A Unified Linear-Time Framework for Sentence-Level Discourse Parsing

Xiang Lin*¶, Shafiq Joty*¶§, Prathyusha Jwalapuram¶ and M Saiful Bari¶
¶Nanyang Technological University, Singapore
§Salesforce Research Asia, Singapore
{linx0057@e., srjoty@, jwal0001@e., bari0001@e.}ntu.edu.sg

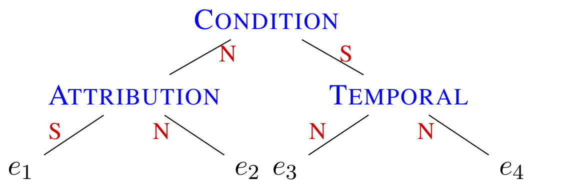
Abstract

We propose an efficient neural framework for sentence-level discourse analysis in accordance with Rhetorical Structure Theory (RST). Our framework comprises a discourse segmenter to identify the elementary discourse units (EDU) in a text, and a discourse parser that constructs a discourse tree in a top-down fashion. Both the segmenter and the parser are based on Pointer Networks and operate in linear time. Our segmenter yields an F_1 score of 95.4, and our parser achieves an F_1 score of 81.7 on the aggregated labeled (relation) metric, surpassing previous approaches by a good margin and approaching human agreement on both tasks (98.3 and 83.0 F_1).

1 Introduction

Coherence analysis of a text is a fundamental task in Natural Language Processing that can benefit many downstream applications. Rhetorical Structure Theory or RST (Mann and Thompson, 1988) is one of the most influential theories of text coherence. According to RST, a text is represented by a hierarchical structure known as a Discourse Tree (DT). As exemplified in Figure 1, the leaves of a DT correspond to contiguous atomic text spans called Elementary Discourse Units (EDUs). The adjacent EDUs and larger units are recursively connected by certain coherence relations (*e.g.,* ATTRIBUTION, EXPLANATION). The discourse units connected by a relation are further categorized based on their relative importance: NUCLEUS refers to the core part(s), while SATELLITE refers to the peripheral one. Coherence analysis in RST involves two subtasks: (*i*) breaking the text into a sequence of EDUs, referred to as **Discourse Segmentation**, and (*ii*) linking the EDUs into a DT, referred to as **Discourse Parsing**.

*equal contribution



[The Treasury also said]$_{e_1}$ [noncompetitive tenders will be considered timely]$_{e_2}$ [if postmarked no later than Sunday, Oct.29,]$_{e_3}$ [and received no later than tomorrow.]$_{e_4}$

Figure 1: An example discourse tree with four EDUs.

In this paper we consider sentence-level coherence analysis, which involves discourse segmentation and sentence-level parsing. For example, consider the DT in figure 1 for the sentence "The Treasury also said noncompetitive tenders will be considered timely if postmarked no later than Sunday, Oct.29, and received no later than tomorrow.", which has four EDUs as shown below the tree. Such sentence-level discourse annotations have been shown to be beneficial for a number of applications including machine translation (Guzmán et al., 2014) and sentence compression (Sporleder and Lapata, 2005). Furthermore, sentence-level analysis is considered to be a crucial step towards full text-level analysis. For example, automatic discourse segmentation has been shown to be the main source of inaccuracies in discourse parsing (Soricut and Marcu, 2003; Joty et al., 2012), and sentence-level parsing is considered as an essential first step in many existing discourse parsers (Feng and Hirst, 2014b; Joty et al., 2015) including the state-of-the-art one (Wang et al., 2017).

While earlier methods have mostly relied on hand-crafted lexical and syntactic features, recently researchers have shown competitive or even better results with neural models. One of the crucial advantages of **neural** models is that they can learn the feature representation of the discourse units in an end-to-end fashion. This capability is particularly enhanced through the use of effective pretrained word embeddings such as Glove (Pennington et al., 2014) that provide better generalization. Despite this, successful discourse parsers (Li et al., 2014; Ji and Eisenstein, 2014; Li et al., 2016) still needed to use hand-engineered features to outperform the non-neural models.

Another important distinction between existing methods is whether they employ a greedy **transition-based** algorithm (Marcu, 1999; Feng and Hirst, 2012, 2014b; Ji and Eisenstein, 2014; Braud et al., 2017; Li et al., 2016; Wang et al., 2017) or a globally optimized **chart parsing** algorithm (Soricut and Marcu, 2003; Li et al., 2014; Joty et al., 2015). Transition-based parsers build the tree incrementally by making a series of shift-reduce action decisions. The advantage of this method is that the parsing time is linear with respect to the number of EDUs (Sagae, 2009). The limitation, however, is that the decisions made at each step are based on local information, causing error propagation to subsequent steps. Also, when humans are asked to perform discourse analysis (segmentation and parsing), they tend to understand the full text first, before executing the tasks.

Methods based on chart parsing, on the other hand, learn scoring functions for discourse subtrees and perform dynamic programming search over all possible trees to find the most probable tree for a text. While these methods are more accurate than greedy parsers, they are generally slow, having a time complexity of $O(n^3M)$ for n EDUs and M different relations (Joty et al., 2015).

In this paper, we propose a unified neural framework for discourse segmentation and parsing based on Pointer Networks (Vinyals et al., 2015). Our parser employs a transition-based procedure to construct a discourse tree in a **top-down** fashion with the same computational efficiency, while still maintaining a global view of the input text. This is thanks to the **encoder-decoder** architecture that makes it possible to capture information from the whole text and the previously derived subtrees, while limiting the number of parsing steps to linear in the number of EDUs. Our framework is purely neural and does not rely on any hand-engineered features. Additionally, the framework allows us to train the segmentation and parsing models seamlessly with a joint objective.

We conduct a series of experiments with our framework on the standard RST Discourse Treebank (RST-DT) dataset for sentence level parsing, and our main findings are as follows.

- Our segmenter achieves an F_1 score of 95.4 giving a *relative error reduction of* 32% over the state-of-the-art segmenter.
- Evaluation of our sentence-level discourse parser with manual segmentation shows that it achieves an F_1 score of 81.3 on the relation labeling task yielding a *relative error reduction of about* 17% over the state-of-the-art parser.
- Joint training of the segmentation and parsing models improves the results further giving 95.5 F_1 on segmentation and 81.7 F_1 on parsing, while the human agreements on these two tasks are 98.3 and 83 F_1, respectively.
- Our end-to-end system (segmenter$\rightarrow$parser) reaches an F_1 of 77.5 on relation labeling providing an *absolute improvement of 10%* compared to the best existing system.
- Both our discourse segmenter and parser operate in linear time with respect to the number of EDUs. In practice, our segmenter and parser individually give 6.79x and 3.92x speedups, while the end-to-end system gives 5.9x speedup compared to the best open-sourced system.

We make our code available at `https://ntunlpsg.github.io/project/parser/pointer-net-parser`

2 Background

In this section, we give a brief overview of coherence analysis with RST and pointer networks.

2.1 Coherence Analysis with RST

Coherence analysis has been a long standing problem. We give a brief overview of the studies that are directly related to our method. Soricut and Marcu (2003) proposed SPADE, a system that uses generative models with syntactic features for discourse segmentation and sentence-level parsing. Subsequent research focuses on the impact of syntax in discourse analysis (Sporleder and Lapata, 2005; Fisher and Roark, 2007; Hernault et al., 2010). Joty et al. (2015) propose CODRA, a system that comprises a discourse segmenter and a two-stage discourse parser – one for sentence-level parsing and the other for multi-sentential parsing. Feng and Hirst (2014a) also propose two-stage parsing based on CRFs that use many hand-crafted features. Li et al. (2014) propose a recursive network for discourse parsing. Ji and Eisenstein (2014) present a representation learning method in a shift-reduce discourse parser. Wang et al. (2017) propose a two-stage parser, where they use shift-reduce parsing to first construct a tree structure with only nuclearity labels, and then in the second stage they identify the relations. They use SVMs with a large number of features. Wang et al. (2018) propose a discourse segmenter based on LSTM-CRF and achieve state-of-the-art results with ELMo. Li et al. (2018) also propose a segmenter based on pointer networks.

Pointer networks have also been used for summarization (See et al., 2017) and dependency parsing (Ma et al., 2018). In our work, we use pointer networks not only for segmentation but also for parsing, and we also show how the segmenter and parser can be trained jointly.

2.2 Pointer Networks

Sequence-to-sequence paradigms (Sutskever et al., 2014) provide the flexibility that the output sequence can be of a different length than the input sequence. However, they still require the output vocabulary size to be fixed a priori, which limits their applicability to problems where one needs to select (or point to) an element in the input sequence; that is, the size of the output vocabulary depends on the length of the input sequence. Pointer Networks (Vinyals et al., 2015) address this limitation by using **attention** (Bahdanau et al., 2015) as a pointing mechanism. Specifically, an encoder network first converts the input sequence $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ into a sequence of hidden states $\boldsymbol{H} = (\boldsymbol{h}_1, \ldots, \boldsymbol{h}_n)$. At each time step t, the decoder network receives the input from previous step and produces a decoder state $\boldsymbol{d}_t$ that modulates an attention over inputs. The output of the attention is a softmax distribution over the inputs.

$$s_{t,i} = \sigma(\boldsymbol{d}_t, \boldsymbol{h}_i) \quad \text{for } i = 1 \ldots n \tag{1}$$

$$\boldsymbol{a}_t = \text{softmax}(\boldsymbol{s}_t) = \frac{\exp(s_{t,i})}{\sum_{i=1}^{n} \exp(s_{t,i})} \tag{2}$$

where $\sigma(.,.)$ is a scoring function for attention, which can be another neural network or simply a dot product. We use $\boldsymbol{a}_t$ for inferring the output: $\hat{\mathrm{y}}_t = \arg\max(\boldsymbol{a}_t) = \arg\max p(\mathrm{y}_t|\mathrm{y}_{<t}, \boldsymbol{X}, \theta)$, where θ is the set of model parameters. To condition on y_{t-1}, the corresponding input $\boldsymbol{x}_{\mathrm{y}_{t-1}}$ is copied as the input to the decoder.

3 Our Discourse Parser

Given a sentence as input, the framework first employs our discourse segmenter to break the sentence into a sequence of EDUs. Our parser then links these EDUs into a labeled tree by identifying (*i*) which discourse units to relate (*i.e.,* finding the right **structure** of the tree), and (*ii*) what relations and nuclearity statuses to use in connecting them (*i.e.,* finding the correct **labels**). In the interests of presentational simplicity, we first describe the discourse parser in this section assuming that the EDUs have already been identified.

Model Overview. As shown in Figure 2, our parser uses a **pointer network** as its backbone parsing model. Given an input sentence containing n words $(w_1, \ldots, w_n)$, we first embed the words into their respective distributed representation by initializing them either randomly or with pretrained embeddings such as Glove (Pennington et al., 2014) or ELMo (Peters et al., 2018). The result of this is a sequence of word vectors $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$, which is fed to the network.

The encoder of the pointer network first composes the entire sentence sequentially into a sequence of hidden states $\boldsymbol{H} = (\boldsymbol{h}_1, \ldots, \boldsymbol{h}_n)$. The last hidden state of each EDU (*e.g.,* $\boldsymbol{h}_2, \boldsymbol{h}_4, \boldsymbol{h}_6, \boldsymbol{h}_8, \boldsymbol{h}_9$ and $\boldsymbol{h}_{10}$ in Figure 2) are selected to represent the corresponding EDU, thus, forming a sequence of EDU representations $\boldsymbol{E} = (\boldsymbol{e}_1, \ldots, \boldsymbol{e}_m)$. From this, the greedy decoder then constructs the discourse tree in a **top-down depth-first** manner.

The decoder maintains a **stack** $\mathcal{S}$ to keep track of the spans that need to be parsed further and their order (depth-first). $\mathcal{S}$ is initialized with the special *Root* symbol. At each decoding step t, the decoder extracts a span $\boldsymbol{e}_{i:j}$ from the top of $\mathcal{S}$, and uses the EDU representation $\boldsymbol{e}_j$ to generate a decoder hidden state $\boldsymbol{d}_t$, which is in turn used to compute the attention scores over the EDU representations in the selected range of spans ($\boldsymbol{e}_i$ to $\boldsymbol{e}_j$). Based on the attention scores, the decoder chooses a position k in the range to generate a new split $(\boldsymbol{e}_{i:k}, \boldsymbol{e}_{k+1:j})$. The parser then applies a **relation classifier** $\Phi_\theta(\boldsymbol{e}_{i:k}, \boldsymbol{e}_{k+1:j})$, parameterized by θ, on

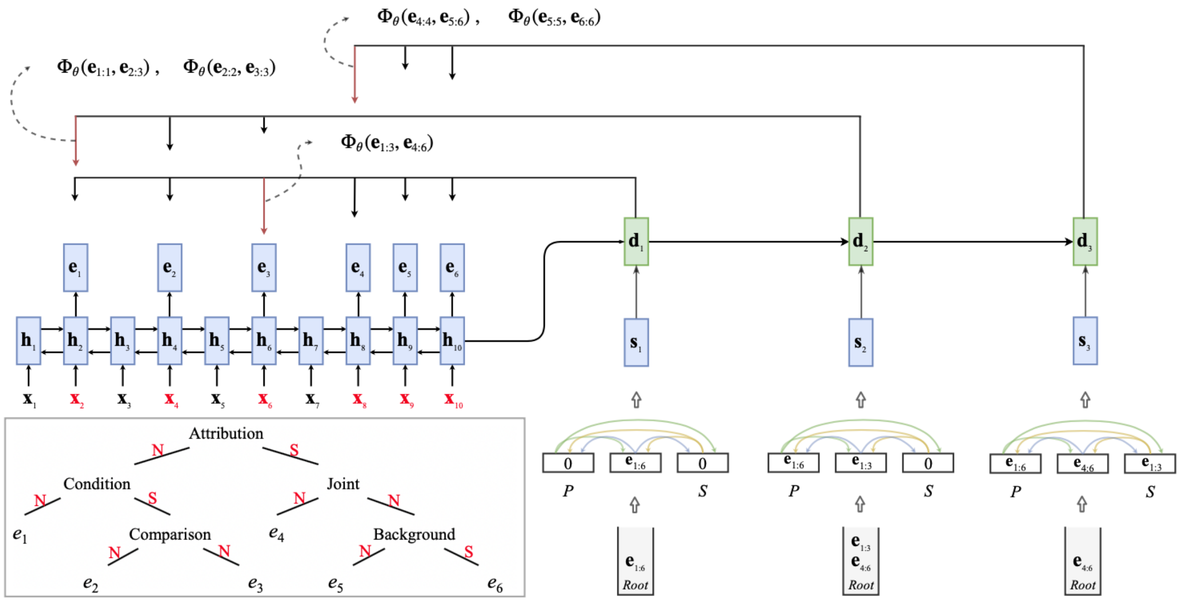


Figure 2: Our discourse parser along with the decoding process for a synthetic sentence with 10 words and 6 EDUs. EDU boundaries are marked in red color. For the inputs to the decoder at each step, P and S indicate the parent and sibling representations, respectively. $\Phi_\theta(\boldsymbol{e}_{i:k}, \boldsymbol{e}_{k+1:j})$ denotes the relation-nuclearity classifier employed by the parser to find the nuclearity and relation labels for the newly created spans, $\boldsymbol{e}_{i:k}$ and $\boldsymbol{e}_{k+1:j}$.

the new split to predict the relation and the nuclearity labels. If the length of any of the newly created spans ($\boldsymbol{e}_{i:k}$ and $\boldsymbol{e}_{k+1:j}$) is larger than two, the parser pushes it onto the stack. For the span containing only two EDUs, the parser would automatically run the classifier Φ_θ to predict the relation and nuclearity between the two EDUs.

Since the parser works in a depth-first manner, a text span is not parsed until a complete subtree for the preceding span is built (assuming we process the leftmost child first). This allows the decoder to exploit information from the generated subtrees in addition to the representation of the span being parsed. In the following paragraphs, we describe the components of our parser in detail.

The Encoder. Our parser uses a recurrent neural network (RNN) based on bidirectional Gated Recurrent Units or BiGRU (Cho et al., 2014) as the encoder. Like LSTM (Hochreiter and Schmidhuber, 1997), GRU cells are also designed to capture long range dependencies, but have fewer parameters than LSTM cells. In particular, our encoder uses six (6) recurrent layers of BiGRU cells, and generates hidden states $\boldsymbol{H} = (\boldsymbol{h}_1, \ldots, \boldsymbol{h}_n)$ by composing the word representations sequentially from left-to-right and from right-to-left, which is, $\boldsymbol{h}_i = [\overrightarrow{\boldsymbol{h}}_i; \overleftarrow{\boldsymbol{h}}_i]$ with $\overrightarrow{\boldsymbol{h}}_i$ and $\overleftarrow{\boldsymbol{h}}_i$ being the forward and the backward states. The last hidden states of an EDU are used as the EDU representation, generating a sequence of EDU representations $\boldsymbol{E} = (\boldsymbol{e}_1, \ldots, \boldsymbol{e}_m)$ for the input sentence.

The Decoder. Our parser uses a six-layer unidirectional GRU as the decoder. Instead of using the word embeddings, we feed our decoder with the corresponding encoder states for the span. This is because the encoder states contain more contextual information than the word embeddings (Ma et al., 2018). We use the representation of the last EDU as the representation of the span. For example, span $\boldsymbol{e}_{1:3}$ in Figure 2 is represented by $\boldsymbol{e}_3$ (or $\boldsymbol{h}_6$). We also experimented with taking the *mean* of the corresponding hidden states (*e.g.*, mean$(\boldsymbol{h}_1, \ldots, \boldsymbol{h}_6)$ for $\boldsymbol{e}_{1:3}$). We found the former to perform better in our experiments.

At each decoding step t, the decoder combines the span representation with its previous state $\boldsymbol{d}_{t-1}$ to generate the current state $\boldsymbol{d}_t$, which is then used to compute the attentions over the corresponding encoder states ($\boldsymbol{e}_1, \ldots, \boldsymbol{e}_3$ for $\boldsymbol{e}_{1:3}$). We use the simple **dot product** as the scoring function (*i.e.*, $\sigma(\boldsymbol{d}_t, \boldsymbol{h}_i)$ in Equation 2).

Remark: In our earlier attempts, we experimented with a self-attention based encoder-decoder with positional encoding similar to (Vaswani et al., 2017) to reduce the encoding time from $O(n)$ (linear) to $O(1)$ (constant) time. However, the performance was inferior to the RNN-based encoder.

3.1 The Relation Classifier

For relation labeling, we adopt a **bi-affine** classifier. The classifier is a two-layer neural network that takes two spans $\boldsymbol{e}_{i:k}$ and $\boldsymbol{e}_{k+1:j}$ as input and predicts the corresponding relation label and the nuclearity statuses. As before, we consider the representation of the last EDU as the representation of the span ($\boldsymbol{e}_k$ for $\boldsymbol{e}_{i:k}$ and $\boldsymbol{e}_j$ for $\boldsymbol{e}_{k+1:j}$). The first layer is a dense layer with Exponential Linear Unit (ELU) activations that maps the span representations $\boldsymbol{e}_k$ and $\boldsymbol{e}_j$ to latent label-specific features $\boldsymbol{c}_k$ and $\boldsymbol{c}_j$ of dimensions d.

$$\boldsymbol{c}_k = \mathrm{ELU}(\boldsymbol{e}_k^T U_1); \quad \boldsymbol{c}_j = \mathrm{ELU}(\boldsymbol{e}_j^T U_2) \qquad (3)$$

The second layer is a bi-affine layer with a softmax activation to get a multinomial distribution over the relation labels:

$$P_{\theta_l}(\mathrm{y}|\boldsymbol{X}) = \mathrm{softmax}(\boldsymbol{c}_k^T \mathbf{W}_{kj} \boldsymbol{c}_j + \boldsymbol{c}_k^T \boldsymbol{W}_k + \boldsymbol{c}_j^T \boldsymbol{W}_j + \mathrm{b}) \qquad (4)$$

where $\mathbf{W}_{kj} \in \mathbb{R}^{d \times d \times R}, \mathbf{W}_k \in \mathbb{R}^{d \times R}$, and $\mathbf{W}_j \in \mathbb{R}^{d \times R}$ are the weights and $\boldsymbol{b}$ is a bias vector with R being the number of relation labels. The bi-affine layer not only does a linear transformation of $\boldsymbol{c}_k$ and $\boldsymbol{c}_j$ but also models the correlation between $\boldsymbol{c}_k$ and $\boldsymbol{c}_j$ vectors (Dozat and Manning, 2016).

Following previous work, we attach the nuclearity statuses with the relation labels. For example, in Figure 1, the ATTRIBUTION relation between $\boldsymbol{e}_1$ as a satellite and $\boldsymbol{e}_2$ as a nucleus is jointly represented as ATTRIBUTION-SN. This representation allows us to perform the two tasks - relation identification and nuclearity assignment jointly.

3.2 Incorporating Partial Tree Information

As mentioned before, parsing a tree in a depth-first manner allows us to incorporate partial tree information while decoding a span. In this work, we consider information from the **parent** (P) and the immediate **left-sibling** (S) of the span being parsed (E). For example, in Figure 2, when parsing span $\boldsymbol{e}_{4:6}$, in addition to the current span, we consider its parent span $\boldsymbol{e}_{1:6}$ (represented by $\boldsymbol{e}_6$) and its left subtree span $\boldsymbol{e}_{1:3}$ (represented by $\boldsymbol{e}_3$). As the relative importance of the three components may vary, we put a self-attention layer before feeding them to the decoder. Formally, we put them as rows in a matrix $\boldsymbol{M} = [P; E; S]$ and perform:

$$\boldsymbol{A} = \mathrm{softmax}(\boldsymbol{M}\boldsymbol{M}^T)\boldsymbol{M} \qquad (5)$$

We take an element-wise sum of the three (row) vectors in $\boldsymbol{A}$ to form the final span representation ($\boldsymbol{s}$ in the figure) and feed it to the decoder.

3.3 Training Loss

Our parser is trained to minimize the sum of the loss for building the right tree **structure** and the loss for finding the correct **labels**. The *structure loss* $\mathcal{L}_s$ is the pointing loss for the pointer network:

$$\mathcal{L}_s(\theta_s) = -\sum_{t=1}^{T} \log P_{\theta_s}(\mathrm{y}_t|\mathrm{y}_{<t}, \boldsymbol{X}) \qquad (6)$$

where θ_s denotes the parameters of the encoder and the decoder, $y_{<t}$ represents the subtrees that have been generated by our parser at previous steps, and T is the number of spans containing more than two EDUs (pushed in the stack).

The *label loss* $\mathcal{L}_l$ is the cross entropy loss for the relation classifier, and can be defined as:

$$\mathcal{L}_l(\theta_l) = -\sum_{i=1}^{I}\sum_{r=1}^{R} y_{i,r} \log P_{\theta_l}(\mathrm{y}_i = r|\boldsymbol{X}) \qquad (7)$$

where θ_l are the parameters for the relation classifier (including the encoder), I is the number of spans with at least two EDUs, R is the total number of relation labels, and $y_{i,r}$ is the one-hot encoding of the relation label. We also apply an L_2-regularization on the parameters. Hence, the final parsing loss $\mathcal{L}_P(\theta_P)$ can be written as:

$$\mathcal{L}_{\mathrm{PAR}}(\theta_P) = \mathcal{L}_s(\theta_s) + \mathcal{L}_l(\theta_l) + \frac{\lambda}{2}\|\theta_P\|_2^2 \qquad (8)$$

where λ is the regularization strength and θ_P denotes the set of all parameters of the parser.

4 Our Discourse Segmenter

The job of the discourse segmenter is to find the EDU boundaries (if any) inside a sentence. Traditionally, discourse segmentation has been treated either as an binary classification problem (Soricut and Marcu, 2003; Fisher and Roark, 2007)

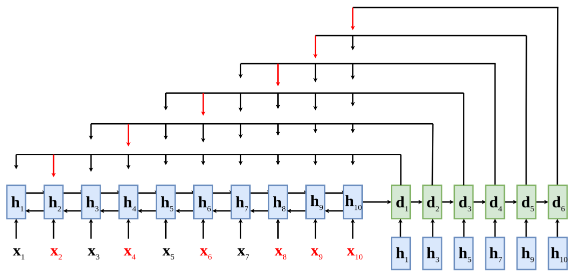


Figure 3: Our neural discourse segmentation model for the same synthetic sentence as in Figure 2. Words in red color denote boundary words.

or as a sequence labeling problem (Wang et al., 2018). Recently, Li et al. (2018) show the benefits of using pointer networks over previous methods for this task, achieving state-of-the-art results. In our work, we adopt their approach and advance the state-of-the-art further by simple modifications. More importantly, this framework allows us to train the discourse segmenter and the parser jointly with a shared encoder.

Model Description. Figure 3 depicts the architecture of our segmenter. Similar to our parser (Figure 2), the encoder of our segmentation model reads the whole sentence and transforms it into a sequence of hidden states. Then, at each time step, the decoder receives an encoder state corresponding to the first token of a segment currently being processed, and produces a decoder state which is in turn used to compute a distribution (attention) over all valid positions of the input sentence.

The encoder and the decoder have the same architecture as in (Li et al., 2018) with the following key improvements. First, following the same idea as in our parser, the decoder takes the encoder states as the input instead of word embeddings. Second, similar to our parser, we adopt dot product attention instead of an additive attention. Dot product attention is simple yet powerful, while using fewer parameters (Vaswani et al., 2017). Third, instead of simple look-up based embedding methods such as Glove, we use the contextual embedding ELMo that captures rich contextual information.

We train the model by minimizing the pointing loss with an L_2-regularization on the weights.

$$\mathcal{L}_{\text{SEG}}(\theta_S) = -\sum_{j=1}^{J} \log P_{\theta_S}(\mathrm{y}_j | \mathrm{y}_{<j}, \boldsymbol{X}) + \frac{\lambda}{2} \|\theta_S\|_2^2 \quad (9)$$

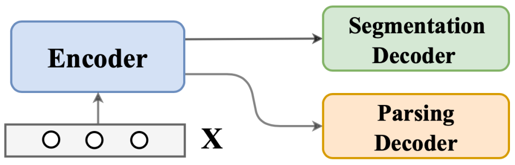


Figure 4: Joint training for segmentation and parsing.

where θ_S represents the model parameters and J is the number of EDUs in a sentence.

4.1 Joint and End-to-End Training

One crucial advantage of our framework is that it allows us to train the segmentation and the parsing models simultaneously and/or in an end-to-end fashion, while sharing a common encoder. Intuitively, both discourse segmentation and parsing can benefit from each other – a plausible segmentation can result in a plausible parse and vice versa. Such multitask learning was not possible in a non-neural setup and the two discourse analysis tasks have always been considered independently.

Figure 4 depicts the schematic diagram of our joint training process. The segmentation and the parsing models share a common encoder while having two separate decoders for the two tasks. The training objective can be written as:

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{SEG}}(\theta) + \mathcal{L}_s(\theta) + \mathcal{L}_l(\theta) + \frac{\lambda}{2} \|\theta\|_2^2 \quad (10)$$

where θ denotes the parameters of our joint model.

5 Experiments

In this section, we present our experiments on discourse segmentation and parsing.

5.1 Datasets

We train and evaluate our models on the standard RST Discourse Treebank (RST-DT) corpus (Carlson et al., 2002). RST-DT contains discourse annotations for 385 news articles from Penn Treebank (Marcus et al., 1994). The training data contains 347 documents (7673 sentences) and the test data contains 38 documents (991 sentences). In addition, 53 documents (1208 sentences) were annotated by two human annotators, which we use to compute human agreement scores.

Since we focus on sentence-level discourse analysis, we follow the same setup as Soricut and Marcu (2003); Joty et al. (2012). For segmentation, we utilize all 7673 sentences for training and

991 sentences for testing. For parsing, we extract sentence-level DTs from a document-level DT by finding the subtrees that span over the respective sentences. This gives 7321 sentence-level DTs for training, 951 for testing, and 1114 for getting human agreements. These numbers match the numbers reported by Joty et al. (2012). We randomly selected 10% of the data from the training set for hyperparameter tuning.

5.2 Discourse Segmentation Experiments

Settings. We compare our segmenter with five baselines: SPADE segmenter (Soricut and Marcu, 2003), F&R (Fisher and Roark, 2007), JCN (Joty et al., 2012), SegBot (Li et al., 2018), and WLY (Wang et al., 2018). Following the standard, we measure F_1-score based on the segmenter's ability to find the intra-sentential segment boundaries.

When we evaluate the WLY segmenter on the standard test set using their released pretrained model,[1] we get much lower results (90.5 F_1) than what they report in their paper (94.3 F_1). Upon investigation, we found that their experimental setting does not match with the standard one. Particularly, when extracting the sentences from the RST-DT dataset, instead of using gold tokenization, they use an automatic tokenizer, which gives fewer sentences – 865 test sentences instead of 991 and 6132 training sentences instead of 7673. This makes the scores artificially high.[2]

For a fair comparison with our model, we train and evaluate WLY and SegBot on the same dataset setting, and report the mean and standard deviation of five runs, each run with a different random seed. WLY uses ELMo embeddings, which we also use in our model. To train our model, we use Adam optimizer with a batch size of 80. We apply 0.2 dropout rate to the encoder and the decoder. The hidden sizes of the encoder, the decoder and the classifier are all set to 64. See Appendix for a complete list of hyperparameter settings. In all our experiments when comparing two systems, we use the *paired t-test* to measure statistical significance.

Results. Table 1 shows our segmentation results. As mentioned in Section 4, we implemented three key improvements on the top of (Li et al., 2018). Using encoder hidden states as decoder inputs and adopting dot product as the attention score function together gives 0.40%-7.29% relative improvement in F_1 over the first four baselines. Using ELMo, our segmenter outperforms all the baselines in all three measures. We achieve 2.3%-11.9%, 2.4%-11.3% and 2.3%-12.3% relative improvements in F_1, Recall and Precision, respectively. Jointly training with the parser improves this further (95.55 F_1). It is worthwhile to mention that our segmenter's performance of 95.55 F_1 is very close to the human agreement of 98.3 F_1. ELMo, as a transfer learning method, provides notable improvements. A similar observation was reported in (Wang et al., 2018). Surprisingly, the results with BERT were not as good. We suspect this is due to BERT's special tokenization.

Approach	**Precision**	**Recall**	**F_1**
Human Agreement	98.5	98.2	98.3
Baselines			
SPADE (Soricut and Marcu, 2003)	83.8	86.8	85.2
F&R (Fisher and Roark, 2007)	91.3	89.7	90.5
JCN (Joty et al., 2012)	88.0	92.3	90.1
SegBot$_{\text{glove}}$ (Li et al., 2018)	$91.08_{\pm 0.46}$	$91.03_{\pm 0.42}$	$91.05_{\pm 0.11}$
WLY$_{\text{ELMo}}$ (Wang et al., 2018)	$92.04_{\pm 0.43}$	$94.41_{\pm 0.53}$	$93.21_{\pm 0.33}$
Our Segmenter			
Pointer Net (Glove)	$90.55_{\pm 0.33}$	$92.29_{\pm 0.09}$	$91.41_{\pm 0.21}$
Pointer Net (BERT)	$92.05_{\pm 0.44}$	$95.03_{\pm 0.28}$	$93.51_{\pm 0.16}$
Pointer Net (ELMo)	$94.12_{\pm 0.20}^{\star}$	$96.63_{\pm 0.12}^{\star}$	$95.35_{\pm 0.10}^{\star}$
+ Joint training	$\mathbf{93.34}_{\pm 0.23}^{\star}$	$\mathbf{97.88}_{\pm 0.16}^{\star}$	$\mathbf{95.55}_{\pm 0.13}^{\star}$

Table 1: Discourse segmentation results. Superscript $\star$ indicates the model is significantly superior to the WLY$_{\text{ELMo}}$ model with a p-value < 0.01.

5.3 Discourse Parsing Experiments

Settings. We evaluate our parser in two different settings: (*a*) **parsing with gold segmentation**, and (*b*) **parsing with our automatic segmentation** or **end-to-end evaluation**. In the first setting, we compare our results with SPADE (Soricut and Marcu, 2003), DCRF (Joty et al., 2012), DPLP (Ji and Eisenstein, 2014), and the most recent 2-Stage Parser (Wang et al., 2017). SPADE and DCRF are both sentence-level parsers. However, DPLP and 2-Stage Parser are document-level parsers, and they do not report sentence-level performance. For DPLP, we feed the parser one sentence at a time to get a sentence-level DT. The 2-Stage Parser constructs a tree in multiple stages – first sentence-level, then paragraph-level, and finally document-level. We ran their parser to generate all the document-level DTs in the test set, from which we extract the sentence-level DTs to evaluate. By our count, this gives 881 valid sentence-level trees as opposed to 951. This is because like their discourse segmenter (WLY), they

[1] https://github.com/PKU-TANGENT/NeuralEDUSeg

[2] We confirmed this by communicating with the authors.

Approach	Span	Nuclearity	Relation
Human Agreement	95.7	90.4	83.0
Baselines			
SPADE (Soricut and Marcu, 2003)	93.5	85.8	67.6
DCRF (Joty et al., 2012)	94.6	86.9	77.1
DPLP (Ji and Eisenstein, 2014)	93.5	81.3	70.5
2-Stage Parser (Wang et al., 2017)	95.6	87.8	77.6
Our Parser			
Stack Pointer (ELMo-medium)	96.37	89.04*	79.03*
Stack Pointer (ELMo-large)	96.86*	90.77*	81.12*
+ Partial tree information	96.94*	90.89*	81.28*
+ Joint training	**97.44***	**91.34***	**81.70***

Table 2: Parsing results with gold segmentation. Superscript * indicates the model is significantly superior to the 2-Stage Parser with a p-value < 0.01.

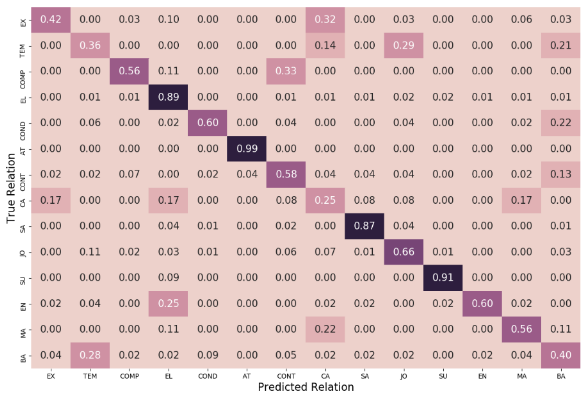


Figure 5: Confusion matrix for 14 most frequent relation labels: **EX**PLANATION, **TEM**PORAL, **COMP**ARISON, **EL**ABORATION, **COND**ITION, **AT**TRIBUTION, **CONT**RAST, **CA**USE, **SA**ME-UNIT, **JO**INT, **SU**MMARY, **EN**ABLEMENT, **MA**NNER-MEANS, and **BA**CKGROUND.

use an automatic tokenizer instead of gold tokenization. We evaluate their parser based on these 881 sentences. This is also what the authors suggested when contacted.

In our second setting for full system evaluation, we compare with the two existing end-to-end systems, SPADE and DCRF. The hyperparameters (learning rate, batch size, layer size) of our models in these two settings remain almost the same as the segmentation model (see Appendix for details).

Metric and Relation Labels. We evaluate the performance by using the standard unlabeled (Span) and labeled (Nuclearity, Relation) micro-averaged precision, recall and F_1-score as described in (Marcu, 2000). For brevity, we report only the F_1-scores here. Following previous work, we use the same 18 relations as used by previous studies, and we also attach the nuclearity statuses (NS, SN, NN) to these relations, giving a total of 39 distinctive relation labels.

Results with Gold Segmentation. We present the results in Table 2. Our base model (with ELMo-medium) outperforms all the existing methods to date in all three tasks. We achieve an absolute 1.43 F_1 improvement on the most difficult task of relation labeling, compared to the 2-stage parser (SOTA). Notably, the F_1 score of 96.37 for Span of our base model even exceeds the human agreement (F_1 score of 95.7) on the doubly-annotated data. As it ought to be, incorporating full-size ELMo boosts the performance in three tasks.

Our parser yields further improvements (+0.16 F_1 in Relation) by exploiting **partial tree** information generated in previous steps. The key component contributing to this improvement is the self-attention over original decoder inputs with partial tree information as described in Section 3.[3]

Thanks to the pointer network as the backbone of our model, we are able to train our segmenter and parser jointly by sharing the same encoder. The last row of Table 2 shows the results when we train the model jointly, and feed the parser with gold EDU segmentation during inference. The performance is improved further with **joint training**, achieving 97.44, 91.34, 81.70 F_1 score, in Span, Nuclearity and Relation, respectively. The results accord with our assumption that discourse segmentation and parsing may benefit from each other. Our parser surpasses human agreement in span and nuclearity. We are also approaching human agreement in the most difficult task of relation labeling. We show a confusion matrix for the relation labels in Figure 5. We see that our model gets confused between relations that are semantically similar (*e.g.,* **CA**USE vs. **EX**PLANATION, **COMP**ARISON vs. **CONT**RAST, and **TEM**PORAL vs. **JO**INT).

Remark: We observe that the relation labels in RST-DT are highly imbalanced, which makes the task harder. Therefore, we experimented with a variant of our parser where we had a separate classifier for nuclearity prediction, leaving 18 labels for relation classifier instead of 39. This model gave 96.74, 90.38, and 80.89 F_1 in Span, Nuclearity and Relation, respectively, which are lower than what we get by having a single classifier.

[3]Simple averaging of the vectors did not show any gain.

Approach	Span	Nuclearity	Relation
Baselines			
SPADE (Soricut and Marcu, 2003)	76.7	70.2	58.0
DCRF (Joty et al., 2012)	82.4	76.6	67.5
Our Model			
Stack Pointer (Pipeline)	91.14⋆	85.80⋆	76.94⋆
Stack Pointer (Joint training)	**91.75**⋆	**86.38**⋆	**77.52**⋆

Table 3: Parsing results with automatic segmentation. Superscript ⋆ indicates the model is significantly superior to the DCRF model with a p-value < 0.01.

Jointly modeling nuclearity and relation enforces the constraint that certain relations can have certain nuclearity orientations. For example, *Elaboration* and *Attribution* are mono-nuclear (takes either NS or SN), and SameUnit and Joint are multi-nuclear relations (takes only NN).

End-to-End Performance. Table 3 shows the results of our model and the two baselines. First, we use our segmenter followed by our best parser (independently trained) in a **pipeline**. The performance of this system is significantly better compared to the baselines. Against the best baseline (DCRF), it yields **8.74%, 9.2%, 9.44%** absolute improvements in Span, Nuclearity, Relation, respectively. We push the performance even further by **joint training** of the segmenter and parser as in Figure 4. Notice that the performance on Relation (77.5 F_1) is even better than the DCRF model with gold segmentation (77.1 F_1) in Table 2.

5.4 Run Time Analysis

As noted earlier, both our segmenter and parser operate in linear time with respect to the number of input units. We compare the speed (sentences per second) of our systems against other baselines in Table 4 from a practical viewpoint. We test all the systems with the same 100 sentences randomly selected from our test set on our machine (CPU: Intel Xeon W-2133, GPU: NVIDIA GTX 1080Ti). We include the model loading time for all the systems.[4] Since SPADE and CODRA need to extract a handful of features, they are typically slower than the neural models which use pretrained embeddings. In addition, CODRA's DCRF parser has a $O(n^3)$ inference time. Our segmenter is 6.8x faster than SPADE. Compared to CODRA (the fastest parser as of yet), our parser is 3.9x faster. Finally, our end-to-end system is 5.9x faster than the fastest system out there (SPADE), making our system not only effective but also highly efficient. Even when tested only on CPU, our model is faster than all the other models.

[4]As a neural model, WLY should be faster than the number we report. We retest both WLY and our model by excluding the model loading time. The speed of WLY and our segmenter are 157.80 sents/s and 181.30 sents/s, respectively. This could be because the two models are implemented in different frameworks (WLY: TensorFlow, ours: PyTorch).

System	Speed (Sents/s)	Speedup
Only Segmenter		
CODRA (Joty et al., 2015)	3.06	1.0x
WLY (Wang et al., 2018)	4.30	1.4x
SPADE (Soricut and Marcu, 2003)	5.24	1.7x
Our (CPU)	12.05	3.9x
Our (GPU)	35.54	11.6x
Only Parser		
SPADE (Soricut and Marcu, 2003)	5.07	1.0x
CODRA (Joty et al., 2015)	7.77	1.5x
Our (CPU)	12.57	2.5x
Our (GPU)	30.45	6.0x
End-to-End (Segmenter $\rightarrow$ Parser)		
CODRA (Joty et al., 2015)	3.05	1.0x
SPADE (Soricut and Marcu, 2003)	4.90	1.6x
Our (CPU)	11.99	3.9x
Our (GPU)	28.96	9.5x

Table 4: Speed comparison of our systems with other open-sourced systems.

6 Conclusions

We have proposed a unified framework for sentence-level discourse analysis based on pointer networks that constructs a discourse tree in linear time. Both our segmenter and parser achieve state-of-the-art results outperforming existing systems by a wide margin, without using any hand-crafted features. We also train the segmenter and the parser jointly through the encoder-decoder architecture and improve the results further. Apart from the effectiveness, our system is 6 times faster than the fastest available system.

Based on what we have done so far, it is natural for us to move our focus from sentence-level to document-level RST parsing. Also, in the future, we would like to investigate how our approach can be generalized to other discourse frameworks such as the Penn Discourse Treebank (PDTB).

Acknowledgments

We would like to extend our thanks for the funding support from MOE Tier-1 (Grant M4011897.020).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *ICLR*.

Chloé Braud, Maximin Coavoux, and Anders Søgaard. 2017. Cross-lingual rst discourse parsing. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 292–304. Association for Computational Linguistics.

Lynn Carlson, Daniel Marcu, and Mary Ellen Okurowski. 2002. RST Discourse Treebank (RST–DT) LDC2002T07. *Linguistic Data Consortium, Philadelphia*.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724–1734.

Timothy Dozat and Christopher D. Manning. 2016. Deep biaffine attention for neural dependency parsing. *CoRR*, abs/1611.01734.

Vanessa Feng and Graeme Hirst. 2012. Text-level Discourse Parsing with Rich Linguistic Features. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, ACL '12, pages 60–68, Jeju Island, Korea. ACL.

Vanessa Feng and Graeme Hirst. 2014a. A Linear-Time Bottom-Up Discourse Parser with Constraints and Post-Editing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, ACL '14, pages 511–521, Baltimore, USA. ACL.

Vanessa Wei Feng and Graeme Hirst. 2014b. A linear-time bottom-up discourse parser with constraints and post-editing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 511–521. Association for Computational Linguistics.

Seeger Fisher and Brian Roark. 2007. The Utility of Parse-derived Features for Automatic Discourse Segmentation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, ACL'07, pages 488–495, Prague, Czech Republic. ACL.

Francisco Guzmán, Shafiq Joty, Lluís Màrquez, and Preslav Nakov. 2014. Using discourse structure improves machine translation evaluation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 687–698, Baltimore, Maryland. ACL.

Hugo Hernault, Helmut Prendinger, David duVerle, and Mitsuru Ishizuka. 2010. HILDA: A Discourse Parser Using Support Vector Machine Classification. *Dialogue and Discourse*, 1(3):1–33.

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

Yangfeng Ji and Jacob Eisenstein. 2014. Representation learning for text-level discourse parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13–24, Baltimore, Maryland. ACL.

Shafiq Joty, Giuseppe Carenini, and Raymond Ng. 2012. A novel discriminative framework for sentence-level discourse analysis. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 904–915. Association for Computational Linguistics.

Shafiq Joty, Giuseppe Carenini, and Raymond T Ng. 2015. Codra: A novel discriminative framework for rhetorical analysis. *Computational Linguistics*, 41:3:385–435.

Jing Li, Aixin Sun, and Shafiq Joty. 2018. Segbot: A generic neural text segmentation model with pointer network. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence and the 23rd European Conference on Artificial Intelligence*, IJCAI-ECAI-2018, pages 4166 – 4172, Stockholm, Sweden.

Jiwei Li, Rumeng Li, and Eduard Hovy. 2014. Recursive deep models for discourse parsing. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2061–2069, Doha, Qatar. ACL.

Qi Li, Tianshi Li, and Baobao Chang. 2016. Discourse parsing with attention-based hierarchical neural networks. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 362–371. Association for Computational Linguistics.

Xuezhe Ma, Zecong Hu, Jingzhou Liu, Nanyun Peng, Graham Neubig, and Eduard H. Hovy. 2018. Stack-pointer networks for dependency parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 1403–1414.

William Mann and Sandra Thompson. 1988. Rhetorical Structure Theory: Toward a Functional Theory of Text Organization. *Text*, 8(3):243–281.

Daniel Marcu. 1999. The automatic construction of large-scale corpora for summarization research. In *Proceedings of SIGIR*, pages 137–144.

Daniel Marcu. 2000. The Rhetorical Parsing of Unrestricted Texts: A Surface-based Approach. *Computational Linguistics*, 26:395–448.

Mitchell Marcus, Mary Marcinkiewicz, and Beatrice Santorini. 1994. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

Kenji Sagae. 2009. Analysis of discourse structure with syntactic dependencies and data-driven shift-reduce parsing. In *Proceedings of the 11th International Conference on Parsing Technologies (IWPT'09)*, pages 81–84. Association for Computational Linguistics.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083. Association for Computational Linguistics.

Radu Soricut and Daniel Marcu. 2003. Sentence Level Discourse Parsing Using Syntactic and Lexical Information. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL'03, pages 149–156, Edmonton, Canada. ACL.

Caroline Sporleder and Mirella Lapata. 2005. Discourse Chunking and its Application to Sentence Compression. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT-EMNLP'05, pages 257–264, Vancouver, British Columbia, Canada. ACL.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2692–2700. Curran Associates, Inc.

Yizhong Wang, Sujian Li, and Houfeng Wang. 2017. A two-stage parsing method for text-level discourse analysis. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 184–188. Association for Computational Linguistics.

Yizhong Wang, Sujian Li, and Jingfeng Yang. 2018. Toward fast and accurate neural discourse segmentation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP 2018)*.