameters of the hierarchical attention model, and λ is the strength of the L_2 regularization. Algorithm 2 summarizes the learning process of the hierarchical attention model.

5 EXPERIMENT I: TEXT SEGMENTATION USING SEGBOT

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

Algorithm 2: Training the hierarchical attention model for sentiment analysis.

Input: A set of sequences $\mathcal{D}$ with ground-truth sentence-level sentiment labels $\{y\}$.
Output: The hierarchical attention model : $\boldsymbol{\xi}$

Initialize all parameters $\boldsymbol{\xi}$ randomly;
foreach *epoch in* $epoch_{max}$ **do** `// iterate epoches`
  Sample a mini-batch from $\mathcal{D}$;
  Clear gradients $d\boldsymbol{\xi} \leftarrow 0$;
  Compute input represntation;
  Compute the hidden states $\boldsymbol{P}$ of Transformer encoder;
  foreach l *in* L **do** `// iterate EDUs`
    Compute word-level attention weights by Eq. (10);
    Compute EDU representation $\boldsymbol{f}_l$ by Eq. (11);
    Compute $p(y|e_l)$ by Eq. (12);
    Compute EDU-leve attention weights by Eq. (13);
  Compute batch $\mathcal{L}$ based on Eq. (15);
  Update $\boldsymbol{\xi} \leftarrow \boldsymbol{\xi} - \frac{\partial \mathcal{L}}{\partial \boldsymbol{\xi}} \cdot lr$;
  return model $\boldsymbol{\xi}$

5.1 Experimental Settings

Choi Dataset. To evaluate topic segmentation models, we utilize the commonly used Choi dataset [2]. It consists of 700 documents, each being a concatenation of 10 segments. The corpus was generated by an automatic procedure. A segment of a document is the first n (*s.t.* $3 \leq n \leq 11$, 4 subsets in total) sentences of a randomly selected document from the Brown corpus.

We use the error metric P_k [84], which is the most common metric for evaluating topic segmentation models. Using a sliding window of size k, P_k compares the inferred segmentation with the gold-standard by:

$$P_k = \sum_{1 \leq s \leq t \leq T} \mathbf{1}(\delta_{tru}(s,t) \neq \delta_{hyp}(s,t)) \tag{16}$$

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

RST-DT Dataset. The Rhetorical Structure Theory Discourse Treebank (RST-DT) [85] 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 [3], [15], we measure EDU 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:

$$\text{Precision} = \frac{c}{h}, \text{ Recall} = \frac{c}{g}, \text{ and F–score} = \frac{2c}{g+h} \tag{17}$$

Baselines. For topic segmentation, we compare SEGBOT with 11 methods.

- **TextTiling** [1] is an unsupervised technique that makes use of patterns of lexical co-occurrence and distribution within texts.
- **C99** [2] is a method for linear text segmentation, which replaces inter-sentence similarity by rank in local context.
- **U00** [29] is a statistical model that finds the maximum-probability segmentation of a given text, for domain-independent text segmentation.
- **ADDP** [86] adapts a dynamic programming technique to find the optimal topical boundaries.
- **TSM** [33] integrates a point-wise boundary sampling algorithm into a structured topic model that can capture a simple hierarchical topic structure latent in texts.
- **GraphSeg** [87] exploits a measure of semantic relatedness of short texts to construct a semantic relatedness graph of the document.
- **TopSeg** [30] is based on probabilistic latent semantic analysis (PLSA) and exploits similarities in word meaning detected by PLSA.
- **F04** [88] uses a new segmentation cost function that incorporates two factors: a) within-segment word similarity and b) prior information about segment length.
- **M09** [31] uses a Latent Dirichlet Allocation (LDA) topic model to produce the topic distribution associated with each segment.
- **TopicTiling** [32] modifies TextTiling with topic IDs, obtained by an LDA model, instead of words.
- **BiLSTM-CRF** [21] is a state-of-the-art neural architecture for sequence labeling.

For EDU segmentation, we compare SEGBOT with 6 methods.

- **HILDA** [15] uses Conditional Random Fields to train a discourse segmenter on the RST Discourse Treebank, using a set of lexical and syntactic features.
- **SPADE** [16] is a discourse segmenter that accounts for both local interactions at the word level and for global interactions at more abstract levels, using lexical and syntactic features.
- **F&R** [17] uses different feature sets to perform EDU segmentation within a general machine learning framework, including wide range of finite-state and context-free derived features.
- **DS** [3] implements a binary classifier to decide, for each word (except the last) in a sentence, whether to place an EDU boundary after that token. It achieves state-of-the-art performance and reduces the time complexity by using fewer features.
- **BiLSTM-CRF** [21] is a state-of-the-art neural architecture for sequence labeling.

TABLE 1
Hyper-parameter settings.

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

Group	Method	P_k (%)
A	TextTiling [1]	45.25
	C99 [2]	10.50
	U00 [29]	7.75
	ADDP [86]	5.68
	TSM [33]	0.92
	GraphSeg [87]	6.64
B	TopSeg [30]	8.22
	F04 [88]	4.20
	M09 [31]	2.72
	SEGBOT (our model)	**0.33**
C	TopicTiling [32]	0.88
	BiLSTM-CRF [21]	0.67
	SEGBOT (our model)	**0.11***

Implementation Details. For the Choi dataset, we split it into training and test sets with the same proportions used in previous studies (see Section 5.2). For the RST-DT dataset, the training/test partition is provided. We use the first 10% of shuffled training set as the development set for both the Choi and RST-DT datasets.

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

5.2 Text Segmentation Results

Topic Segmentation. In Table 2, we report the performance of SEGBOT and prominent methods on the Choi dataset. Note that the methods in Group A involve no training set, but still require certain hyper-parameters to be specified. In Group B, the full dataset is split into 500 documents for training and 200 documents for testing. In Group C, the full dataset is split into 630 documents for training and 70 for testing. For fair comparison, the data partition of SEGBOT is consistent with these existing methods. We also reimplement BiLSTM-CRF, which is the state-of-the-art neural model for sequence labeling. Except for BiLSTM-CRF, the P_k values of the baselines are obtained from published results by averaging the 4 subsets (detailed in Section 5.1).

From the results, we make the following observations:

1) SEGBOT significantly outperforms all existing methods to date.

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

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

Method	Precision	Recall	F-score
HILDA [15]	77.9	70.6	74.1
SPADE [16]	83.8	86.8	85.2
F&R [17]	91.3	89.7	90.5
DS [3]	88.0	92.3	90.1
BiLSTM-CRF [21]	89.1	87.8	88.5
SEGBOT (our model)	**91.6***	**92.8***	**92.2***

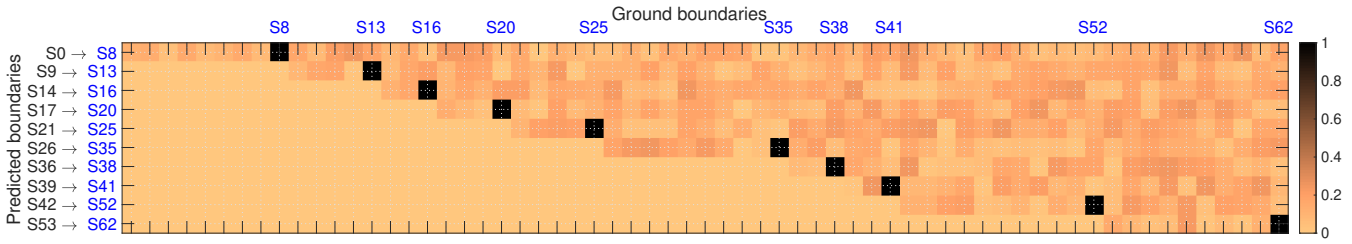
2) The performance of unsupervised methods (in Group A) is relatively poor as these methods do not utilize training data to learn accurate models. TSM achieves relatively high accuracy because it estimates parameters from the whole corpus, not only the test data.
3) SEGBOT outperforms all topic-modeling-based 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 than 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 sentences when the boundaries are sparse.

Figure 6 shows an example of 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 the boundaries predicted 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 purple. Notice that SEGBOT successfully identifies topic shifts (from "*Utopia*" to "*Oxidation ponds*", to "*Polycrystalline Afj*").

EDU Segmentation. We compare SEGBOT with the six baselines described in Section 5.1. Specifically, we run HILDA with its default settings. For SPADE, we apply the same modifications to its default settings as described in [17], which delivers a significant improvement over its original version. We reimplement DS and BiLSTM-CRF in our experiments. The F&R [17] segmenter is not publicly available, so its performance is taken from the published results.

Table 3 reports the Precision, Recall and F-score, of SEGBOT and the six baseline systems. We make the following observations from the results.

1) SEGBOT outperforms all baselines, in all measures. The improvements over baselines range from 0.3% to 17.6% in precision, 0.5%-31.4% in recall, and 1.8%-24.4% in 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



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

(514) 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". (515) With respect to this view, two points are worth making. (516) First, it appears to be based on the fact that on its title page Utopia is described as "festivus", "gay". (517) The Midwest , oxidation ponds are used extensively for the treatment of domestic sewage from suburban areas. (518) 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. (519) 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. (520) 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. (521) A proton magnetic resonance study of polycrystalline Afj as a function of magnetic field and temperature is presented. (522) Afj is paramagnetic, and electron paramagnetic dipole as well as nuclear dipole effects lead to line broadening. (523) The lines are asymmetric and over the range of field Afj gauss and temperature Afj the asymmetry increases with increasing Afj and decreasing T. (524) The lines are asymmetric and over the range of field Afj gauss and temperature Afj the asymmetry increases with increasing Afj and decreasing T. (525) 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.

Fig. 6. Visualization of topic segmentation. The boundaries of topics are in purple. This document consists of 63 sentences. The upper part illustrates the boundary distribution and the lower part shows segmentation of $S14$-$S25$, which consists of three topics: "*Utopia*" , "*Oxidation ponds*", and "*Polycrystalline Afj*".

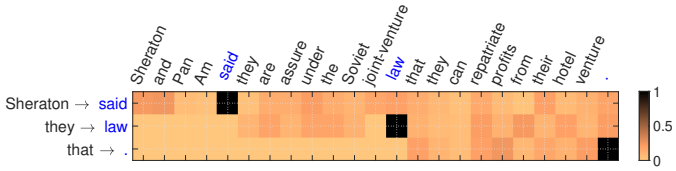


Fig. 7. Visualization of EDU segmentation. Given the decoder input "*Sheraton*", SEGBOT predicts its boundary at the position of "*siad*". For the input "*they*", SEGBOT predicts the corresponding boundary at the position of "*law*". Finally, given "*that*", "." is predicted as the boundary.

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 7 gives an example of EDU 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.

5.3 Model Analysis

Effect of 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 appearing in our training data will have embeddings. Table 4 reports the performance on the RST-DT dataset, with and without fine-tuning the GloVe vectors. Observe that the performance when using off-the-shelf GloVe vectors, i.e., without fine-tuning, is much better than when using fine-tuned embeddings.

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

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

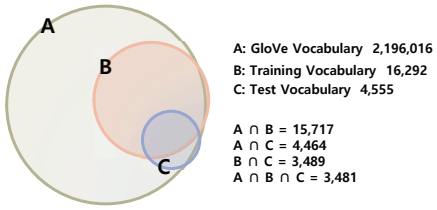


Fig. 8. The venn diagram of the three vocabularies.

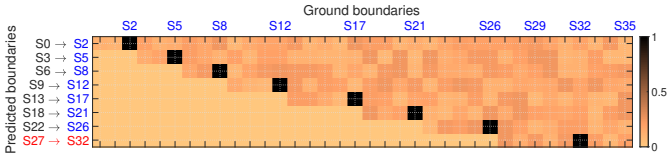
The main reason for this is that the fine-tuning mechanism results in more out-of-vocabularies in the test data. Figure 8 illustrates the relationships among GloVe vocabulary, training data vocabulary, and test data vocabulary. It shows that GloVe vocabulary covers 96.5% of training vocabulary and 98% of test vocabulary. With fine-tuning, only 76.6% of test vocabulary appear in the training data ($B \cap C = 3,489$), resulting in 1,066 (i.e., $4,555 - 3,489$) "unknown" words. In short, GloVe vocabulary covers more words in the test data than the fine-tuned embeddings. On the other hand, the higher performance of SEGBOT obtained on GloVe vocabulary shows that our model effectively handles those words that only appear in test data, i.e., out-of-training-vocabulary words.

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[2], trained on 100 billion words from Google News, and FastText embeddings[3], trained on 600 billion words from Common Crawl. We also

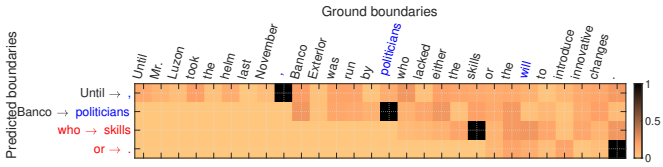
2. https://code.google.com/archive/p/word2vec/
3. https://fasttext.cc/docs/en/english-vectors.html

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

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



(a) A mistakenly detected case in Choi dataset.



(b) A mistakenly detected case in RST-DT dataset.

Fig. 9. Two error examples. The gold boundaries are in blue, and the error boundaries are in red.

include random initialization as a reference. Table 5 reports the performance of SEGBOT with the different word embeddings as input, on the 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.

The results show that Google embeddings deliver the weakest performance. One possible reason is vocabulary mismatch. Google embeddings exclude punctuation marks, digits, and stopwords, which are extremely important for EDU segmentation. FastText embeddings obtain a similar performance to GloVe embeddings, as both are trained in a case-sensitive manner, and include common symbols such as punctuation marks, digits, and stopwords.

Error Analysis. In Figure 9, we show two error examples, one for topic segmentation and the other for EDU segmentation. When the decoder RNNs reach sentence $S27$, SEGBOT predicts a boundary at $S32$ and misses the gold boundary $S29$. However, missing this gold boundary does not prevent SEGBOT from correctly detecting the subsequent boundaries ($S32$ and $S35$).

For EDU segmentation, shown in Figure 9, 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 prevent the correct detection of "." as the next boundary.

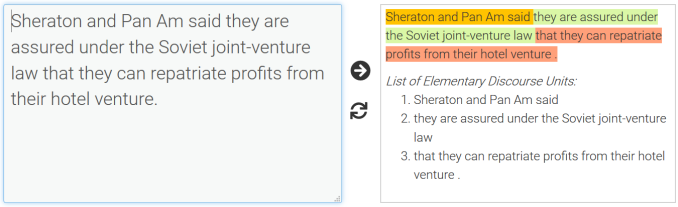


Fig. 10. The Web interface: http://138.197.118.157:8000/segbot/

5.4 Tool Support

We demonstrate the effectiveness of SEGBOT for discourse segmentation by developing a concise web interface[4], as shown in Figure 10. The interface consists of two panels: the input panel and the output panel.

The input panel takes sentences from users. Once a user clicks the button labeled with a right arrow, the input sentence is passed to the SEGBOT model, where each word in the sentence is an input unit (e.g., $U0$ to $U8$ in Figure 4).

The output panel displays the segmentation results in two forms. As shown in the output panel in Figure 10, on the top is the color-coded sentence, where each EDU is displayed in a different color. The color-coded sentence presents EDUs in their context to facilitate easy interpretation for the user. On the bottom of the output panel is the list of segmented EDUs, as shown in Figure 10. The order of the EDUs are determined by their positions in the original sentence. The list of EDUs allows users to consider individual EDUs and to pay more attention to their boundaries.

SEGBOT allows all valid English sentences as input. In our demonstration, we provide an example sentence for users to begin with. The three elementary discourse units are displayed in the output panel, as shown in Figure 10. Clicking the reset button results in clearing the input panel to take in the next sentence.

6 EXPERIMENT II: SEGBOT-BASED SENTIMENT ANALYSIS

6.1 Experimental Settings

We conduct experiments on two benchmark datasets: Movie Review and Stanford Sentiment Treebank.

Movie Review (MR) Dataset[5]. The MR dataset [89] consists of 10,662 movie-review "snippets" (a striking extract usually one sentence long) downloaded from www.rottentomatoes.com; each snippet was annotated with its source review's label (positive or negative), as provided by Rotten Tomatoes. There are 5,331 positive snippets and 5,331 negative snippets.

Stanford Sentiment Treebank (SST-5) Dataset[6]. The SST-5 dataset [90] is extracted from movie reviews, with human annotations of their sentiment. It contains 215,154 phrases with fine-grained sentiment labels in the parse trees of 11,855 sentences from movie reviews. This dataset was

4. An online version of SEGBOT (EDU segmentation) is available at http://138.197.118.157:8000/segbot/
5. http://www.cs.cornell.edu/people/pabo/movie-review-data/rt-polaritydata.tar.gz
6. https://nlp.stanford.edu/sentiment/index.html

created specifically to evaluate more complex compositional language models. There are five sentiment labels (very negative, negative, neutral, positive, and very positive). Following [72], we only utilize the sentence-level annotations and report results on sentence-level sentiment analysis.

Baselines. We compare our hierarchical attention model with 12 baseline methods.

- **RAE** [27] learns a distribution over sentiment labels at each node of the hierarchy, based on recursive autoencoders.
- **RNTN** [91] employs a syntactically untied recursive tensor neural network that learns syntactico-semantic, compositional vector representations.
- **LSTM** [70] is the classical long short-term memory.
- **Bi-LSTM** [71] represents bidirectional LSTM, which learns bidirectional long-term dependencies.
- **LR-LSTM** [66] is a linguistically-regularized variant of LSTM, regularizing the difference between the predicted sentiment distribution of the current position and that of the previous or next positions.
- **LR-Bi-LSTM** [66] is a linguistically-regularized variant of Bi-LSTM.
- **Tree-LSTM** [73] is a generalization of the standard LSTM architecture to tree-structured network topologies.
- **CNN** [68] uses a convolutional neural network on top of word vectors obtained from an unsupervised neural language model.
- **CNN-Tensor** [69] extends the n-gram convolution to non-consecutive words through a combination of low-rank tensors and pattern weighting.
- **DAN** [92] is a deep averaging network, which feeds an unweighted average of word vectors through multiple hidden layers before classification.
- **NCSL** [93] utilizes the strength of semantic feature learning of LSTM models to calculate a context-dependent weight for each word of a given an input sentence.
- **RNN-Capsule** [72] is a capsule model based on recurrent neural networks for sentiment analysis.

Implementation Details. For the MR dataset, we use the training/dev/test partition provided by the previous work [72]. For the SST-5 dataset, the standard data partition is provided. We measure the accuracy at sentence level on both datasets.

For the Transformer layer, we use the pre-trained BERT model[7]. That is, the number of layers (i.e., Transformer blocks) is 12; the number of self-attention heads is 12; and the hidden size is 768. This results in 110M parameters in the Transformer layer. As suggested in [81], the dropout probability is always kept at 0.1. The batch size is 16 and the learning rate is $3e-5$. We use the Adam optimizer to update model parameters. The hierarchical attention model is implemented using the PyTorch framework and evaluated on NVIDIA Tesla P100 GPUs.

TABLE 6
Sentiment analysis results on Movie Review (MR) and Stanford Sentiment Treebank (SST-5) datasets. Note that all models only use sentence-level annotations, rather than phrase-level annotations in the SST-5 dataset. Significant improvements over RNN-Capsule are marked with * (p-value < 0.01).

Method	MR dataset	SST-5 dataset
RAE [27]	77.7	43.2
RNTN [91]	75.9	43.4
LSTM [70]	77.4	45.6
Bi-LSTM [71]	79.3	46.5
LR-LSTM [66]	81.5	48.2
LR-Bi-LSTM [66]	82.1	48.6
Tree-LSTM [73]	80.7	48.1
CNN [68]	81.5	46.9
CNN-Tensor [69]	-	50.6
DAN [92]	-	47.7
NCSL [93]	82.9	47.1
RNN-Capsule [72]	83.8	49.3
Our model	**88.9***	**53.3***

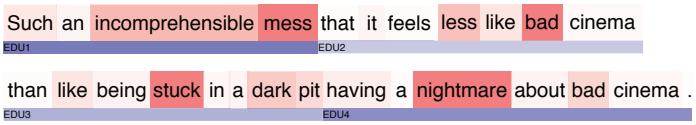
6.2 Sentiment Analysis Results

Table 6 reports the accuracy of different methods on the MR and SST-5 datasets. We make the following observations from the experimental results.

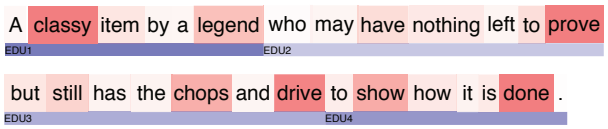
1) Our hierarchical attention model significantly outperforms all baseline methods on both the MR and SST-5 datasets. In particular, our model achieves relative accuracy improvements of 17.12% and 23.27% against the RAE method on MR and SST-5 datasets, respectively. Notably, our model significantly outperforms the second best method (i.e., RNN-Capsule on the MR dataset and CNN-Tensor on the SST-5 dataset) with relative improvements of 6.08% and 5.33%, respectively.
2) LR-Bi-LSTM and NCSL achieve better accuracy than other baseline methods on the MR dataset. However, these two methods require linguistic knowledge, such as intensity regularizer and sentiment lexicon. It is worth reiterating that our approach does not require any any tedious feature engineering except for the EDU segmentation results of SEGBOT.
3) CNN-Tensor outperforms other baseline methods on the SST-5 dataset. Unlike this method, our hierarchical attention model is based solely on an attention mechanism, dispensing with complex recurrences and convolutions entirely. Thus, it is superior in quality while being more parallelizable.
4) The recent work RNN-Capsule [72], based on Capsule networks [94], was the state-of-art method on the MR dataset until now. Our model significantly outperforms this method, achieving the new state-of-art performance.
5) Different from all baseline methods, our model leverages both word-level and EDU-level information simultaneously for sentiment polarity score calculation. We attribute our improvements to the fact that our hierarchical attention model can make full use of EDU-level information, such as the inner properties of EDUs, which cannot be fully encoded in word-level features.

As shown in Figure 5, our model is a hierarchical attention model consisting of word-level attention and EDU-level

7. https://github.com/huggingface/pytorch-pretrained-BERT



(a) A sentence (negative) from the MR test dataset.



(b) A sentence (very positive) from the SST-5 test dataset.

Fig. 11. Visualization of word-level (red color) and EDU-level (blue color) attention mechanisms. The depth of the color corresponds to the importance of words/EDUs for sentiment polarity calculation.

attention. For better understanding, we visualize the two-level attention weights with two examples in Figure 11. The first example is from the MR test dataset. SEGBOT detects four EDUs: *[Such an incomprehensible mess]*$_{EDU1}$ *[that it feels less like bad cinema]*$_{EDU2}$ *[than like being stuck in a dark pit]*$_{EDU3}$ *[having a nightmare about bad cinema]*$_{EDU4}$. From Figure 11(a), we observe that our hierarchical attention model captures more important sentiment-carrying words such as "mess", "bad" and "stuck" and "nightmare" in EDUs. Meanwhile, our model also pays more attention on the sentiment-carrying EDU, such as "Such an incomprehensible mess".

The second visualization example is shown in Figure 11(b). SEGBOT detects four EDUs: *[A classy item by a legend]*$_{EDU1}$ *[who may have nothing left prove]*$_{EDU2}$ *[but still has the chops and drive]*$_{EDU3}$ *[to show how it is done]*$_{EDU4}$. There is little information about sentiment polarity in EDU 2, 3, and 4. Our hierarchical attention model pays much more attention to EDU1 and successfully classifies this sentence as "very positive".

6.3 Ablation Study

We study the impacts of various architectural decisions on model performance. Table 7 reports an ablation analysis conducted on the test sets of MR and SST-5. The model variations are deviated from the standard model (i.e., the full architecture in Figure 5). The ablation study is based on 6 different variations, with the aims of clearly showcasing the importance of each component. In (1), we replace the Transformer with an RNN. In (2), we remove the word-level attention mechanism only. In (3), we remove the EDU-level attention mechanism only. In (4), we remove both the word-level and EDU-level attention mechanism, and detect sentence sentiment polarity using the representations of the first and last words. In (5), we detect sentence sentiment polarity using the average representation of the output of the Transformer layer. In (6), we detect sentence sentiment polarity using the average representation of the output of the word-level attention layer.

First, we observe that the full architecture is the best model configuration across the MR and SST-5 datasets. We attribute this to the fact that our model relies on the Transformer encoder and the hierarchical attention mechanism when modeling sentence sentiment polarity. *Second*, there is a sharp drop in performance when we replace the Transformer encoder with an RNN encoder. This observation ascertains the effectiveness of the Transformer encoder for capturing contextual dependency. *Third*, the impact of removing the hierarchical attention mechanism (see (2), (3) and (4)) is quite noticeable, leading to a huge degradation in performance on the two datasets. In particular, the model configuration (2) obtains a lower performance than (3), which implies that the word-level attention is more important in our architecture. *Finally*, the simple heuristics (i.e., averaging word/EDU representations) also lead to performance degradation, which signifies that modeling at word-level and EDU-level simultaneously is essential.

TABLE 7
Ablation analysis on the MR and SST-5 datasets.

Ablation	MR dataset	SST-5 dataset
(1) Replace Transformer with RNN	81.2	47.3
(2) w/o word-level attention	85.7	50.8
(3) w/o EDU-level attention	86.4	51.3
(4) w/o two-level attentions	84.9	50.5
(5) Average words	84.2	49.6
(6) Average EDUs	86.9	51.1
Full architecture	88.9	53.3

7 CONCLUSION

In this paper, we proposed SEGBOT, an end-to-end neural model for text segmentation at different levels of granularity. SEGBOT does not require hand-crafted features or any prior knowledge of the given texts. Our model effectively addresses the sparsity of boundary tags in text segmentation. More importantly, compared with existing neural models, SEGBOT has the key advantage of inherently handling variable size output vocabulary. To evaluate the effectiveness of SEGBOT, we conducted two sets of experiments: document-level topic segmentation and sentence-level EDU segmentation tasks, respectively. Experimental results show that SEGBOT significantly outperforms existing state-of-the-art solutions on both tasks.

EDU segmentation helps to preserve the semantic meaning of sentences, which subsequently benefits many downstream applications, e.g., sentiment analysis. As such, we proposed a hierarchical attention model to make full use of both word-level and EDU-level information simultaneously for sentence-level sentiment analysis. As a result, our hierarchical model pay more attention to sentiment-carrying words and EDUs. Experimental results show that our hierarchical model achieves new state-of-the-art results on the Movie Review and Stanford Sentiment Treebank benchmarks.

REFERENCES

[1] M. A. Hearst, "Texttiling: Segmenting text into multi-paragraph subtopic passages," *Comput. Linguist.*, vol. 23, no. 1, pp. 33–64, 1997.
[2] F. Y. Choi, "Advances in domain independent linear text segmentation," in *NAACL*, 2000, pp. 26–33.
[3] S. Joty, G. Carenini, and R. T. Ng, "Codra: A novel discriminative framework for rhetorical analysis," *Comput. Linguist.*, vol. 41, pp. 385–435, 2015.
[4] S. Harabagiu and F. Lacatusu, "Topic themes for multi-document summarization," in *SIGIR*, 2005, pp. 202–209.
[5] X. Huang, F. Peng, D. Schuurmans, N. Cercone, and S. E. Robertson, "Applying machine learning to text segmentation for information retrieval," *Inf. Retr.*, vol. 6, no. 3, pp. 333–362, 2003.

[6] G. Dias, E. Alves, and J. G. P. Lopes, "Topic segmentation algorithms for text summarization and passage retrieval: an exhaustive evaluation," in *AAAI*, 2007, pp. 1334–1339.

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

[8] M. William and S. Thompson, "Rhetorical structure theory: Toward a functional theory of text organization," *Text*, vol. 8, no. 3, pp. 243–281, 1988.

[9] C. Sporleder and M. Lapata, "Discourse chunking and its application to sentence compression," in *HLT-EMNLP*, 2005, pp. 257–264.

[10] M. Stede, *Discourse Processing*, ser. Synthesis Lectures on Human Language Technologies. Morgan And Claypool Publishers, 2011.

[11] M. Galley, K. McKeown, E. Fosler-Lussier, and H. Jing, "Discourse segmentation of multi-party conversation," in *ACL*, 2003, pp. 562–569.

[12] D. M. Blei and P. J. Moreno, "Topic segmentation with an aspect hidden markov model," in *SIGIR*, 2001, pp. 343–348.

[13] D. Blei, A. Ng, and M. Jordan, "Latent Dirichlet Allocation," *JMLR*, vol. 3, pp. 993–1022, 2003.

[14] J. Eisenstein and R. Barzilay, "Bayesian unsupervised topic segmentation," in *EMNLP*, Honolulu, Hawaii, 2008, pp. 334–343.

[15] H. Hernault, D. Bollegala, and M. Ishizuka, "A sequential model for discourse segmentation." in *CICLing*, 2010, pp. 315–326.

[16] R. Soricut and D. Marcu, "Sentence level discourse parsing using syntactic and lexical information," in *NAACL*, 2003, pp. 149–156.

[17] S. Fisher and B. Roark, "The utility of parse-derived features for automatic discourse segmentation," in *ACL*, vol. 45, no. 1, 2007, p. 488.

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

[19] J. Li, A. Sun, and Z. Xing, "Learning to answer programming questions with software documentation through social context embedding," *Inf. Sci.*, vol. 448-449, pp. 36–52, 2018.

[20] J. Lafferty, A. McCallum, and F. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *ICML*, San Francisco, CA, USA, 2001, pp. 282–289.

[21] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer, "Neural architectures for named entity recognition," *arXiv preprint arXiv:1603.01360*, 2016.

[22] X. Ma and E. Hovy, "End-to-end sequence labeling via bi-directional lstm-cnns-crf," in *ACL*, 2016, pp. 1064–1074.

[23] J. Li, A. Sun, and S. R. Joty, "Segbot: A generic neural text segmentation model with pointer network," in *IJCAI*, 2018, pp. 4166–4172.

[24] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer networks," in *NIPS*, 2015, pp. 2692–2700.

[25] J. Li, D. Ye, and S. Shang, "Adversarial transfer for named entity boundary detection with pointer networks," in *IJCAI*, 2019, pp. 5053–5059.

[26] L. Zhang, S. Wang, and B. Liu, "Deep learning for sentiment analysis: A survey," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 8, no. 4, p. e1253, 2018.

[27] R. Socher, J. Pennington, E. H. Huang, A. Y. Ng, and C. D. Manning, "Semi-supervised recursive autoencoders for predicting sentiment distributions," in *EMNLP*, 2011, pp. 151–161.

[28] L. Dong, F. Wei, C. Tan, D. Tang, M. Zhou, and K. Xu, "Adaptive recursive neural network for target-dependent twitter sentiment classification," in *ACL*, vol. 2, 2014, pp. 49–54.

[29] M. Utiyama and H. Isahara, "A statistical model for domain-independent text segmentation," in *ACL*, 2001, pp. 499–506.

[30] T. Brants, F. Chen, and I. Tsochantaridis, "Topic-based document segmentation with probabilistic latent semantic analysis," in *CIKM*, 2002, pp. 211–218.

[31] H. Misra, F. Yvon, J. M. Jose, and O. Cappe, "Text segmentation via topic modeling: an analytical study," in *CIKM*, 2009, pp. 1553–1556.

[32] M. Riedl and C. Biemann, "Topictiling: a text segmentation algorithm based on lda," in *ACL*, 2012, pp. 37–42.

[33] L. Du, W. L. Buntine, and M. Johnson, "Topic segmentation with a structured topic model." in *HLT-NAACL*, 2013, pp. 190–200.

[34] A. Voutilainen, "Part-of-speech tagging," *The Oxford handbook of computational linguistics*, pp. 219–232, 2003.

[35] L. A. Ramshaw and M. P. Marcus, "Text chunking using transformation-based learning," in *Natural language processing using very large corpora*. Springer, 1999, pp. 157–176.

[36] G. Zhou and J. Su, "Named entity recognition using an hmm-based chunk tagger," in *ACL*, 2002, pp. 473–480.

[37] M. Qiao, W. Bian, R. Xu, and D. Tao, "Diversified hidden markov models for sequential labeling," *TKDE*, vol. 27, no. 11, pp. 2947–2960, 2015.

[38] X. Liu, S. Zhang, F. Wei, and M. Zhou, "Recognizing named entities in tweets," in *ACL*, 2011, pp. 359–367.

[39] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *JMLR*, vol. 12, no. Aug, pp. 2493–2537, 2011.

[40] J. Li, A. Sun, J. Han, and C. Li, "A survey on deep learning for named entity recognition," *arXiv preprint arXiv:1812.09449*, 2018.

[41] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *ICLR*, 2013.

[42] P.-H. Li, R.-P. Dong, Y.-S. Wang, J.-C. Chou, and W.-Y. Ma, "Leveraging linguistic structures for named entity recognition with bidirectional recursive neural networks," in *EMNLP*, 2017, pp. 2664–2669.

[43] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *NAACL-HLT*, 2018, pp. 2227–2237.

[44] O. Kuru, O. A. Can, and D. Yuret, "Charner: Character-level named entity recognition," in *COLING*, 2016, pp. 911–921.

[45] Q. Tran, A. MacKinlay, and A. J. Yepes, "Named entity recognition with stack residual lstm and trainable bias decoding," in *IJCNLP*, 2017, pp. 566–575.

[46] M. Gridach, "Character-level neural network for biomedical named entity recognition," *Journal of biomedical informatics*, vol. 70, pp. 85–91, 2017.

[47] E. Strubell, P. Verga, D. Belanger, and A. McCallum, "Fast and accurate entity recognition with iterated dilated convolutions," in *ACL*, 2017, pp. 2670–2680.

[48] Z. Huang, W. Xu, and K. Yu, "Bidirectional lstm-crf models for sequence tagging," *arXiv preprint arXiv:1508.01991*, 2015.

[49] J. P. Chiu and E. Nichols, "Named entity recognition with bidirectional lstm-cnns," *TACL*, pp. 357–370, 2016.

[50] M. Rei, G. K. Crichton, and S. Pyysalo, "Attending to characters in neural sequence labeling models," in *COLING*, 2016, pp. 309–318.

[51] P. Zhou, S. Zheng, J. Xu, Z. Qi, H. Bao, and B. Xu, "Joint extraction of multiple relations and entities by using a hybrid neural network," in *Proc. Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data*. Springer, 2017, pp. 135–146.

[52] A. Vaswani, Y. Bisk, K. Sagae, and R. Musa, "Supertagging with lstms." in *HLT-NAACL*, 2016, pp. 232–237.

[53] Y. Shen, H. Yun, Z. C. Lipton, Y. Kronrod, and A. Anandkumar, "Deep active learning for named entity recognition," in *ICLR*, 2017.

[54] F. Zhai, S. Potdar, B. Xiang, and B. Zhou, "Neural models for sequence chunking." in *AAAI*, 2017, pp. 3365–3371.

[55] C. Kearney and S. Liu, "Textual sentiment in finance: A survey of methods and models," *International Review of Financial Analysis*, vol. 33, pp. 171–185, 2014.

[56] J. Smailović, M. Grčar, N. Lavrač, and M. Žnidaršič, "Predictive sentiment analysis of tweets: A stock market application," in *Human-computer interaction and knowledge discovery in complex, unstructured, Big Data*. Springer, 2013, pp. 77–88.

[57] L. Goeuriot, J.-C. Na, W. Y. Min Kyaing, C. Khoo, Y.-K. Chang, Y.-L. Theng, and J.-J. Kim, "Sentiment lexicons for health-related opinion mining," in *SIGHIT*, 2012, pp. 219–226.

[58] J. Kamps, M. Marx, R. J. Mokken, M. De Rijke *et al.*, "Using wordnet to measure semantic orientations of adjectives." in *LREC*, vol. 4. Citeseer, 2004, pp. 1115–1118.

[59] Z. Zhang, Q. Ye, Z. Zhang, and Y. Li, "Sentiment classification of internet restaurant reviews written in cantonese," *Expert Systems with Applications*, vol. 38, no. 6, pp. 7674–7682, 2011.

[60] V. Narayanan, I. Arora, and A. Bhatia, "Fast and accurate sentiment classification using an enhanced naive bayes model," in *IDEAL*, 2013, pp. 194–201.

[61] S. Tan and J. Zhang, "An empirical study of sentiment analysis for chinese documents," *Expert Systems with applications*, vol. 34, no. 4, pp. 2622–2629, 2008.

[62] K. Xu, S. S. Liao, J. Li, and Y. Song, "Mining comparative opinions from customer reviews for competitive intelligence," *Decision support systems*, vol. 50, no. 4, pp. 743–754, 2011.

[63] J. Li, Z. Xing, and A. Kabir, "Leveraging official content and social context to recommend software documentation," *IEEE TSC*, 2018.

[64] J. Li, Z. Xing, and A. Sun, "Linklive: discovering web learning resources for developers from q&a discussions," *World Wide Web*, vol. 22, no. 4, pp. 1699–1725, 2019.
[65] Z. Li, Y. Zhang, Y. Wei, Y. Wu, and Q. Yang, "End-to-end adversarial memory network for cross-domain sentiment classification." in *IJCAI*, 2017, pp. 2237–2243.
[66] Q. Qian, M. Huang, J. Lei, and X. Zhu, "Linguistically regularized lstms for sentiment classification," in *ACL*, 2016, pp. 1679–1689.
[67] S. Ruder, P. Ghaffari, and J. G. Breslin, "A hierarchical model of reviews for aspect-based sentiment analysis," in *EMNLP*, 2016, pp. 999–1005.
[68] Y. Kim, "Convolutional neural networks for sentence classification," in *EMNLP*, 2014, pp. 1746–1751.
[69] T. Lei, R. Barzilay, and T. Jaakkola, "Molding cnns for text: nonlinear, non-consecutive convolutions," in *ACL*, 2015, pp. 1565–1575.
[70] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
[71] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional lstm and other neural network architectures," *Neural Networks*, vol. 18, no. 5-6, pp. 602–610, 2005.
[72] Y. Wang, A. Sun, J. Han, Y. Liu, and X. Zhu, "Sentiment analysis by capsules," in *WWW*, 2018, pp. 1165–1174.
[73] K. S. Tai, R. Socher, and C. D. Manning, "Improved semantic representations from tree-structured long short-term memory networks," in *ACL*, 2015, pp. 1556–1566.
[74] Y. Zhang and Y. Zhang, "Tree communication models for sentiment analysis," in *ACL*, 2019, pp. 3518–3527.
[75] T. Mikolov, W.-t. Yih, and G. Zweig, "Linguistic regularities in continuous space word representations," in *NAACL-HLT*, 2013, pp. 746–751.
[76] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *EMNLP*, 2014, pp. 1532–1543.
[77] S. Arora, Y. Liang, and T. Ma, "A simple but tough-to-beat baseline for sentence embeddings," in *ICLR*, 2017.
[78] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," *arXiv preprint arXiv:1409.1259*, 2014.
[79] A. M. Lamb, A. GOYAL, Goyal, Y. Zhang, S. Zhang, A. C. Courville, and Y. Bengio, "Professor forcing: A new algorithm for training recurrent networks," in *NIPS*, 2016, pp. 4601–4609.
[80] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," *arXiv preprint arXiv:1508.04025*, 2015.
[81] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.
[82] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey *et al.*, "Google's neural machine translation system: Bridging the gap between human and machine translation," *arXiv preprint arXiv:1609.08144*, 2016.
[83] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *NIPS*, 2017, pp. 5998–6008.
[84] D. Beeferman, A. Berger, and J. Lafferty, "Statistical models for text segmentation," *Machine learning*, vol. 34, no. 1, pp. 177–210, 1999.
[85] L. Carlson, M. E. Okurowski, and D. Marcu, *RST discourse treebank*. Linguistic Data Consortium, University of Pennsylvania, 2002.
[86] X. Ji and H. Zha, "Domain-independent text segmentation using anisotropic diffusion and dynamic programming," in *SIGIR*, 2003, pp. 322–329.
[87] G. Glavaš, F. Nanni, and S. P. Ponzetto, "Unsupervised text segmentation using semantic relatedness graphs," in *SEM@ACL*, 2016.
[88] P. Fragkou, V. Petridis, and A. Kehagias, "A dynamic programming algorithm for linear text segmentation," *JIIS*, vol. 23, no. 2, pp. 179–197, 2004.
[89] B. Pang and L. Lee, "Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales," in *ACL*, 2005, pp. 115–124.
[90] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *EMNLP*, 2013, pp. 1631–1642.
[91] R. Socher, J. Bauer, C. D. Manning *et al.*, "Parsing with compositional vector grammars," in *ACL*, vol. 1, 2013, pp. 455–465.
[92] M. Iyyer, V. Manjunatha, J. Boyd-Graber, and H. Daumé III, "Deep unordered composition rivals syntactic methods for text classification," in *ACL*, vol. 1, 2015, pp. 1681–1691.
[93] Z. Teng, D. T. Vo, and Y. Zhang, "Context-sensitive lexicon features for neural sentiment analysis," in *EMNLP*, 2016, pp. 1629–1638.
[94] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," in *NIPS*, 2017, pp. 3856–3866.

```
@article{li22segsenti,
author    = {Jing Li and Billy Chiu and Shuo Shang and Ling Shao},
title     = {Neural Text Segmentation and Its Application to Sentiment Analysis},
journal   = {IEEE Transactions on Knowledge and Data Engineering (TKDE)},
volume    = {34},
number    = {2},
pages     = {828--842},
year      = {2022},
url       = {https://doi.org/10.1109/TKDE.2020.2983360},
doi       = {10.1109/TKDE.2020.2983360},
}
```