Neural Discourse Segmentation

Jing Li
Inception Institute of Artificial Intelligence, Abu Dhabi, United Arab Emirates
jingli.phd@hotmail.com

Abstract
Identifying discourse structures and coherence relations in a piece of text is a fundamental task in natural language processing. The first step of this process is segmenting sentences into clause-like units called elementary discourse units (EDUs). Traditional solutions to discourse segmentation heavily rely on carefully designed features. In this demonstration, we present SEGBOT, a system to split a given piece of text into sequence of EDUs by using an end-to-end neural segmentation model. Our model does not require hand-crafted features or external knowledge except word embeddings, yet it outperforms state-of-the-art solutions to discourse segmentation.

1 Introduction
Identifying discourse structures and coherence relations plays an important role in various natural language processing (NLP) applications such as text summarization [Durrett et al., 2016; Li et al., 2018d], information extraction [Li et al., 2018a], question answering [Li et al., 2018c] and passage retrieval [Dias et al., 2007; Li et al., 2018e]. The first necessary step of this process is to perform discourse segmentation, which refers to the task of breaking a piece of text into a sequence of elementary discourse units or EDUs [Marcu, 2000]. As exemplified in Figure 1, EDUs are clause-like units that serve as building blocks for discourse parsing in Rhetorical Structure Theory or RST [Mann and Thompson, 1988]. If the given text is wrongly segmented in this stage, the error propagates to the subsequent steps and it becomes unreliable to assign correct discourse relations. Thus, discourse segmentation is a crucial step for effective discourse parsing [Joty et al., 2015]. It is also important for downstream applications like text compression [Sporleder and Lapata, 2005] and machine translation [Joty et al., 2017].

Discourse segmentation can be treated as a sequence labeling problem, where the task is to predict a sequence of ‘yes/no’ boundary tags. Traditional approaches to sequence labeling use conditional random fields (CRFs) with hand-crafted features. Recent methods use recurrent neural networks (RNNs) with possibly a CRF layer at the output [Lample et al., 2016; Xu et al., 2019]. However, when it comes to EDU segmentation, studies have shown that sequence labeling does not provide any additional gain over simple binary classification (e.g., using MaxEnt) due to the sparsity of ‘yes’ boundary tags [Fisher and Roark, 2007; Joty et al., 2015]. Therefore, in our work, we frame the task as a sequence generation task with a seq2seq model [Sutskever et al., 2014]. Our goal is to generate a sequence of indices in the input text each indicating an EDU boundary; see Figure 2(a). Notice that the set of candidate positions changes at each decoding step. This is unlike the existing seq2seq model, where the output vocabulary of the decoder RNN is fixed at each decoding step.

To alleviate these issues, we propose SEGBOT, an end-to-end neural system for discourse segmentation. SEGBOT uses distributed representations to better capture lexical semantics, and employs a bidirectional RNN to model sequential dependencies while encoding the given text. The decoder, which is a unidirectional RNN, uses a pointer mechanism [Vinyals et al., 2015; Li et al., 2018b] to infer the segment boundaries. SEGBOT effectively handles variable size vocabulary in the output, to produce segment boundaries depending on the input sequence. Experiments show that SEGBOT outperforms state-of-the-art results on discourse segmentation.

Note that SEGBOT is a generic neural segmenter, and it can also be applied to topic segmentation (i.e., breaking a document into a sequence of topically coherent segments). In this demonstration, we focus on EDU segmentation.

2 The SEGBOT Model
Encoding Phase. For the task of discourse segmentation, the units in the input (U0 to U8 in Figure 2(a)) are words in a sentence. Each word is represented with a distributed representation. We use the pretrained word vectors from GloVe [Pennington et al., 2014], which are validated on various NLP tasks including text classification [Iyyer et al., 2015] and reading comprehension [Wang et al., 2017].

---

1 The online demonstration of SEGBOT can be accessed at http://138.197.118.157:8000/segbot/
Formally, given an input sequence \( U = (U_1, U_2, \ldots, U_N) \) of length \( N \), we get its distributed representations \( X = (x_1, x_2, \ldots, x_N) \) by looking up the corresponding embedding matrix, where \( x_n \in \mathbb{R}^K \) 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., \( U_3, U_5 \) and \( U_10 \) in Figure 2(a)).

**Decoding Phase.** The decoder of \( \text{SEG} \text{BOT} \) takes a start unit \( i.e., \text{start of a segment} U_m \) in the input sequence as input and transforms it to its distributed representation \( x_m \) by looking up the corresponding embedding matrix. It then passes \( x_m \) through a GRU-based (unidirectional) hidden layer. We use “teacher forcing” [Lamb et al., 2016] to train our model by supplying the ground-truth start units to decoder RNNs.

**Pointing Phase.** 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 2(a), as the decoder starts with input \( U1 \), it computes an output distribution over all positions (\( U1 \) to \( U10 \)) in the input sequence. Then, for \( U4 \) as input, it computes an output distribution over positions \( U4 \) to \( U10 \), and finally for \( U6 \) as input, it computes a distribution over \( U6 \) to \( U10 \).

### Interface and Demonstration

We demonstrate the effectiveness of \( \text{SEG} \text{BOT} \) for discourse segmentation by developing a concise web interface, as shown in Figure 2(b). 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 \( \text{SEG} \text{BOT} \) model, where each word in the sentence is an input unit (e.g., \( U0 \) to \( U8 \) in Figure 2(a)).

The output panel displays the segmentation results in two forms. As shown in the output panel in Figure 2(b), on the top is the color-coded sentence where every EDU is displayed in a different color. The color-coded sentence presents EDUs in its 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 2(b). The order of the EDUs are determined by their positions in the original sentence. The list of EDUs allows user to consider individual EDUs and to pay more attention to their boundaries.

\( \text{SEG} \text{BOT} \) allows all valid English sentences as input. In our demonstration, we provide an example sentence “Sheraton and Pan Am said they are assured under the Soviet joint-venture law that they can repatriate profits from their hotel venture” for users to begin with. The three elementary discourse units are displayed in the output panel as shown in Figure 2(b). Clicking the reset button results in clearing the input panel to take in the next sentence.

### Effectiveness Evaluation

The RST Discourse Treebank (RST-DT) [Carlson et al., 2002] is a publicly available corpus manually annotated with EDUs and relations according to RST. We use GloVe 300-dimensional pre-trained word embeddings released by Stanford, and the word vectors are kept fixed during training. Following previous work [Hernault et al., 2010; Joty et al., 2015], we measure segmentation performance using Precision, Recall, and F-score, reported in Table 1.

Comparing against six baselines, \( \text{SEG} \text{BOT} \) consistently outperforms all baselines on all measures. The improvement against baselines is from 0.3% to 17.6% on precision, and 0.5%-31.4% on recall, 1.8%-24.4% on F-score. It is worth mentioning that \( \text{SEG} \text{BOT} \) does not require any feature engineering. Simply taking pre-trained word embeddings as input, \( \text{SEG} \text{BOT} \) outperforms all models with carefully designed features, such as HILDA, SPADE, F&R and DS. The state-of-the-art neural model (BiLSTM-CRF) takes the same input as our model, \( i.e., \) pre-trained word embeddings. \( \text{SEG} \text{BOT} \) beats BiLSTM-CRF with an absolute F-score improvement of 4.2% (\( p\)-value < 0.01).

**Table 1:** Segmentation results on RST-DT Dataset. Significant improvements over the underlined methods are marked with * (\( t\)-test, \( p\)-value < 0.01).

<table>
<thead>
<tr>
<th>Method</th>
<th>Precision</th>
<th>Recall</th>
<th>F-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>HILDA [Hernault et al., 2010]</td>
<td>77.9</td>
<td>70.6</td>
<td>74.1</td>
</tr>
<tr>
<td>SPADE [Soricut and Marcu, 2003]</td>
<td>83.8</td>
<td>86.8</td>
<td>85.2</td>
</tr>
<tr>
<td>F&amp;R [Fisher and Roark, 2007]</td>
<td>91.3</td>
<td>89.7</td>
<td>90.5</td>
</tr>
<tr>
<td>DS [Joty et al., 2015]</td>
<td>88.0</td>
<td>92.3</td>
<td>90.1</td>
</tr>
<tr>
<td>BiLSTM-CRF [Lample et al., 2016]</td>
<td>89.1</td>
<td>87.8</td>
<td>88.5</td>
</tr>
<tr>
<td>( \text{SEG} \text{BOT} ) (our model)</td>
<td><strong>91.6</strong></td>
<td><strong>92.8</strong></td>
<td><strong>92.2</strong></td>
</tr>
</tbody>
</table>
References


[Li et al., 2018c] Jing Li, Aixin Sun, and Zhenchang Xing. Learning to answer programming questions with software documentation through social context embedding. *Information Sciences*, 448:36–52, 2018.


