RST Parsing from Scratch

Thanh-Tung Nguyen[†¶], **Xuan-Phi Nguyen**[†¶], **Shafiq Joty**[¶§], **Xiaoli Li**[†¶]
[¶]Nanyang Technological University
[§]Salesforce Research Asia
[†]Institute for Infocomm Research, A-STAR
Singapore
{ng0155ng@e.;nguyenxu002@e.;srjoty@}ntu.edu.sg
xlli@i2r.a-star.edu.sg

Abstract

We introduce a novel top-down end-to-end formulation of document level discourse parsing in the Rhetorical Structure Theory (RST) framework. In this formulation, we consider discourse parsing as a sequence of splitting decisions at token boundaries and use a seq2seq network to model the splitting decisions. Our framework facilitates discourse parsing from scratch without requiring discourse segmentation as a prerequisite; rather, it yields segmentation as part of the parsing process. Our unified parsing model adopts a beam search to decode the best tree structure by searching through a space of high scoring trees. With extensive experiments on the standard English RST discourse treebank, we demonstrate that our parser outperforms existing methods by a good margin in both end-to-end parsing and parsing with gold segmentation. More importantly, it does so without using any handcrafted features, making it faster and easily adaptable to new languages and domains.

1 Introduction

In a document, the clauses, sentences and paragraphs are logically connected together to form a coherent discourse. The goal of discourse parsing is to uncover this underlying coherence structure, which has been shown to benefit numerous NLP applications including text classification (Ji and Smith, 2017), summarization (Gerani et al., 2014), sentiment analysis (Bhatia et al., 2015), machine translation evaluation (Joty et al., 2017) and conversational machine reading (Gao et al., 2020).

Rhetorical Structure Theory or RST (Mann and Thompson, 1988), one of the most influential theories of discourse, postulates a hierarchical discourse structure called discourse tree (DT). The leaves of a DT are clause-like units, known as elementary discourse units (EDUs). Adjacent EDUs and higher-order spans are connected hierarchically through coherence relations (*e.g., Contrast, Explanation*). Spans connected through a relation are categorized based on their relative importance — *nucleus* being the main part, with *satellite* being the subordinate one. Fig. 1 exemplifies a DT spanning over two sentences and six EDUs. Finding discourse structure generally requires breaking the text into EDUs (discourse segmentation) and linking the EDUs into a DT (discourse parsing).

Discourse parsers can be singled out by whether they apply a bottom-up or top-down procedure. Bottom-up parsers include transition-based models (Feng and Hirst, 2014; Ji and Eisenstein, 2014; Braud et al., 2017; Wang et al., 2017) or globally optimized chart parsing models (Soricut and Marcu, 2003; Joty et al., 2013, 2015). The former constructs a DT by a sequence of shift and reduce decisions, and can parse a text in asymptotic running time that is linear in number of EDUs. However, the transition-based parsers make greedy local decisions at each decoding step, which could propagate errors into future steps. In contrast, chart parsers learn scoring functions for sub-trees and adopt a CKY-like algorithm to search for the highest scoring tree. These methods normally have higher accuracy but suffer from a slow parsing speed with a complexity of $\mathcal{O}(n^3)$ for n EDUs. The top-down parsers are relatively new in discourse (Lin et al., 2019; Zhang et al., 2020; Kobayashi et al., 2020). These methods focus on finding splitting points in each iteration to build a DT. However, the local decisions could still affect the performance as most of the methods are still greedy.

Like most other fields in NLP, language parsing has also undergone a major paradigm shift from traditional feature-based statistical parsing to end-to-end neural parsing. Being able to parse a document end-to-end from scratch is appealing for several key reasons. First, it makes the overall development procedure easily adaptable to new languages, domains and tasks by surpassing the expensive feature engineering step that often requires more time

Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 1613–1625
June 6–11, 2021.

and domain/language expertise. Second, the lack of an explicit feature extraction phase makes the training and testing (decoding) faster.

Because of the task complexity, it is only recently that neural approaches have started to outperform traditional feature-rich methods. However, successful document level neural parsers still rely heavily on handcrafted features (Ji and Eisenstein, 2014; Yu et al., 2018; Zhang et al., 2020; Kobayashi et al., 2020). Therefore, even though these methods adopt a neural framework, they are not "end-to-end" and do not enjoy the above mentioned benefits of an end-to-end neural parser. Moreover, in existing methods (both traditional and neural), discourse segmentation is detached from parsing and treated as a prerequisite step. Therefore, the errors in segmentation affect the overall parsing performance (Soricut and Marcu, 2003; Joty et al., 2012).

In view of the limitations of existing approaches, in this work we propose an end-to-end top-down document level parsing model that:

- Can generate a discourse tree from scratch without requiring discourse segmentation as a prerequisite step; rather, it generates the EDUs as a by-product of parsing. Crucially, this novel formulation facilitates solving the two tasks in a single neural model. Our formulation is generic and works in the same way when it is provided with the EDU segmentation.
- Treats discourse parsing as a sequence of splitting decisions at token boundaries and uses a seq2seq pointer network (Vinyals et al., 2015) to model the splitting decisions at each decoding step. Importantly, our seq2seq parsing model can adopt beam search to widen the search space for the highest scoring tree, which to our knowledge is also novel for the parsing problem.
- Does not rely on any handcrafted features, which makes it faster to train or test, and easily adaptable to other domains and languages.
- Achieves the state of the art (SoTA) with an F_1 score of 46.6 in the Full (label+structure) metric for end-to-end parsing on the English RST Discourse Treebank, which outperforms many parsers that use gold EDU segmentation. With gold segmentation, our model achieves a SoTA F_1 score of 50.2 (Full), outperforming the best existing system by 2.1 absolute points. More imporantly, it does so without using any handcrafted features (not even part-of-speech tags).

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

2 Model

Assuming that a document has already been segmented into EDUs, following the traditional approach, the corresponding discourse tree (DT) can be represented as a set of labeled constituents.

$$\mathbb{C} := \{((i_t, k_t, j_t), r_t) | i_t \leq k_t < j_t\}_{t=1}^{m} \quad (1)$$

where $m = |\mathbb{C}|$ is the number of internal nodes in the tree and r_t is the relation label between the discourse unit containing EDUs i_t through k_t and the one containing EDUs $k_t + 1$ through j_t.

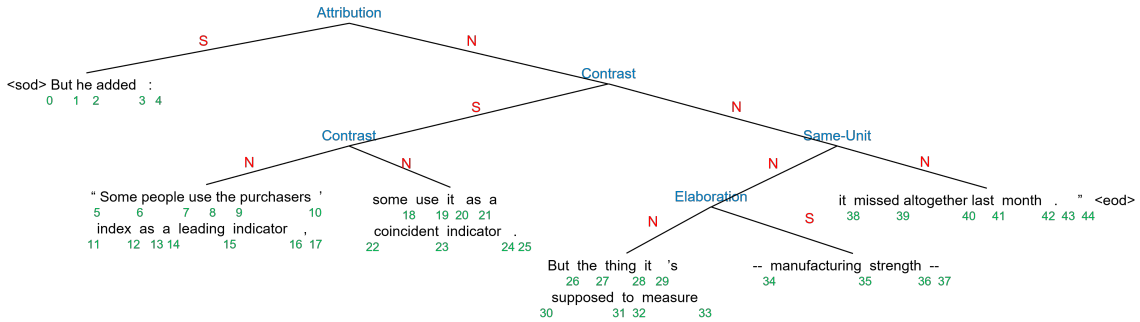
Traditionally, in RST parsing, discourse segmentation is performed first to obtain the sequence of EDUs, which is followed by the parsing process to assemble the EDUs into a labeled tree. In other words, traditionally discourse segmentation and parsing have been considered as two distinct tasks that are solved by two different models.

On the contrary, in this work we take a radically different approach that directly starts with parsing the (unsegmented) document in a top-down manner and treats discourse segmentation as a special case of parsing that we get as a by-product. Importantly, this novel formulation of the problem allows us to solve the two problems in a single neural model. Our parsing model is generic and also works in the same way when it is fed with an EDU-segmented text. Before presenting the model architecture, we first formulate the problem as a splitting decision problem at the token level.

2.1 Parsing as a Splitting Decision Problem

We reformulate the discourse parsing problem from Eq. (1) as a sequence of splitting decisions at *token boundaries* (instead of EDUs). Specifically, the input text is first prepended and appended with the special start (<sod>) and end (<eod>) tokens, respectively. We define the token-boundary as the indexed position between two consecutive tokens. For example, the constituent spanning "But he added :" in Fig. 2 is defined as $(0, 4)$.

Following the standard practice, we convert the discourse tree by transforming each multi-nuclear constituent into a hierarchical right-branching binary sub-tree. Every internal node in the resulting binary tree will have a left and a right constituent, allowing us to represent it by its split into the left and right children. Based on this, we define the



Boundary-based splitting representation when EDUs are provided

$$\mathbb{S}_{\text{edu}} = \{(0,44)\rightarrow 4,\ (4,44)\rightarrow 25,\ (4,25)\rightarrow 17,(25,44)\rightarrow 37,\ (25,37)\rightarrow 33\}$$

Boundary-based splitting representation for end-to-end parsing

$$\mathbb{S} = \{(0,44)\rightarrow 4,\ (\mathbf{0},\mathbf{4})\rightarrow \mathbf{4},\ (4,44)\rightarrow 25,\ (4,25)\rightarrow 17,\ (\mathbf{4},\mathbf{17})\rightarrow \mathbf{17},\ (\mathbf{17},\mathbf{25})\rightarrow \mathbf{25},$$
$$(25,44)\rightarrow 37,\ (25,37)\rightarrow 33,\ (\mathbf{25},\mathbf{33})\rightarrow \mathbf{33},\ (\mathbf{33},\mathbf{37})\rightarrow \mathbf{37},\ (\mathbf{37},\mathbf{44})\rightarrow \mathbf{44}\}$$

Figure 1: A discourse tree for two sentences in the RST discourse treebank. The internal nodes (*e.g., Attribution*, *Contrast*) denote the coherence relations and the edge labels reflect the nuclearity of the child span. Below the tree, we show the sequence of splitting decisions $\mathbb{S}_{\text{edu}}$ when EDUs are provided and $\mathbb{S}$ when EDUs are not provided (end-to-end parsing). The **bold** splitting decision represents the final split of the span, forming an EDU.

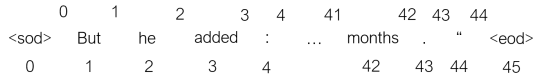


Figure 2: Relation between token-boundary (above) and token (below) representations. A token-boundary position k is located between the tokens at k and $k+1$.

parsing as a set of splitting decisions $\mathbb{S}$ at token-boundaries by the following proposition:

Proposition 1 *Given a binarized discourse tree for a document containing n tokens, the tree can be converted into a set of token-boundary splitting decisions $\mathbb{S} = \{(i,j)\rightarrow k | i < k \leq j\}$ such that the parent constituent (i,j) either gets split into two child constituents (i,k) and (k,j) for $k < j$, or forms a terminal EDU unit for $k = j$,* i.e., *the span will not be split further (*i.e., *marks segmentation).*

Notice that $\mathbb{S}$ is a generalized formulation of RST parsing, which also includes the decoding of EDUs as a special case ($k = j$). It is quite straightforward to change this formulation to the parsing scenario, where discourse segmentation (sequence of EDUs) is provided. Formally, in that case, the tree can be converted into a set of splitting decisions $\mathbb{S}_{\text{edu}} = \{(i,j)\rightarrow k | i < k < j\}$ such that the constituent (i,j) gets split into two constituents (i,k) and (k,j) for $k < j$, *i.e.,* we simply omit the special case of $k = j$ as the EDUs are given. In other words, in our generalized formulation, discourse segmentation is just one extra step of parsing, and can be done top-down end-to-end.

An example of our formalism of the parsing problem is shown in Fig. 1 for a discourse tree spanning over two sentences (44 tokens); for simplicity, we do not show the relation labels corresponding to the splitting decisions (marked by $\rightarrow$). Since each splitting decision corresponds to one and only one internal node in the tree, it guarantees that the transformation from the tree to $\mathbb{S}$ (and $\mathbb{S}_{\text{edu}}$) has a one-to-one mapping. Therefore, predicting the sequence of such splitting decisions is equivalent to predicting the discourse tree (DT).

Seq2Seq Parsing Model. In this work, we adopt a structure-then-label framework. Specifically, we factorize the probability of a DT into the probability of the tree structure and the probability of the relations (*i.e.,* the node labels) as follows:

$$P_\theta(DT|\boldsymbol{x}) = P_\theta(\mathbb{S},\mathbb{L}|\boldsymbol{x}) = P_\theta(\mathbb{L}|\mathbb{S},\boldsymbol{x})P_\theta(\mathbb{S}|\boldsymbol{x}) \quad (2)$$

where $\boldsymbol{x}$ is the input document, and $\mathbb{S}$ and $\mathbb{L}$ respectively denote the structure and labels of the DT. This formulation allows us to first infer the best tree structure (*e.g.,* using beam search), and then find the corresponding labels.

As discussed, we consider the structure prediction problem as a sequence of splitting decisions to generate the tree in a top-down manner. We use a seq2seq pointer network (Vinyals et al., 2015) to model the sequence of splitting decisions (Fig. 3). We adopt a depth-first order of the decision sequence, which showed more consistent

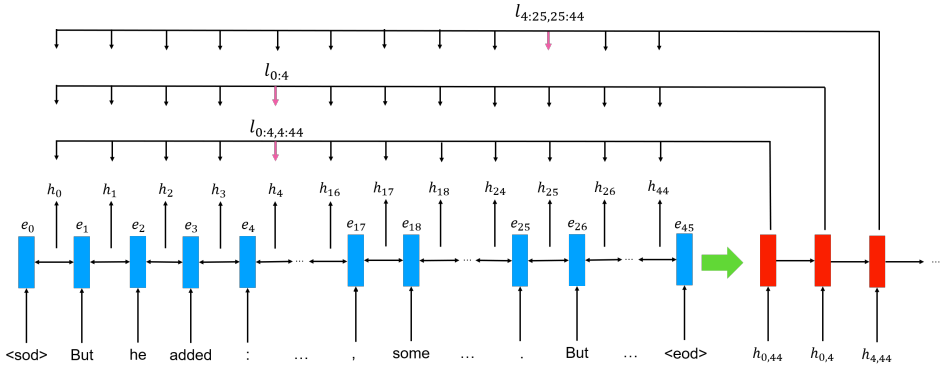


Figure 3: Our discourse parser along with a few decoding steps for a given document. The input to the decoder at each step is the representation of the span to be split. We predict the splitting point using the biaffine function between the corresponding decoder state and the token-boundary encoder representations. The figure is for end-to-end parsing, where each EDU-corresponding span points to its right edge to mark the EDU. The coherence relations between the left and right spans are assigned using a label classifier after the (approximately) optimal tree structure is formed using beam search.

performance in our preliminary experiments than other alternatives, such as breath-first order.

First, we encode the tokens in a document $\boldsymbol{x} = (x_0, \ldots, x_n)$ with a document encoder and get the token-boundary representations $(\boldsymbol{h}_0, \ldots, \boldsymbol{h}_n)$. Then, at each decoding step t, the model takes as input an internal node (i_t, j_t), and produces an output y_t (by pointing to the token boundaries) that represents the splitting decision $(i_t, j_t) \rightarrow k_t$ to split it into two child constituents (i_t, k_t) and (k_t, j_t). For example, the initial span $(0, 44)$ in Fig. 1 is split at boundary position 4, yielding two child spans $(0, 4)$ and $(4, 44)$. If the span $(0, 4)$ is given as an EDU (*i.e.,* segmentation given), the splitting stops at $(0, 4)$, thus omitted in $\mathbb{S}_{\text{edu}}$ (Fig. 1). Otherwise, an extra decision $(0, 4) \rightarrow 4 \in \mathbb{S}$ needs to be made to mark the EDUs for end-to-end parsing. With this, the probability of $\mathbb{S}$ can be expressed as:

$$
\begin{aligned}
P_\theta(\mathbb{S}|\boldsymbol{x}) &= \prod_{y_t \in \mathbb{S}} P_\theta(y_t | y_{<t}, \boldsymbol{x}) \\
&= \prod_{t=1}^{|\mathbb{S}|} P_\theta\Big((i_t, j_t) \rightarrow k_t | ((i, j) \rightarrow k)_{<t}, \boldsymbol{x}\Big)
\end{aligned}
$$

This end-to-end conditional splitting formulation is the main novelty of our method and is in contrast to previous approaches which rely on offline-inferred EDUs from a separate discourse segmenter. Our formalism streamlines the overall parsing process, unifies the neural components seamlessly and smoothens the training process.

2.2 Model Architecture

In the following, we describe the components of our parsing model: the document encoder, the boundary and span representations, the decoding process through the decoder and the label classifier.

Document Encoder. Given an input document of n words $\boldsymbol{x} = (x_1, \ldots, x_n)$, we first add <sod> and <eod> markers to the sequence. After that, each token x_i in the sequence is mapped into its dense vector representation $\boldsymbol{e}_i$ as: $\boldsymbol{e}_i = [\boldsymbol{e}_i^{\text{char}}, \boldsymbol{e}_i^{\text{word}}]$, where $\boldsymbol{e}_i^{\text{char}}$, and $\boldsymbol{e}_i^{\text{word}}$ are respectively the character and word embeddings of token x_i. For word embedding, we experiment with (*i*) randomly initialized, (*ii*) pretrained static embeddings ,*e.g.,* GloVe (Pennington et al., 2014)). To represent the character embedding of a token, we apply a character bidirectional LSTM *i.e.,* Bi-LSTM (Hochreiter and Schmidhuber, 1997) or pretrained contextualized embeddings, *e.g.,* XLNet (Yang et al., 2019). The token representations are then passed to a sequence encoder of a three-layer Bi-LSTM to obtain their forward $\boldsymbol{f}_i$ and backward $\boldsymbol{b}_i$ contextual representations.

Token-boundary Span Representations. To represent each token-boundary position k between token positions k and $k+1$, we use the fencepost representation (Cross and Huang, 2016):

$$
\boldsymbol{h}_k = [\boldsymbol{f}_k; \boldsymbol{b}_{k+1}] \tag{3}
$$

where $\boldsymbol{f}_k$ and $\boldsymbol{b}_{k+1}$ are the forward and backward LSTM hidden vectors of positions k and $k+1$ respectively, and $[\cdot;\cdot]$ is the concatenation operation.

Then, to represent the token-boundary span (i, j), we use the linear combination of the two endpoints i and j as:

$$
\boldsymbol{h}_{i,j} = \boldsymbol{W}_1 \boldsymbol{h}_i + \boldsymbol{W}_2 \boldsymbol{h}_j \tag{4}
$$

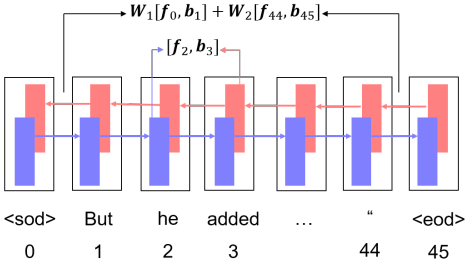


Figure 4: Illustration of token-boundary span encoder. The figure lays out an example representation for the boundary at 2 and the representation of the token-boundary span $(0, 44)$, which corresponds to the whole document.

where $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$ are trainable weights. These span representations will be used as input to the decoder or the label classifier. Fig. 4 illustrates an example boundary span representation.

The Decoder. Our model uses a unidirectional LSTM as the decoder. At each decoding step t, the decoder takes as input the corresponding span (i, j) (*i.e.,* $\boldsymbol{h}_{i,j}$) and its previous LSTM state $\boldsymbol{d}_{t-1}$ to generate the current state $\boldsymbol{d}_t$ and then the biaffine function (Dozat and Manning, 2017) is applied between $\boldsymbol{d}_t$ and *all* the encoded token-boundary representations $(\boldsymbol{h}_0, \boldsymbol{h}_1, \ldots, \boldsymbol{h}_n)$ as follows:

$$\boldsymbol{d}'_t = \text{MLP}_d(\boldsymbol{d}_t) \quad \boldsymbol{h}'_i = \text{MLP}_h(\boldsymbol{h}_i) \tag{5}$$

$$s_t^i = \boldsymbol{d}_t'^T \boldsymbol{W}_{dh} \boldsymbol{h}'_i + \boldsymbol{h}_i'^T \boldsymbol{w}_h \tag{6}$$

$$a_t^i = \frac{\exp(s_t^i)}{\sum_{i=0}^{n} \exp(s_t^i)} \quad \text{for } i = 0, \ldots, n \tag{7}$$

where each MLP operation comprises a linear transformation with LeakyReLU activation (Maas et al., 2013) to transform $\boldsymbol{d}_i$ and $\boldsymbol{h}_i$ into equal-sized vectors $\boldsymbol{d}'_t, \boldsymbol{h}'_i \in \mathbb{R}^d$, and $\boldsymbol{W}_{dh} \in \mathbb{R}^{d \times d}$ and $\boldsymbol{w}_h \in \mathbb{R}^d$ are respectively the weight matrix and weight vector for the biaffine function. The resulting biaffine scores s_t^i are then fed into a *softmax* layer to acquire the pointing distribution $\boldsymbol{a}_t^i \in [0, 1]^{n+1}$ for the splitting decision. During inference, when decoding the tree at step t, we only examine the "valid" splitting points between i and j, and we look for k such that $i < k \leq j$.

Label Classifier. We perform label assignment after decoding the entire tree structure. Each assignment takes into account the splitting decision that generated it since the label represents the relation between the child spans. Specifically, for a constituent (i, j) that was split into two child constituents (i, k) and (k, j), we determine the coherence relation between them as follows:

$$\boldsymbol{h}_{ik}^l = \text{MLP}_l([\boldsymbol{h}_i; \boldsymbol{h}_k]); \quad \boldsymbol{h}_{kj}^r = \text{MLP}_r([\boldsymbol{h}_k; \boldsymbol{h}_j]) \tag{8}$$

$$P_\theta(l|(i,k),(k,j)) = \text{softmax}\Big((\boldsymbol{h}_{ik}^l)^T \mathbf{W}_{lr} \boldsymbol{h}_{kj}^r + (\boldsymbol{h}_{ik}^l)^T \boldsymbol{W}_l + (\boldsymbol{h}_{kj}^r)^T \boldsymbol{W}_r + \boldsymbol{b}\Big) \tag{9}$$

$$l^*_{(i,k),(k,j)} = \arg\max_{l \in L} P_\theta(l|(i,k),(k,j)) \tag{10}$$

where L is the total number of labels (*i.e.,* coherence relations with nuclearity attached); each of MLP_l and MLP_r includes a linear transformation with LeakyReLU activation to transform the left and right spans into equal-sized vectors $\boldsymbol{h}_{ik}^l, \boldsymbol{h}_{kj}^r \in \mathbb{R}^d$; $\mathbf{W}_{lr} \in \mathbb{R}^{d \times L \times d}$, $\boldsymbol{W}_l \in \mathbb{R}^{d \times L}$, $\boldsymbol{W}_r \in \mathbb{R}^{d \times L}$ are the weights and $\boldsymbol{b}$ is a bias vector.

Training Objective. Our parsing model is trained by minimizing the total loss defined as:

$$\mathcal{L}(\theta_e, \theta_d, \theta_l) = \mathcal{L}_s(\theta_e, \theta_d) + \mathcal{L}_l(\theta_e, \theta_l) \tag{11}$$

where structure $\mathcal{L}_s$ and label $\mathcal{L}_l$ losses are cross-entropy losses computed for the splitting and labeling tasks respectively, and θ_e, θ_d and θ_l denote the encoder, decoder and labeling parameters.

2.3 Complete Discourse Parsing Models

Having presented the generic framework, we now describe how it can be easily adapted to the two parsing scenarios: (*i*) end-to-end parsing and (*ii*) parsing with EDUs. We also describe the incorporation of beam search for inference.

End-to-End Parsing. As mentioned, previous work for end-to-end parsing assumes a separate segmenter that provides EDU-segmented texts to the parser. Our method, however, is an end-to-end framework that produces both the EDUs as well as the parse tree in the same inference process. To guide the search better, we incorporate an inductive bias into our inference based on the finding that most sentences have a well-formed subtree in the document-level tree (Soricut and Marcu, 2003), *i.e.,* discourse structure tends to align with the text structure (sentence boundary in this case); for example, Fisher and Roark (2007); Joty et al. (2013) found that more than 95% of the sentences have a well-formed subtree in the RST discourse treebank.

Our goal is to ensure that each sentence corresponds to an internal node in the tree. This can be achieved by a simple adjustment in our inference. When decoding at time step t with the span (i_t, j_t) as input, if the span contains $M > 0$ sentence boundaries within it, we pick the one that

Algorithm 1 Discourse Tree Inference (end-to-end)

Input: Document length n; boundary encoder states: $(\boldsymbol{h}_0, \boldsymbol{h}_1, \ldots, \boldsymbol{h}_n)$; sentence boundary set $\mathbb{SB}$; label scores: $P(l|(i,k),(k,j))$, $0 \leq i < k \leq j \leq n, l \in L$, initial decoder state st.
Output: Parse tree DT

$\mathcal{ST} = [(0, n)]$ ▷*stack of spans*
$\mathcal{S} = []$
while $\mathcal{ST} \neq \varnothing$ **do**
  $(i, j) = \text{pop}(\mathcal{ST})$ ▷*Current span to split*
  $\boldsymbol{a}, st = dec(st, \boldsymbol{h}_{i,j})$ ▷$\boldsymbol{a}$*: split prob. dist.*
  if $(i, j) \cap \mathbb{SB} \neq \varnothing$ **then**
    $k = \arg\max_{i<k\leq j \ \&\ k \in \mathbb{SB}} \boldsymbol{a}$
  else
    $k = \arg\max_{i<k\leq j} \boldsymbol{a}$
  end if
  if $j - 1 > k > i + 1$ **then**
    $\text{push}(\mathcal{ST}, (k, j))$
    $\text{push}(\mathcal{ST}, (i, k))$
  else if $j - 1 > k = i + 1$ **then**
    $\text{push}(\mathcal{ST}, (k, j))$
  else if $k = j - 1 > i + 1$ **then**
    $\text{push}(\mathcal{ST}, (i, k))$
  end if
  if $k \neq j$ **then**
    $\text{push}(\mathcal{S}((i, k, j))$
  end if
end while
$DT = [((i, k, j), argmax_l P(l|(i,k)(k,j)) \forall (i, k, j) \in \mathcal{S}]$

Algorithm 2 Discourse Tree Inference with Beam Search (with gold EDUs)

Input: Number of EDUs in document m; beam width B; EDU boundary-based encoder states: $(\overline{\boldsymbol{h}}_0, \ldots, \overline{\boldsymbol{h}}_m)$; label scores: $P_\theta(l|(i,k),(k,j))$, $0 \leq i < k < j \leq m, l \in \{1, \ldots, L\}$, initial decoder state $\boldsymbol{s}$.
Output: Parse tree DT

$L_d = m - 1$ ▷*Decoding length*
beam = array of L_d items ▷*List of empty beam items*
init_input_span= $[(0, m), (0, 0), \ldots, (0, 0)]$ ▷m*-2 paddings (0,0)*
init_tree= $[(0, 0, 0), (0, 0, 0), \ldots, (0, 0, 0)]$ ▷m*-1 elements*
beam[0] = $(0, \boldsymbol{s}, init\_input\_span$, init_tree$)$ ▷*Initialize first item (log-prob,state,input_span,tree)*
for $t = 1$ **to** L_d **do**
  for (logp, $\boldsymbol{s}$, input-span, tree) $\in$ beam$[t-1]$ **do**
    (i, j) = input-span$[t-1]$ ▷*Current span to split*
    $\boldsymbol{a}, \boldsymbol{s}'$ = decoder-step$(\boldsymbol{s}, \overline{\boldsymbol{h}}_{i,j})$ ▷$\boldsymbol{a}$*: split prob. dist.*
    for $(k, p_k) \in$ top-$B(\boldsymbol{a})$ **and** $i < k < j$ **do**
      curr-input-span = input-span
      curr-tree = tree
      curr-tree$[t-1] = (i, k, j)$
      if $k > i + 1$ **then**
        curr-input-span$[t] = (i, k)$
      end if
      if $j > k + 1$ **then**
        curr-input-span$[t + j - k - 1] = (k, j)$
      end if
      push (logp + $\log(p_k)$, $\boldsymbol{s}'$, curr-input-span, curr-tree) to beam[t]
    end for
  end for
  prune beam[t] ▷*Keep top-B highest score trees*
end for
logp*, $\boldsymbol{s}^*$, $\boldsymbol{ip}^*$, $\mathbb{S}^* = \arg\max_{\text{logp}}$ beam$[L_d]$ ▷$\mathbb{S}^*$*: best structure*
DT $= [(i, k, j, \arg\max_l P_\theta(l|(i,k),(k,j)) \ \forall (i, k, j) \in \mathcal{S}^*]$

has the highest pointing score (Eq. 7) among the M alternatives as the split point k_t. If there is no sentence boundary within the input span ($M = 0$), we find the next split point as usual. In other words, sentence boundaries in a document get the chance to be split before the token boundaries inside a sentence. This constraint is indeed similar to the 1S-1S (1 subtree for 1 sentence) constraint of Joty et al. (2013)'s bottom-up parsing, and is also consistent with the property that EDUs are always within the sentence boundary. Algorithm 1 illustrate the end-to-end inference algorithm.

Parsing with EDUs. When segmentation information is provided, we can have a better encoding of the EDUs to construct the tree. Specifically, rather than simply taking the token-boundary representation corresponding to the EDU boundary as the EDU representation, we adopt a hierarchical approach, where we add another Bi-LSTM layer (called "Boundary LSTM") that connects EDU boundaries (a figure of this framework is in the Appendix). In other words, the input sequence to this LSTM layer is $(\overline{\boldsymbol{h}}_0, \ldots, \overline{\boldsymbol{h}}_m)$, where $\overline{\boldsymbol{h}}_0 = \boldsymbol{h}_0$, $\overline{\boldsymbol{h}}_m = \boldsymbol{h}_n$ and $\overline{\boldsymbol{h}}_j \in \{\boldsymbol{h}_1, \ldots, \boldsymbol{h}_{n-1}\}$ such that $\overline{\boldsymbol{h}}_j$ is an EDU boundary. For instance, for the example in Fig. 1, the input to the Boundary LSTM layer is $(\boldsymbol{h}_0, \boldsymbol{h}_4, \boldsymbol{h}_{17}, \boldsymbol{h}_{25}, \boldsymbol{h}_{33}, \boldsymbol{h}_{37}, \boldsymbol{h}_{44})$.

This hierarchical representation facilitates better modeling of relations between EDUs and higher order spans, and can capture long-range dependencies better, especially for long documents.

Incorporating Beam Search. Previous work (Lin et al., 2019; Zhang et al., 2020) which also uses a seq2seq architecture, computes the pointing scores over the token or span representations only within the input span. For example, for an input span (i, j), the pointing scores are computed considering only $(\boldsymbol{h}_i, \ldots, \boldsymbol{h}_j)$ as opposed to $(\boldsymbol{h}_1, \ldots, \boldsymbol{h}_n)$ in our Eq. 7. This makes the scales of the scores uneven across different input spans as the lengths of the spans vary. Thus, such scores cannot be objectively compared across sub-trees globally at the full-tree level. In addition, since efficient global search methods like beam search cannot be applied properly with non-uniform scores, these previous methods had to remain greedy at each decoding step. In contrast, our decoder points to all the encoded token-boundary representations in every step (Eq. 7). This ensures that the pointing scores are evenly scaled, allowing fair comparisons between the scores of all candidate sub-trees. Therefore, our method enables the effective use of beam search through highly probable candidate trees. Algorithm 2 illustrates the beam search inference when EDUs are given.

3 Experiments

We conduct our experiments on discourse parsing with and without gold segmentation. We use the standard English RST Discourse Treebank or RST-DT (Lynn et al., 2002) for training and evaluation.

It consists of 385 annotated Wall Street Journal news articles: 347 for training and 38 for testing. We randomly select 10% of the training set as our development set for hyper-parameter tuning. Following prior work, we adopted the same 18 courser relations defined in (Carlson and Marcu, 2001). For evaluation, we report the standard metrics Span, Nuclearity, Relation and Full F1 scores, computed using the standard Parseval (Morey et al., 2017, 2018) and RST-Parseval (Marcu, 2000) metrics.

3.1 Parsing with Gold Segmentation

Settings. Discourse parsing with gold EDUs has been the standard practice in many previous studies. We compare our model with ten different baselines as shown in Table 1. We report most results from Morey et al. (2018); Zhang et al. (2020); Kobayashi et al. (2020), while we reproduce Yu et al. (2018) using their provided source code.

For our model setup, we use the encoder-decoder framework with a 3-layer Bi-LSTM encoder and 3-layer unidirectional LSTM decoder. The LSTM hidden size is 400, the word embedding size is 100 for random initialization, while the character embedding size is 50. The hidden dimension in MLP modules and biaffine function for structure prediction is 500. The beam width B is 20. Our model is trained by Adam optimizer (Kingma and Ba, 2015) with a batch size of 10000 tokens. Our learning rate is initialized at 0.002 and scheduled to decay at an exponential rate of 0.75 for every 5000 steps. Model selection for testing is performed based on the Full F1 score on the development set. When using pretrained word embeddings, we use the 100D vectors from GloVe (Pennington et al., 2014). For pretrained model, we use the *XLNet-base-cased* version (Yang et al., 2019).[1] The pretrained models/embeddings are kept frozen during training.

Results. From the results in Table 1, we see that our model with GloVe (static) embeddings achieves a Full F1 score of 46.8, the highest among all the parsers that do not use pretrained models (or contextual embeddings). This suggests that a BiLSTM-based parser can be competitive with effective modeling. The model also outperforms the one proposed by Zhang et al. (2020), which is closest to ours in terms of modelling, by 3.9%, 4.1%, 2.4% and 2.5% absolute in Span, Nuclearity, Relation and Full, respectively. More importantly, our system achieves such results without relying on external data or features, in contrast to previous approaches. In addition, by using XLNet-base pretrained model, our system surpasses all existing methods (with or without pretraining) in all four metrics, achieving the state of the art with 2.9%, 4.0%, 2.4% and 2.1% absolute improvements. It also reduces the gap between system performance and human agreement. When evaluated with the RST-Parseval (Marcu, 2000) metric, our model outperforms the baselines by 0.6%, 1.4% and 1.8% in Span, Nuclearity and Relation, respectively.

Systems	Span	Nuc	Rel	Full
Parseval Metric (Morey et al., 2017)				
Human Agreement	78.7	66.8	57.1	55.0
Ji and Eisenstein (2014)$^{+}$	64.1	54.2	46.8	46.3
Feng and Hirst (2014)$^{+}$	68.6	55.9	45.8	44.6
Joty et al. (2015)$^{+}$	65.1	55.5	45.1	44.3
Li et al. (2016)$^{+}$	64.5	54.0	38.1	36.6
Braud et al. (2016)	59.5	47.2	34.7	34.3
Braud et al. (2017)*	62.7	54.5	45.5	45.1
Yu et al. (2018)$^{+\S}$	71.4	60.3	49.2	48.1
Zhang et al. (2020)$^{+}$	67.2	55.5	45.3	44.3
Our with GloVe	71.1	59.6	47.7	46.8
Our with XLNet§	**74.3**	**64.3**	**51.6**	**50.2**
RST-Parseval Metric (Marcu, 2000)				
Human Agreement	88.7	77.3	65.4	
Yu et al. (2018)$^{+\S}$	85.5	73.1	60.2	
Wang et al. (2017)$^{+\S}$	86.0	72.4	59.7	
Kobayashi et al. (2020)$^{+\S}$	87.0	74.6	60.0	
Our with XLNet§	**87.6**	**76.0**	**61.8**	

Table 1: Parsing results with gold segmentation. The sign $^{+}$ denotes that systems use handcrafted features such as lexical, syntactic, sentence/paragraph boundary features and so on, * denotes that systems use external cross-lingual features and § means that systems use pretrained models.

3.2 End-to-end Parsing

For end-to-end parsing, we compare our method with the model proposed by Zhang et al. (2020). Their parsing model uses the EDU segmentation from Li et al. (2018). Our method, in contrast, predicts the EDUs along with the discourse tree in a unified process (§2.3). In terms of model setup, we use a setup identical to the experiments with gold segmentation (§3.1).

Table 2 reports the performance for document-level end-to-end parsing. Compared to Zhang et al. (2020), our model with GloVe embeddings yields 1.5%, 2.9%, 2.4% and 2.5% absolute gains in Span, Nuclearity, Relation and Full F1 scores, respectively. Furthermore, the model with XLNet

[1] Our initial attempt with BERT did not offer significant gain as BERT is not explicitly designed to process long documents and has a limit of maximum 512 tokens.

Model	Span	Nuc	Rel	Full
Zhang et al. (2020)	62.3	50.1	40.7	39.6
Our model				
with GloVe	63.8	53.0	43.1	42.1
with XLNet	**68.4**	**59.1**	**47.8**	**46.6**

Table 2: End-to-end parsing performance.

Model	Span	Nuc	Rel	Full
Final model	71.1	59.6	47.7	46.8
~~Beam search~~	70.1	58.1	46.8	45.8
~~Boundary LSTM~~	68.5	55.5	46.1	44.7

Table 3: Ablation test of our models *with* gold EDUs. ~~Beam search~~ indicates the full model with greedy decoding (beam width 1), while ~~Boundary LSTM~~ is the full model with greedy decoding and no LSTM connection between EDU-boundary representations.

achieves even better performance and outperforms many models that use gold segmentation (Table 1).

EDU Segmentation Results. Our end-to-end parsing method gets an F1 score of 96.30 for the resulting EDUs. Our result rivals existing SoTA segmentation methods – 92.20 F1 of Li et al. (2018) and 95.55 F1 of Lin et al. (2019). This shows the efficacy of our unified framework for not only discourse parsing but also segmentation.[2]

3.3 Ablation Study

To further understand the contributions from the different components of our unified parsing framework, we perform an ablation study by removing selected components from a network trained with the best set of parameters.

With Gold Segmentation. Table 3 shows two ablations for parsing with gold EDUs. We see that both beam search and boundary LSTM (hierarchical encoding as shown in Fig. 7) are important to the model. The former can find better tree structure by searching a larger searching space. The latter, meanwhile, connects the EDU-boundary representations, which enhances the model's ability to capture long-range dependencies between EDUs.

[2]We could not compare our segmentation results with the DISRPT 2019 Shared Task (Zeldes et al., 2019) participants. We found few inconsistencies in the settings. First, in their "gold sentence" dataset, instead of using the gold sentence, they pre-process the text with an automatic tokenizer and sentence segmenter. Second, in the evaluation, under the same settings, they do not exclude the trivial BeginSegment label at the beginning of each sentence which we exclude in evaluating our segmentation result (following the RST standard).

Model	Span	Nuc	Rel	Full
Final model (GloVe)	63.8	53.0	43.1	42.1
~~GloVe~~	63.3	52.3	42.4	41.4
~~Sentence guidance~~	59.2	48.8	40.7	38.9

Table 4: Ablation test of our end-to-end model. ~~GloVe~~ is the full model with randomized word embeddings, while ~~Sentence guidance~~ is the full model with randomized word embeddings and without sentence guidance.

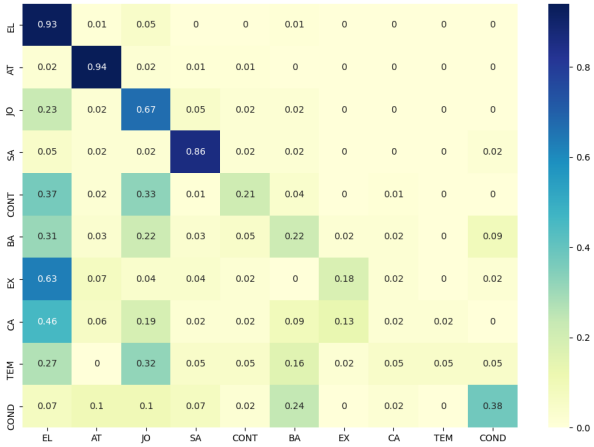


Figure 5: Confusion matrix for the 10 most frequent relations on the RST–DT test set. The vertical axis represents true and horizontal axis represents predicted relations. The relations are: Elaboration (EL), Attribution (AT), Joint (JO), Same-Unit (SA), Contrast (CONT), Background (BA), Explanation (EX), Cause (CA), Temporal (TEM), Condition (COND).

End-to-end Parsing. For end-to-end parsing, Table 4 shows that the sentence boundary constraint (§2.3) is indeed quite important to guide the model as it decodes long texts. Since sentence segmentation models are quite accurate, they can be employed if ground truth sentence segmentation is not available. We also notice that pretraining (GloVe) leads to improved performance.

Error Analysis. We show our best parser's (with gold EDUs) confusion matrix for the 10 most frequent relation labels in Fig. 5. The complete matrix with the 18 relations is shown in Appendix (Fig. 8). The imbalanced relation distribution in RST-DT affects our model's performance to some extent. Also semantic similar relations tend to be confused with each other. Fig. 6 shows an example where our model mistakenly labels Summary as Elaboration. However, one could argue that the relation Elaboration is also valid here because the parenthesized text brings additional information (the equivalent amount of money). We show more error examples in the Appendix (Fig. 9 - 11), where our parser la-

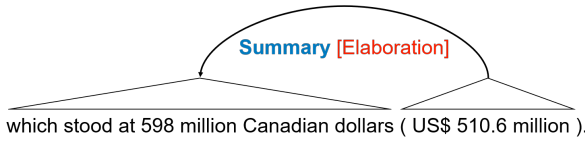


Figure 6: An error example where our system incorrectly labels a Summary as Elaboration.

System	Gold Seg.	End-to-End	Time (s)	Speedup
(Feng and Hirst, 2014)	✓		210	1.0x
(Yu et al., 2018)	✓		79	2.7x
Our parser (Glove)	✓		19	11.1x
Our parser (XLNet)	✓		33	6.4x
Our parser (GloVe)		✓	45	4.7x
Our parser (XLNet)		✓	60	3.5x

Table 5: The wall time for parsing the RST-DT test set.

bels a Condition as Background, Temporal as Joint and Explanation as Elaboration. As we can see, all these relations are semantically close and arguably interchangeable.

3.4 Parsing Speed

Table 5 compares the parsing speed of our models with a representative non-neural (Feng and Hirst, 2014) and neural model (Yu et al., 2018). We measure speed empirically using the wall time for parsing the test set. We ran the baselines and our models under the same settings (CPU: Intel Xeon W-2133 and GPU: Nvidia GTX 1080 Ti).

With gold-segmentation, our model with GloVe embeddings can parse the test set in 19 seconds, which is up to 11 times faster than (Feng and Hirst, 2014), and this is when their features are precomputed. The speed gain can be attributed to (*i*) to the efficient GPU implementation of neural modules to process the decoding steps, and (*ii*) the fact that our model does not need to compute any handcrafted features. With pretrained models, our parser with gold segmentation is about 2.4 times faster than (Yu et al., 2018). Our end-to-end parser that also performs segmentation is faster than the baselines that are provided with the EDUs. Nonetheless, we believe there is still room for speed improvement by choosing a better network, like the Longformer (Beltagy et al., 2020) which has an $\mathcal{O}(1)$ parallel time complexity in encoding a text, compared to the $\mathcal{O}(n)$ complexity of the recurrent encoder.

4 Related Work

Discourse analysis has been a long-established problem in NLP. Prior to the neural tsunami in NLP, discourse parsing methods commonly employed statistical models with handcrafted features (Soricut and Marcu, 2003; Hernault et al., 2010; Feng and Hirst, 2014; Joty et al., 2015). Even within the neural paradigm, most previous studies still rely on external features to achieve their best performances (Ji and Eisenstein, 2014; Wang et al., 2017; Braud et al., 2016, 2017; Yu et al., 2018). These parsers adopt a bottom-up approach, either transition-based or chart-based parsing.

Recently, top-down parsing has attracted more attention due to its ability to maintain an overall view of the input text. Inspired by the Stack-Pointer network (Ma et al., 2018) for dependency parsing, Lin et al. (2019) first propose a seq2seq model for sentence-level parsing. Zhang et al. (2020) extend this to the document level. Kobayashi et al. (2020) adopt a greedy splitting mechanism for discourse parsing inspired by Stern et al. (2017)'s work in constituency parsing. By using pretrained models/embeddings and extra features (*e.g.,* syntactic, text organizational features), these models achieve competitive results. However, their decoder infers a tree greedily.

Our approach differs from previous work in that it can perform end-to-end discourse parsing in a single neural framework without needing segmentation as a prerequisite. Our model can parse a document from scratch without relying on any external features. Moreover, it can apply efficient beam search decoding to search for the best tree.

5 Conclusion

We have presented a novel top-down end-to-end method for discourse parsing based on a seq2seq model. Our model casts discourse parsing as a series of splitting decisions at token boundaries, which can solve discourse parsing and segmentation in a single model. In both end-to-end parsing and parsing with gold segmentation, our parser achieves state-of-the-art, surpassing existing methods by a good margin, without relying on handcrafted features. Our parser is not only more effective but also more efficient than the existing ones.

This work leads us to several future directions. Our short-term goal is to improve the model with better architecture and training mechanisms. For example, joint training on discourse and syntactic parsing tasks could be a good future direction since both tasks are related and can be modeled within our unified conditional splitting framework. We also plan to extend our parser to other languages.

References

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv:2004.05150*.

Parminder Bhatia, Yangfeng Ji, and Jacob Eisenstein. 2015. Better document-level sentiment analysis from RST discourse parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2212–2218, Lisbon, Portugal. Association for Computational Linguistics.

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, Valencia, Spain. Association for Computational Linguistics.

Chloé Braud, Barbara Plank, and Anders Søgaard. 2016. Multi-view and multi-task training of RST discourse parsers. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1903–1913, Osaka, Japan. The COLING 2016 Organizing Committee.

Lynn Carlson and Daniel Marcu. 2001. Discourse tagging reference manual. Technical Report ISI-TR-545, University of Southern California Information Sciences Institute.

James Cross and Liang Huang. 2016. Span-based constituency parsing with a structure-label system and provably optimal dynamic oracles. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1–11, Austin, Texas. Association for Computational Linguistics.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Vanessa Wei Feng and Graeme Hirst. 2014. 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, Baltimore, Maryland. 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 of Computational Linguistics*, pages 488–495, Prague, Czech Republic. Association for Computational Linguistics.

Yifan Gao, Chien-Sheng Wu, Jingjing Li, Shafiq Joty, Steven C.H. Hoi, Caiming Xiong, Irwin King, and Michael Lyu. 2020. Discern: Discourse-aware entailment reasoning network for conversational machine reading. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2439–2449, Online. Association for Computational Linguistics.

Shima Gerani, Yashar Mehdad, Giuseppe Carenini, Raymond T. Ng, and Bita Nejat. 2014. Abstractive summarization of product reviews using discourse structure. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1602–1613, Doha, Qatar. Association for Computational Linguistics.

Hugo Hernault, Helmut Prendinger, David A. 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. Association for Computational Linguistics.

Yangfeng Ji and Noah A. Smith. 2017. Neural discourse structure for text categorization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 996–1005, Vancouver, Canada. Association for Computational Linguistics.

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*, EMNLP-CoNLL'12, pages 904–915, Jeju Island, Korea. ACL.

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

Shafiq Joty, Giuseppe Carenini, Raymond T. Ng, and Yashar Mehdad. 2013. Combining Intra- and Multi-sentential Rhetorical Parsing for Document-level Discourse Analysis. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, ACL'13, pages 486–496, Sofia, Bulgaria. ACL.

Shafiq Joty, Francisco Guzmán, Lluís Màrquez, and Preslav Nakov. 2017. Discourse structure in machine translation evaluation. *Computational Linguistics*, 43(4):683–722.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Naoki Kobayashi, Tsutomu Hirao, Hidetaka Kamigaito, Manabu Okumura, and Masaaki Nagata. 2020. Top-down rst parsing utilizing granularity levels in documents. In *Proceedings of the 2020 Conference on Artificial Intelligence for the American (AAAI)*, pages 8099–8106.

Jing Li, Aixin Sun, and Shafiq Joty. 2018. Segbot: A generic neural text segmentation model with pointer network. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 4166–4172. International Joint Conferences on Artificial Intelligence Organization.

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, Austin, Texas. Association for Computational Linguistics.

Xiang Lin, Shafiq Joty, Prathyusha Jwalapuram, and M Saiful Bari. 2019. A unified linear-time framework for sentence-level discourse parsing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4190–4200, Florence, Italy. Association for Computational Linguistics.

Carlson Lynn, Daniel Marcu, and Mary Ellen Okurowski. 2002. Rst discourse treebank (rst–dt) ldc2002t07. *Linguistic Data Consortium*.

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

Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng. 2013. Rectifier nonlinearities improve neural network acoustic models. In *in ICML Workshop on Deep Learning for Audio, Speech and Language Processing*.

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

Daniel Marcu. 2000. The rhetorical parsing of unrestricted texts: a surface-based approach. *Computational Linguistics*, 26(3):395–448.

Mathieu Morey, Philippe Muller, and Nicholas Asher. 2017. How much progress have we made on RST discourse parsing? a replication study of recent results on the RST-DT. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1319–1324, Copenhagen, Denmark. Association for Computational Linguistics.

Mathieu Morey, Philippe Muller, and Nicholas Asher. 2018. A dependency perspective on RST discourse parsing and evaluation. *Computational Linguistics*, 44(2):197–235.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Radu Soricut and Daniel Marcu. 2003. Sentence level discourse parsing using syntactic and lexical information. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 228–235.

Mitchell Stern, Jacob Andreas, and Dan Klein. 2017. A minimal span-based neural constituency parser. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 818–827.

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, Vancouver, Canada. Association for Computational Linguistics.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*, volume 32, pages 5753–5763. Curran Associates, Inc.

Nan Yu, Meishan Zhang, and Guohong Fu. 2018. Transition-based neural RST parsing with implicit syntax features. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 559–570, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Amir Zeldes, Debopam Das, Erick Galani Maziero, Juliano Antonio, and Mikel Iruskieta. 2019. The DISRPT 2019 shared task on elementary discourse unit segmentation and connective detection. In *Proceedings of the Workshop on Discourse Relation Parsing and Treebanking 2019*, pages 97–104, Minneapolis, MN. Association for Computational Linguistics.

Longyin Zhang, Yuqing Xing, Fang Kong, Peifeng Li, and Guodong Zhou. 2020. A top-down neural architecture towards text-level parsing of discourse rhetorical structure. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6386–6395, Online. Association for Computational Linguistics.

Appendix

6 Parsing with EDUs

Figure 7 shows first few decoding steps with final parsers with EDUs.

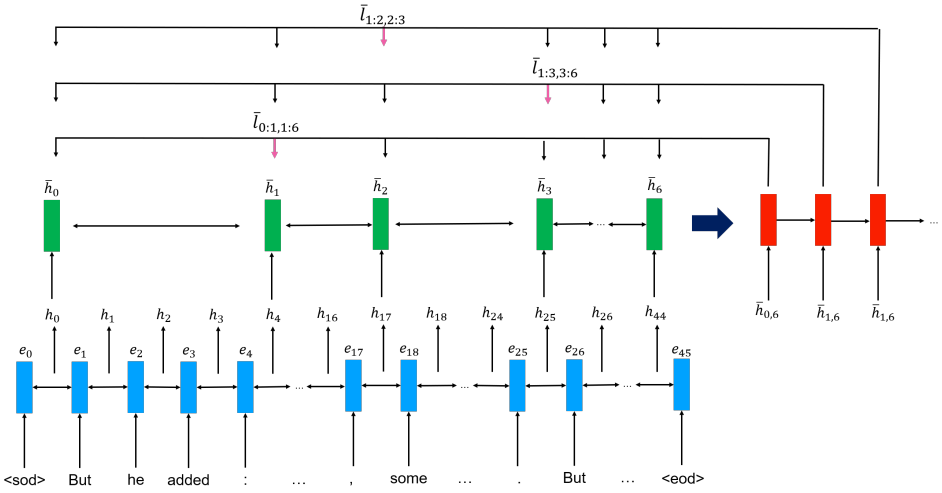


Figure 7: Our discourse parser along with the first few decoding steps. When EDUs are given, we use a hierarchical EDU encoding (Boundary LSTM) to model EDU boundary representations.

7 Error Analysis

We show our best parser's (with gold EDUs) confusion matrix for all relation labels in Fig. 5. Fig. 9 - 11 present examples where our parser falsely labels a Condition as Background, Temporal as Joint and Explanation as Elaboration.

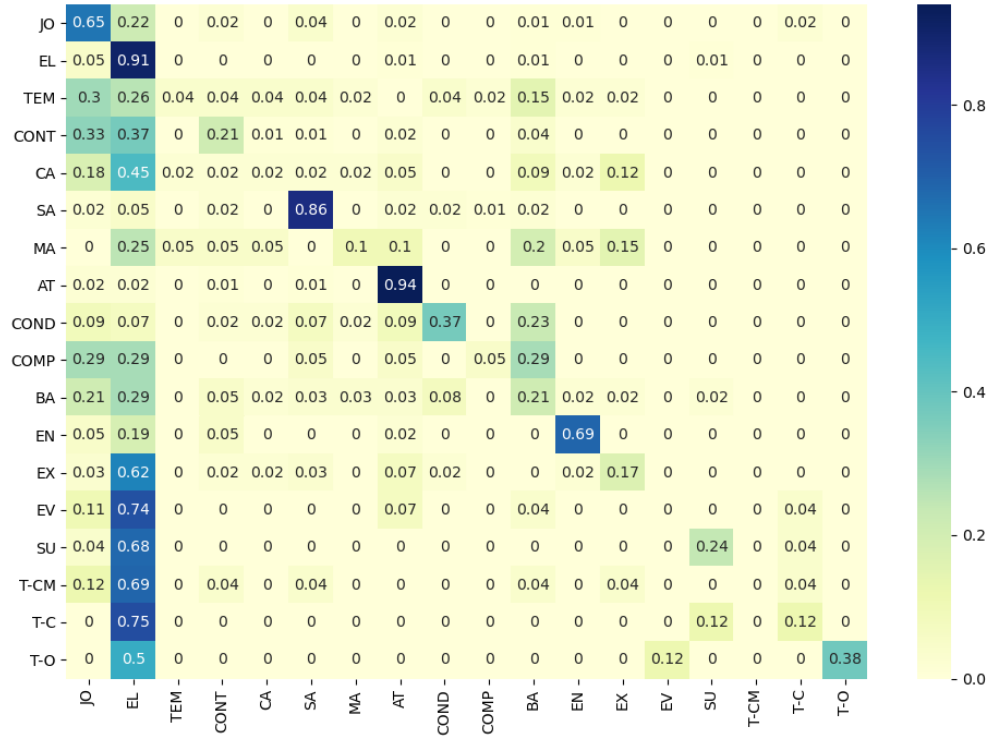


Figure 8: Confusion matrix for relation labels on the RST–DT test set. The vertical axis represents true and horizontal axis represents predicted relations. The relations are: Joint (JO), Elaboration (EL), Temporal (TEM), Contrast (CONT), Cause (CA), Same-Unit (SA), Manner-Means (MA), Attribution (AT), Condition (COND), Comparison (COMP), Background (BA), Enablement (EN), Explanation (EX), Evaluation (EV), Summary (SU), Topic-Comment (T-CM), Topic-Change (T-C) and TextualOrganization (T-O).

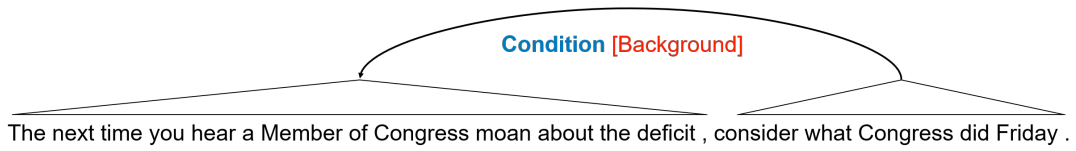


Figure 9: Our system incorrectly labels a Condition as Background.

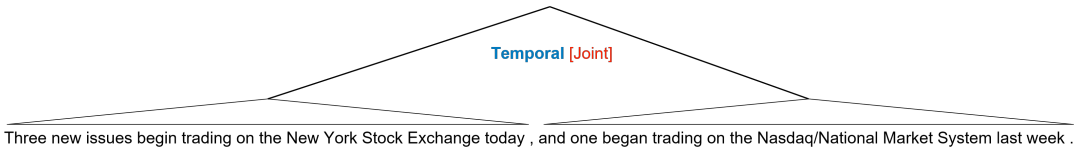


Figure 10: Our system incorrectly labels a Temporal as Joint.

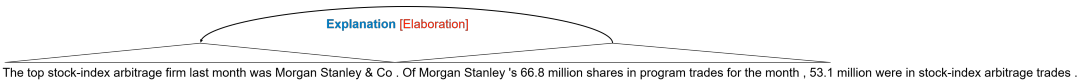


Figure 11: Our system incorrectly labels a Explanation as Elaboration.