Can we obtain significant success in RST discourse parsing by using Large Language Models?

Aru Maekawa[1], **Tsutomu Hirao**[2], **Hidetaka Kamigaito**[1], **Manabu Okumura**[1]
[1]Institute of Innovative Research, Tokyo Institute of Technology,
[2]NTT Communication Science Laboratories, NTT Corporation
{maekawa@lr., kamigaito@lr., oku@}pi.titech.ac.jp
tsutomu.hirao@ntt.com

Abstract

Recently, decoder-only pre-trained large language models (LLMs), with several tens of billion parameters, have significantly impacted a wide range of natural language processing (NLP) tasks. While encoder-only or encoder-decoder pre-trained language models have already proved to be effective in discourse parsing, the extent to which LLMs can perform this task remains an open research question. Therefore, this paper explores how beneficial such LLMs are for Rhetorical Structure Theory (RST) discourse parsing. Here, the parsing process for both fundamental top-down and bottom-up strategies is converted into prompts, which LLMs can work with. We employ Llama 2 and fine-tune it with QLoRA, which has fewer parameters that can be tuned. Experimental results on three benchmark datasets, RST-DT, Instr-DT, and the GUM corpus, demonstrate that Llama 2 with 70 billion parameters in the bottom-up strategy obtained state-of-the-art (SOTA) results with significant differences. Furthermore, our parsers demonstrated generalizability when evaluated on RST-DT, showing that, in spite of being trained with the GUM corpus, it obtained similar performances to those of existing parsers trained with RST-DT.

1 Introduction

Rhetorical Structure Theory (RST) (Mann and Thompson, 1987) is one of the influential discourse theories used to explain the coherence of texts. It plays an important role in various natural language processing (NLP) tasks at the document level, including sentiment analysis (Bhatia et al., 2015), automatic summarization (Marcu, 1998; Xu et al., 2020; Kwon et al., 2021), question answering (Gao et al., 2020), machine translation (Chen et al., 2020; Tan et al., 2022), and MT evaluation (Joty et al., 2017). According to RST, a text is represented as a binarized constituent tree (RST-tree), whose terminal nodes correspond to elementary discourse units (EDUs), clause-like units, and non-terminal nodes indicate the nuclearity status, i.e., either Nucleus or Satellite, of text spans consisting of single or contiguous EDUs. The edges represent the rhetorical relation between two adjacent text spans dominated by non-terminal nodes. Figure 1 shows an example of the RST tree obtained from RST Discourse Treebank (RST-DT) (Lynn Carison, 2002). In the figure, the nucleurity status of the text span consisting of e_1 and e_2 is the nucleus, and it is modified by the satellite, e_3. A mono-nuclear relation, Attribution, is given to the two spans.

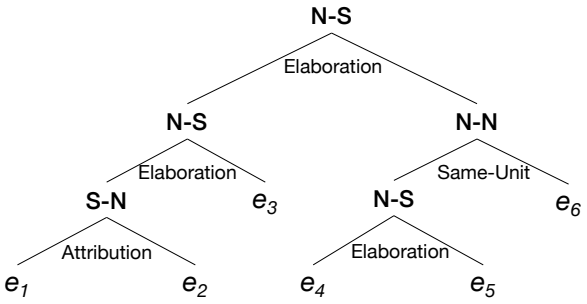

Figure 1: Example of RST tree from WSJ_1100 in RST-DT (Lynn Carison, 2002), consisting of six EDUs (e): e_1: [Westinghouse Electric Corp. said], e_2: [it will buy Shaw-Walker Co.], e_3: [Terms weren't disclosed.], e_4: [Shaw-Walker,], e_5: [based in Muskegon, Mich.,], e_6: [makes metal files and desks, and seating and office systems furniture.]. N and S represent the Nucleus and Satellite, respectively.

Since the late 2010s, neural RST discourse parsing methods that use encoder-only pre-trained language models (PLMs) to encode text spans into vectors have been proposed due to advances in neural models. While earlier models, e.g., Yu et al. (2018); Lin et al. (2019); Kobayashi et al. (2020), obtained vector representations for text spans from a static PLM, such as GloVe (Pennington et al., 2014), recent models, e.g., Guz and Carenini (2020); Shi et al. (2020); Nguyen et al. (2021); Zhang et al. (2021a), obtained them from a transformer-based

Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics
Volume 1: Long Papers, pages 2803–2815

PLM, such as XLNet (Yang et al., 2019). To form RST trees, they obtained vectors via PLMs and exploited them to determine the parsing actions in a top-down or bottom-up strategy. More recently, there is also a parser (Hu and Wan, 2023) that utilizes an encoder-decoder PLM to transform input text into a linearized RST tree.

There was a shift in focus from encoder-only to massive-scale decoder-only PLMs. Some large language models (LLMs), such as GPT-3 (Brown et al., 2020) and Llama 2 (Touvron et al., 2023), have several tens of billions of parameters and are pre-trained with only a decoder. These have significantly impacted NLP, similar to encoder-only and encoder-decoder PLMs. LLMs have demonstrated remarkable success in various NLP tasks due to their large numbers of parameters and ease of availability. Their impact extends beyond generation tasks and includes classification tasks (Brown et al., 2020; Wei et al., 2022; Wu et al., 2023). Therefore, they could also be advantageous in RST discourse parsing. Furthermore, we are strongly motivated to adopt LLMs because previous discourse parsing methods have been greatly improved by using encoder-only or encoder-decoder pre-trained language models.

In this paper, we explore the potential of using LLMs for RST discourse parsing. As a first step in exploiting LLMs, our approach is to translate the parsing steps of both fundamental top-down and bottom-up strategies into prompts. Then, we fine-tune Llama 2 using QLoRA (Dettmers et al., 2023), an extension of LoRA (Hu et al., 2022), which is an adapter that injects trainable low-rank matrices into each layer of the Transformer, while it freezes the weights of the pre-trained model for efficient computing. The experimental results from RST-DT, Instructional Discourse Treebank (Instr-DT) (Subba and Di Eugenio, 2009), and the GUM corpus (Zeldes, 2017) demonstrate that our parser with the bottom-up parsing strategy surpassed the current state-of-the-art (SOTA) results. It outperformed the current SOTA models by around 2-3 points on RST-DT, by 0.4-3.7 points on Instr-DT, and by 1.5-6 points on the GUM corpus. Furthermore, out-of-domain evaluations using RST-DT and the GUM corpus demonstrate the potential generalizability of our parsers. Our parsers, trained with the GUM corpus, achieved smaller degradation when evaluated on RST-DT. The performances are close to those of the existing parsers trained with RST-DT itself. These findings provide valuable insights into the future direction of RST discourse parsing. We will release our code at https://github.com/nttcslab-nlp/RSTParser_EACL24.

2 Related Work

2.1 RST Discourse Parsing with Encoder-only PLMs

Most neural RST discourse parsers have two fundamental components: a feature extraction layer to obtain vector representations for text spans and a classification layer to form RST trees. The feature extraction layer receives tokens in the text spans as an input and obtains their vector representations through a PLM. The classification layer, located on top of the feature extractor, makes decisions that guide the form of RST trees by the parsers.

Yu et al. (2018) proposed a bottom-up parsing model with a feature extractor based on GloVe and syntactic features. The parser merges text spans using shift-reduce operations to build RST trees based on Feed-Forward Networks (FFNs). To extend the parser, they incorporated a BERT-based tailored PLM with objectives that include the prediction of the next EDU and a discourse marker (Yu et al., 2022). By enhancing the PLM, the performance was greatly improved: They achieved a fully-labeled span F1 score of 53.8. Guz and Carenini (2020) extended Wang et al.'s classical shift-reduce parser (Wang et al., 2017), replacing SVMs with FFNs and a feature extractor with SpanBERT (Joshi et al., 2020). The gain of F1 scores against Wang et al.'s parser was around 3 points due to having more sophisticated contextual word embeddings.

Kobayashi et al. (2020) proposed a top-down parsing model based on a minimal span-based approach, that recursively splits a span into smaller ones by exploiting a classification layer with FFNs. Their feature extractor was a combination of Glove and ELMo (Peters et al., 2018). Another top-down parsing model was proposed using a decoder instead of FFNs for a classification layer. Lin et al. (2019) proposed top-down depth-first parsing at the sentence-level based on a pointer-generator network. The parser employs GloVe in the feature extractor, and then the decoder recursively generates a split for an input span. Shi et al. (2020) introduced layer-wise beam search and used XLNet (Yang et al., 2019) in the feature extractor to extend the top-down model to the document level, achieving SOTA results at that time. Nguyen et al.

(2021) and Zhang et al. (2021a) also reported that XLNet is beneficial for enhancing performance in a similar top-down approach.

Recently, Kobayashi et al. (2022) explored simple and strong baselines based on Guz and Carenini's bottom-up parser (2020) and Kobayashi et al.'s top-down parser (2020) with varying encoder-only PLMs. The results suggest that the success of the parsing method heavily relies on the PLMs rather than on the parsing strategies themselves. The current best score, a fully-labeled span score of 55.4, was obtained by the bottom-up parser combined with DeBERTa (He et al., 2021), a SOTA encoder-only PLM.

As another approach, Braud et al. (2016) proposed RST discourse parsing as a text-to-text generation task.[1] They used an LSTM-based encoder-decoder to receive a text as an input and output an S-expression that expresses a path from a root node to leaf nodes of an RST tree. They adopted an earlier PLM, PolyGlot (Al-Rfou' et al., 2013). Zhang et al. (2021b) proposed sentence-level parsing by re-ranking linearized parse trees obtained from an external parser, based on MPNet (Song et al., 2020).

2.2 RST Discourse Parsing with Encoder-decoder PLMs

As an extension of Braud et al.'s approach (2016), Hu and Wan (2023) proposed a more straightforward model as a text-to-text generation task, that exploits a SOTA encoder-decoder PLM, T5 (Raffel et al., 2020). It directly learns the transformation from an input text into the linearized S-expression of the entire RST tree by benefiting from a powerful encoder-decoder PLM. The parser performed better than Nguyen et al. (2021).

3 Proposed Approach

Three approaches can be possible when using decoder-only LLMs for RST discourse parsing. The first is to create linearized S-expressions for RST trees using LLMs, which is similar to how encoder-decoder PLMs have been used. The second is to replace encoder-only PLMs with an encoder of LLMs in conventional top-down or bottom-up parsing. The third method involves using LLMs to imitate the parsing process, which involves combining feature extraction and classification layers, similar to how encoder-only PLMs have been used. The first approach can be challenging, especially when dealing with lengthy documents. This is because the number of output tokens increases disproportionately with the number of input tokens. Furthermore, additional techniques, such as constrained beam search, are required to obtain valid linearization forming a tree. The second approach is not promising because it does not perform as well as encoder-only PLMs (Devlin et al., 2019). Thus, we adopt the third approach in this work.

Before describing our own approach, we first illustrate fundamental top-down and bottom-up parsing methodologies using encoder-only PLMs.

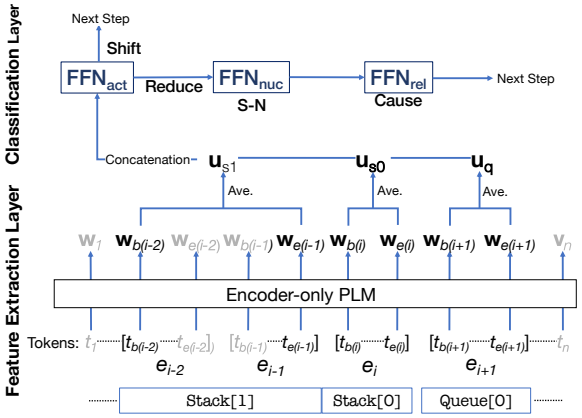

Figure 2: Bottom-up parsing by shift-reduce operations

3.1 Bottom-up Parsing

Figure 2 gives an overview of bottom-up parsing based on shift-reduce algorithms. The text's tokens are first converted into word embeddings using an encoder-only PLM. Then, a vector representation for a text span is obtained by averaging the word embeddings for the leftmost token in the first EDU and the rightmost token in the final EDU.

In the figure, FFNs in the classification layer handles shit-reduce operations based on a stack and a queue; a stack stores subtrees, i.e., text spans that have already been parsed, and a queue contains incoming EDUs. The parser builds an RST tree by merging two adjacent text spans while selecting one of the following actions:

Shift: Pop the first EDU off the queue and push it onto the stack.

Reduce: Pop two text spans from the stack and merge them into a new span, then push it onto

[1]While this approach may seem appropriate to the next section, since the studies exclusively used only encoder-only PLMs, we categorize them here.

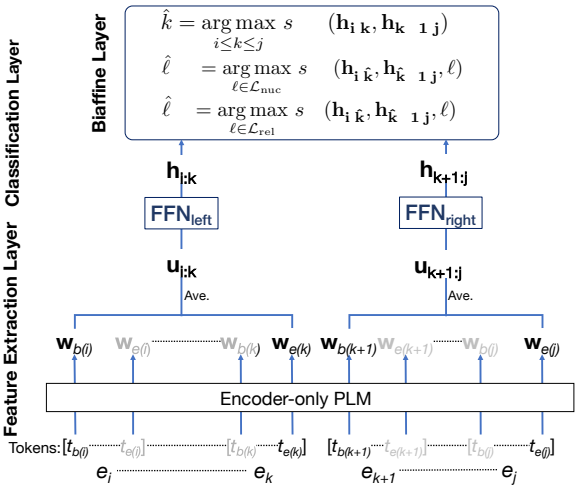

Figure 3: Top-down parsing by span split

the stack.

Note that the nuclearity status and rhetorical relation labels are independently predicted by different classifiers when the Reduce operation is selected. FFN_{act}, FFN_{nuc}, and FFN_{rel} in the classification layer are feed-forward networks for predicting the action, nuclearity, and relation labels, respectively. FFN_{act} solves a binary classification problem (Shift or Reduce), FFN_{nuc} solves a three-class classification problem (either nucleus-nucleus, nucleus-satellite, or satellite-nucleus), and FFN_{rel} solves a multi-class classification problem (the number of classes derives from the number of rhetorical relations used in the dataset): $s_* = \text{FFN}_*(\text{Concat}(\mathbf{u}_{\text{s}_0}, \mathbf{u}_{\text{s}_1}, \mathbf{u}_{\text{q}_0}))$, where the function "Concat" concatenates the vectors received as the arguments. $\mathbf{u}_{\text{s}_0}$ is the vector representation of a text span stored in the first position of the stack, $\mathbf{u}_{\text{s}_1}$ is that in the second position, and $\mathbf{u}_{\text{q}_0}$ is that in the first position of the queue.

3.2 Top-down Parsing

An overview of top-down parsing is presented in Figure 3. We obtain two vector representations of text spans, $\mathbf{u}_{i:k}$ and $\mathbf{u}_{k+1:j}$, for each possible split point k of the span between the i-th and j-th EDUs, using the same approach as in the bottom-up parsing. Then, the classification layer consisting of FFNs and the biaffine layer identify the best-split point based on a scoring function, $s_{\text{split}}(\mathbf{h}_{i:k}, \mathbf{h}_{k+1:j})$, which is defined as

$$s_{\text{split}}(\mathbf{h}_{i:k}, \mathbf{h}_{k+1:j}) = \mathbf{h}_{i:k}\mathbf{W}\mathbf{h}_{k+1:j} + \mathbf{v}_{\text{left}}\mathbf{h}_{i:k} + \mathbf{v}_{\text{right}}\mathbf{h}_{k+1:j}, \quad (1)$$

where $\mathbf{W}$ is a weight matrix and $\mathbf{v}_{\text{left}}$ and $\mathbf{v}_{\text{right}}$ are weight vectors corresponding to the left and right spans, respectively. Here, $\mathbf{h}_{i:k}$ and $\mathbf{h}_{k+1:j}$ are obtained via FFNs as follows:

$$\mathbf{h}_{i:k} = \text{FFN}_{\text{left}}(\mathbf{u}_{i:k}), \quad (2)$$

$$\mathbf{h}_{k+1:j} = \text{FFN}_{\text{right}}(\mathbf{u}_{k+1:j}). \quad (3)$$

Then, the span is split at the position k that maximizes Eq. (1).

For a pair of text spans divided at point $\hat{k}$, we assign either nucleus-nucleus, nucleus-satellite, or satellite-nucleus and a rhetorical relation from a pre-defined set using the following scoring function:

$$s_{\text{label}}(\mathbf{h}_{i:\hat{k}}, \mathbf{h}_{\hat{k}+1:j}, \ell) = \mathbf{h}_{i:\hat{k}}\mathbf{W}^{\ell}\mathbf{h}_{\hat{k}+1:j} + \mathbf{v}^{\ell}_{\text{left}}\mathbf{h}_{i:\hat{k}} + \mathbf{v}^{\ell}_{\text{right}}\mathbf{h}_{\hat{k}+1:j}, \quad (4)$$

where $\mathbf{W}^{\ell}$ is a weight matrix for a specific nuclearity or relation label ℓ and $\mathbf{v}^{\ell}_{\text{left}}$ and $\mathbf{v}^{\ell}_{\text{right}}$ are weight vectors corresponding to the left and right spans for the label ℓ, respectively.

3.3 Prompts for Bottom-up Parsing

To parse a document in a bottom-up manner with LLMs, we translate the shift-reduce operations, described in §3.1, into prompts. In this case, we emulate the parsing process with a stack and a queue while using the following prompt template $\mathbf{x}_{act}$ to predict an action $\mathbf{y}_{act}$:

> **`Stack2`**: *Text span(s) in the second position of the stack.*
> **`Stack1`**: *Text span(s) in the first position of the stack.*
> **`Queue1`**: *An EDU in the first position of the queue.*
> **`Action (shift or reduce)`**: *Either Shift or Reduce.*

An LLM determines whether to Shift or Reduce based on the text spans in Stack1, Stack2, and Queue1 at each step with the above prompts. Then, we assign nuclearity and rhetorical relation labels between two text spans in Stack1 and Stack2 when the Reduce action is selected. We use the following prompt template $\mathbf{x}_{nuc}$ to predict a nuclearity label $\mathbf{y}_{nuc}$:

> **`Span2:`** *Text span(s) in the second position of the stack.*

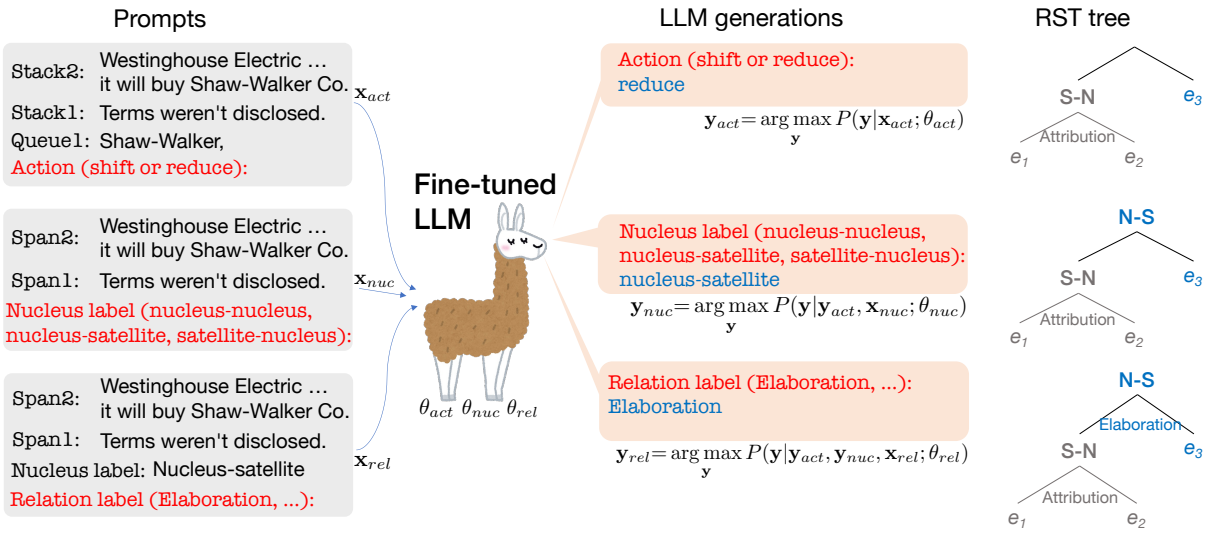

Figure 4: Example of the bottom-up parsing process using an LLM with prompts. In the example, `Stack2` stores a text span, an already constructed subtree, consisting of two EDUs: e_1: [Westinghouse Electric Corp. said], e_2: [it will buy Shaw-Walker Co.]. `Stack1` stores a text span of single EDU, e_3: [Terms weren't disclosed.]. `Queue1` stores an EDU, e_4: [Shaw-Walker,]. After this step, the parsing process goes on to the next steps while updating Stack* and Queue.

> **`Span1`:** *Text span(s) in the first position of the stack.*
> **`Nucleus label (nucleus-nucleus, nucleus-satellite, satellite-nucleus)`:** *Either one of them.*

Here, Span1 and Span2 are text spans in Stack1 and Stack2, respectively. To predict a rhetorical relation label $\mathbf{y}_{rel}$, we use the prompt template $\mathbf{x}_{rel}$ by replacing the third prompt of $\mathbf{x}_{nuc}$ with the following two new prompts:

> **`Nucleus label`**: *predicted nuclearity label.*
> **`Relation label (Rel`$_1$**, **`Rel`$_2$**, ..., **`Rel`$_n$`)`**: *Either one of them.*

Here, Rel_i indicates the i-th rhetorical relation, and the set of rhetorical relations is different for each dataset. We construct an RST tree based on LLM's decisions using the above prompts for each parsing step. Figure 4 shows an example of our bottom-up parsing with prompts.

LLMs infer outputs by choosing the sequence with the maximum probability as follows:

$$\mathbf{y}_{act} = \arg\max_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}_{act}; \theta_{act}), \quad (5)$$

$$\mathbf{y}_{nuc} = \arg\max_{\mathbf{y}} P(\mathbf{y}|\mathbf{y}_{act}, \mathbf{x}_{nuc}; \theta_{nuc}), \quad (6)$$

$$\mathbf{y}_{rel} = \arg\max_{\mathbf{y}} P(\mathbf{y}|\mathbf{y}_{act}, \mathbf{y}_{nuc}, \mathbf{x}_{rel}; \theta_{rel}), \quad (7)$$

where θ_{act}, θ_{nuc}, and θ_{rel} are weight parameters of LLMs for predicting action, nuclearity, and rhetorical relation labels, respectively.

3.4 Prompts for Top-down Parsing

To perform top-down parsing with LLMs, we translate the span split procedure, described in §3.2, into the following prompt template $\mathbf{x}_{span}$ for predicting a split span $\mathbf{y}_{span}$:

> **`Input:`** *A text span, a sequence of EDUs, to be split.*
> **`Split point`** $(0 — j{-}i{-}1)$: *An index of EDU* $(0 \leq k \leq j{-}i{-}1)$.

Note that we adjust the index of EDUs in the given text span, consisting of the i-th EDU to the j-th EDU, so that the index of the first EDU in the prompt is always 0. We give the above prompts recursively to an LLM in order to identify a split point for a span obtained during parsing.

We use the same prompts as in bottom-up parsing to assign nuclearity and rhetorical relation labels. Figure 5 shows an example.

Similar to the bottom-up parsing, inference is performed as follows:

$$\mathbf{y}_{span} = \arg\max_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}_{span}; \theta_{span}), \quad (8)$$

$$\mathbf{y}_{nuc} = \arg\max_{\mathbf{y}} P(\mathbf{y}|\mathbf{y}_{span}, \mathbf{x}_{nuc}; \theta_{nuc}), \quad (9)$$

$$\mathbf{y}_{rel} = \arg\max_{\mathbf{y}} P(\mathbf{y}|\mathbf{y}_{span}, \mathbf{y}_{nuc}, \mathbf{x}_{rel}; \theta_{rel}), \quad (10)$$

where θ_{span} denotes weight parameters for the LLM to split spans.

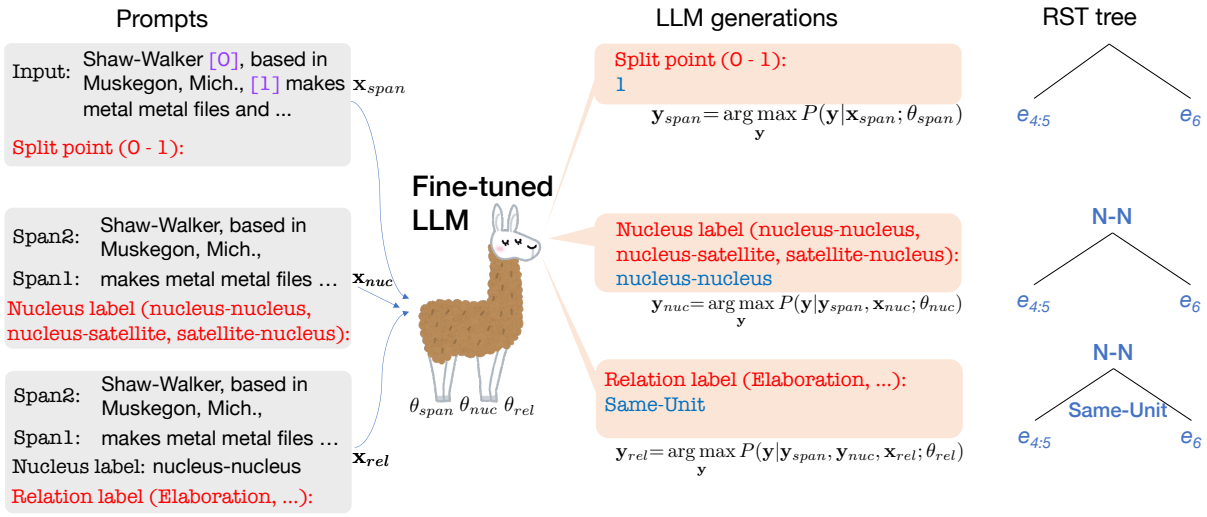

Figure 5: Example of the top-down parsing process using an LLM with prompts. In the example, a text span consisting of three EDUs, e_4: [Shaw-Walker,], e_5: [based in Muskegon, Mich.,], and e_6: [makes metal files and desks, and seating and office systems furniture.], are divided and labeled by an LLM's decision. The process is recursively applied until divided spans are identical to single EDUs.

3.5 Handling Erroneous Generation

Either of our parsing approaches might generate labels not in a pre-defined set for the classification. Such labels prevent the construction of valid RST trees. Accordingly, we introduce default rules to correct such invalid generation. When invalid generation occurs, one of the following rules is applied and the default label is used in place of the generated one: Shift is the default action for bottom-up parsing, 0 is the default split point for top-down parsing. Nucleus-satellite and Elaboration are the default nuclearity and rhetorical relation labels, respectively, for both parsing strategies. These are selected because of the majority labels in our datasets.

4 Experimental Settings

4.1 Large Language Models

We utilized Llama 2 (Touvron et al., 2023),[2] one of the largest publicly available open-source decoder-only PLMs, for our LLM-based RST discourse parsing models. Llama 2 can handle up to 70 billion parameters, which have been trained using a trillion tokens of text data. Although the technical report did not provide information on the specifics of the training dataset, it was made clear that the dataset comprises publicly available sources. Hence, we are confident that the datasets used for training and testing our parsers are not included in Llama 2.

Since we have found that zero-shot and few-shot approaches do not produce satisfactory results,[3] we instead opted to fine-tune Llama 2 with the prompts and the correct outputs. However, the large GPU memory requirements and computational costs made it impractical to fully fine-tune Llama 2. As a result, we turned to QLoRA (Dettmers et al., 2023)[4] to update θ_{act}, θ_{span}, θ_{nuc}, and θ_{rel}. QLoRA is a quantized version of LoRA (Hu et al., 2022), which introduces trainable low-rank matrices into each layer of the Transformer architecture without altering the weights for the parameters in LLMs.

4.2 Datasets

LLMs are pre-trained on large open-domain datasets, allowing our parsers to easily adapt to specific domains through fine-tuning. To demonstrate this capability and make a fair comparison with Kobayashi et al.'s bottom-up and top-down parsers, we used two benchmark datasets from different domains, RST-DT, Instr-DT, and GUM Corpus.

RST-DT contains 385 documents selected from the Wall Street Journal. It is officially divided into

[2] https://huggingface.co/meta-llama/Llama-2-{7,13,70}b-hf

[3] A zero-shot approach resulted in only a 9.27 Span F-score, indicating no further consideration. This finding is expected because LLM pre-training did not cover parsing actions like shift or reduce. Since LLMs lack this inherent knowledge due to their pre-training, fine-tuning them emerges as the most viable approach.

[4] https://github.com/artidoro/qlora

		RST-DT				Instr-DT				GUM			
		Span	**Nuc.**	**Rel.**	**Full**	**Span**	**Nuc.**	**Rel.**	**Full**	**Span**	**Nuc.**	**Rel.**	**Full**
Top-down	Liu et al.	76.5	65.2	54.2	–	–	–	–	–	68.6	54.9	–	–
	Yu et al.	72.9	62.7	52.5	50.5	–	–	–	–	–	–	–	–
	Kobayashi et al.	78.5	67.9	56.6	54.4	77.3	57.9	50.0	43.4	74.4	62.2	50.9	48.7
	Llama 2 (7B)	76.3	65.4	55.2	53.4	75.7	56.2	49.8	43.6	72.8	60.9	52.1	50.9
	Llama 2 (13B)	78.6	67.9	57.7	55.6	75.7	57.3	50.2	43.6	74.9	62.5	53.8	52.5
	Llama 2 (70B)	78.8	68.7	57.7	56.0	76.2	57.1	53.1	45.2	75.8	64.0	55.8	54.8
Bottom-up	Guz et al.	76.5	65.9	54.8	–	–	–	–	–	69.9	57.0	–	–
	Yu et al.	76.4	66.1	54.5	53.5	–	–	–	–	–	–	–	–
	Kobayashi et al.	77.8	68.0	57.3	55.4	77.8	60.0	51.4	44.4	73.4	60.9	50.3	48.5
	Llama 2 (7B)	78.2	67.5	57.6	55.8	76.7	58.2	48.5	43.5	74.4	63.0	53.4	52.1
	Llama 2 (13B)	78.3	68.1	57.8	56.0	77.4	60.4	52.1	46.1	74.8	63.4	54.0	52.8
	Llama 2 (70B)	**79.8**	**70.4**	**60.0**	**58.1**	**79.1**	**60.4**	**55.1**	**47.3**	**76.4**	**64.7**	**56.4**	**55.2**

Table 1: Results on RST-DT, Instr-DT, and the GUM Corpus with Standard-Parseval. Liu et al.'s top-down parser (Liu et al., 2021) employed XLM-RoBERTa-base with 125M parameters, Guz et al's bottom-up parser (Guz and Carenini, 2020) employed SpanBERT-base with 110M parameters, Yu et al.'s parsers (Yu et al., 2022) employed XLNet-base with 110M parameters, and Kobayahi et al.'s parsers employed DeBERTa-base with 140M parameters. We omit the reported Relation scores for Liu et al.'s and Guz et al.'s parsers on the GUM corpus because they are the results based on GUM's own relation label set.

347 documents as the training set and 38 as the test dataset. We used 18 coarse rhetorical relations derived from 78 fine-grained ones. We used 40 documents in the training dataset as the development dataset, following Kobayashi et al. (2022).

Instr-DT contains 176 documents obtained from the home-repair instruction manuals. The number of rhetorical relations in the dataset is 39. We followed Kobayashi et al.'s setting (2022), i.e., 126, 25, and 25 documents were used for the training, development, and test datasets, respectively.

The GUM corpus contains 213 documents in total for 12 genres, e.g., News, Speech, Reddit, and Vlog. We used officially divided 165, 24, and 24 documents for training, development, and test datasets. In this experiment, we translated rhetorical relation labels in the GUM corpus to match them with those in RST-DT by using a label correspondence described in (Liu and Zeldes, 2023).

We used gold EDU segmentation for both datasets by following conventional studies.

4.3 Evaluation Metrics

We evaluated the results with micro-averaged F_1 scores of unlabeled, nuclearity-, relation-, and fully-labeled span, based on Standard-Parseval (Morey et al., 2017), the standard evaluation metrics for RST discourse parsing. Note that during both the training and test phases, RST-trees were converted into right-heavy binary trees (Sagae and Lavie, 2005).

4.4 Configurations

Our implementations are based on the official implementation of QLoRA,[2] which is based on Huggingface Transformers (Wolf et al., 2020). We employed Adam (Kingma and Ba, 2015) to optimize the parameters. We used QLoRA with 4-bit quantization, setting `lora_r` to 64, `lora_alpha` to 16, and `lora_dropout` to 0.1. A learning rate of 2e-4 was used at a batch size of 16. We scheduled the learning rate by linear warm-up, which increases it linearly during the first 3% of training steps and then decreases it with cosine annealing to 0 until the final epoch. We trained the model with a different LoRA adapter for each subtask, i.e., span, nuclearity, and relation labeling, for 5 epochs and chose the best checkpoint by evaluating the performance on the development dataset. We provide a summary of all hyperparameter settings in Appendix A.

5 Results and Discussion

Overall Performance: Table 1 shows the results, where the scores of Kobayashi et al.'s and Yu et al.'s parsers are borrowed from their papers (Kobayashi et al., 2022; Yu et al., 2022) and those of Guz et al's and Liu et al's parsers are borrowed from (Liu and Zeldes, 2023). We show the results of our parsers with Llama 2 for 7B, 13B, and 70B parameters.

When focusing on the number of the parameters in Llama 2, the largest number naturally yields the best results. In particular, 70B parameters obtained

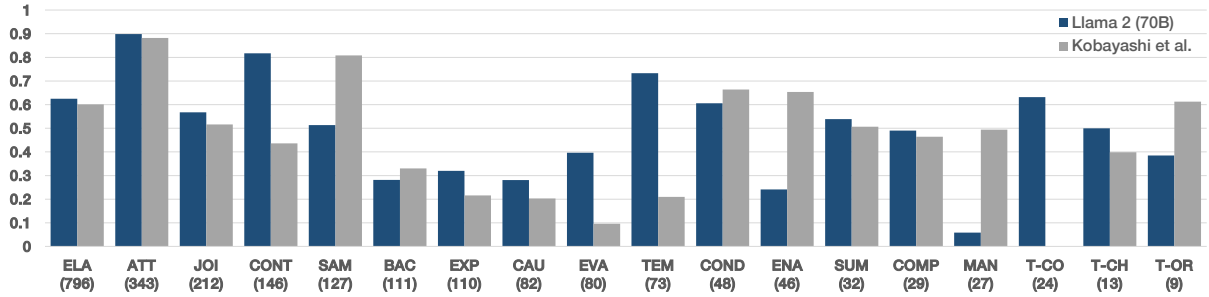

Figure 6: F_1 scores of Llama 2 (70B) and Kobayashi et al. for each relation label in bottom-up parsing: ELABORATION, ATTRIBUTION, JOINT, CONTRAST, SAME-UNIT, EXPLANATION, CAUSE, EVALUATION, TEMPORAL, CONDITION, ENABLEMENT, SUMMARY, COMPARISON, MANNER-MEANS, TOPIC-COMMENT, TOPIC-CHANGE, and TEXTUAL-ORGANIZATION. Numbers in parenthesis are the frequency of the label.

the current best scores for both datasets; however, the performance of the parsers with 7B and 13B parameters are still comparable to Kobayashi et al.'s parsers. The gains by bottom-up parsing with 70B are impressive, surpassing Kobayashi et al.'s parser by approximately 3 points in RST-DT and Instr-DT, and by around 7 points in the GUM corpus.

Bottom-up parsing consistently outperforms top-down parsing by 1 to 2 points when comparing the parsing strategies in our parsers. Since both strategies used the same prompts for nuclearity and rhetorical relation labeling, the differences are considered to come from different prompts to build the skeleton of the RST tree. The prompts for bottom-up parsing mention three text spans in `Stack1`, `Stack2`, and `Queue1`, while those for top-down parsing mention only one text span in `Input`. In other words, the former handles richer information than the latter.

Performance of Relation Labeling: To investigate the effectiveness of our parsers in more detail, we show Relation F_1 scores for each relation label in RST-DT with Llama 2 (70B) and Kobayashi et al.'s bottom-up parsers in Figure 6. In most cases, Llama 2 (70B) outperforms Kobayashi et al., even for less frequent relation labels, indicating Llama 2 (70B) has greater potential for generalization.

These results indicate the effectiveness of LLMs for RST discourse parsing. The findings are interesting in that simple pre-trained language models consisting of only a transformer decoder can be easily tailored for determining parsing actions by fine-tuning with prompts. Our parsers perform well on all datasets, which are from different domains. The results demonstrate the advantage of LLM-based RST discourse parsing in domain portability, particularly in achieving the best scores on Instr-DT with less training data.

Cross-corpus Generalization: To examine the generalizability of our parsers in detail, we evaluated them with out-of-domain evaluations using RST-DT and the GUM corpus. Tables 2 and 3 show the results of training parsers on one dataset and evaluating them on the other.

Comparing the results with Table 1, the performances are lower. In particular, the parsers trained with RST-DT degraded more when tested on the GUM corpus than the opposite, the parsers trained with the GUM corpus and tested on RST-DT. The findings suggest that using a single genre dataset for generalization across multiple genres is challenging. This aligns with the observation made by Liu and Zeldes (2023).

In Table 2, when comparing our parsers with Liu et al.'s and Kobayashi et al.'s parsers, our parsers obtained better scores than them in most cases. In particular, Llama 2 (70B) with the bottom-up strategy achieved the best scores. It obtained around 2-point gains against Kobayashi et al.'s parser in all metrics. Notably, it further outperformed both Liu et al.'s and Guz et al.'s parsers trained with the GUM corpus in Table 1 on Span while it obtained a slightly lower score in Nuc.

On the other hand, performance degradation in Table 3 is much lower than that in Table 2. Our parsers obtained remarkable gains against Liu et al.'s and Kobayashi et al.'s parsers. The gains are emphasized in Rel. and Full. Despite being trained with an out-of-domain dataset, our parsers in Table 2 achieved Span F_1 scores that are comparable to those in Table 1. Furthermore, the performance of our parsers in Full degraded by 12%, while that of Kobayashi et al.'s parser degraded by 15-20%.

The successful improvements were achieved by using a large number of LLM parameters and training them with a massive amount of text, compared

		Span	Nuc.	Rel.	Full
Top-down	Liu et al.	66.2	50.8	–	–
	Kobayashi et al.	70.3	53.7	41.1	38.3
	Llama 2 (7B)	68.2	51.8	39.8	37.6
	Llama 2 (13B)	69.0	51.5	39.2	37.3
	Llama 2 (70B)	71.0	53.3	42.1	39.8
Bottom-up	Guz et al.	65.3	49.5	–	–
	Kobayashi et al.	68.3	52.6	40.8	38.0
	Llama 2 (7B)	69.4	53.0	40.2	38.2
	Llama 2 (13B)	69.3	52.1	39.8	37.8
	Llama 2 (70B)	72.6	55.6	43.0	40.5

Table 2: Cross-corpus generalization results on the GUM corpus. Parsers were trained using RST-DT and their performance was evaluated on the GUM corpus.

		Span	Nuc.	Rel.	Full
Top-down	Liu et al.	72.7	57.4	–	–
	Kobayashi et al.	76.1	61.8	49.3	46.5
	Llama 2 (7B)	75.3	61.7	49.9	48.0
	Llama 2 (13B)	76.3	63.4	51.3	49.4
	Llama 2 (70B)	78.2	64.4	51.5	49.7
Bottom-up	Guz et al.	71.1	55.9	–	–
	Kobayashi et al.	72.0	58.5	46.3	44.3
	Llama 2 (7B)	77.4	63.6	51.3	49.0
	Llama 2 (13B)	77.4	64.5	52.2	50.3
	Llama 2 (70B)	79.7	66.5	53.2	51.1

Table 3: Cross-corpus generalization results on RST-DT. Parsers were trained using the GUM corpus and their performance was evaluated on RST-DT.

to the encoder-only models. Although we could potentially improve the performance of the encoder-only PLMs by increasing their parameters to the level of the current LLMs, this task poses a significant challenge. Furthermore, considering the focus of the research has shifted from encoder-only PLMs to LLMs, our findings are highly valuable for future research in RST discourse parsing.

6 Conclusion

This paper explored the potential of using Llama 2, the largest publicly available decoder-only language model, pre-trained with a trillion tokens of text data, for RST discourse parsing. To exploit Llama 2, we translated fundamental bottom-up and top-down parsing processes into prompts. Then, we fine-tuned Llama 2 with them using QLoRA for efficient computing. The experimental results obtained from three datasets, RST-DT, Instr-DT, and the GUM corpus, which come from different domains, demonstrated the effectiveness of our parsers with Llama 2, including their domain portability. Specifically, our approach with Llama 2 (70B) obtained better results than the current SOTA parser on all datasets. Furthermore, findings from the experimental results for cross-corpus generalization showed the significant promise of our approach, that is, that our parsers, in spite of being trained with the GUM corpus, obtained comparable performance to parsers trained with RST-DT itself in Span F_1 scores, keeping small degradation in Nuc., Rel., and Full F_1 scores.

Since this work is just a first step in exploiting LLMs for RST discourse parsing, we can work in this direction with various topics to further improve the LLM-based model; e.g., incorporating richer information to improve the current top-down parsing model.

Limitations

While our parsers achieved the best performance with high generalizability, they have serious limitations: Our parsers require a significant amount of computational resources and time. Although our parsers with Llama 2 (7B) work on a standard GPU with 24 GB memory, such as RTX 3090, and those with 13B require over 40 GB GPU memory, those with 70B require a high-end GPU with 80 GB memory, such as A100. To train the parsers, we need 1-2 days for 7B and 13B; however, we need nearly five days for 70B. Furthermore, it takes several minutes to parse a document with our models, whereas it takes only a few seconds with a standard parser.

Acknowledgements

Part of this work was supported by JSPS KAKENHI Grant Numbers P21H03505.

References

Rami Al-Rfou', Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed word representations for multilingual NLP. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 183–192, Sofia, Bulgaria. Association for Computational Linguistics.

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, 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.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Junxuan Chen, Xiang Li, Jiarui Zhang, Chulun Zhou, Jianwei Cui, Bin Wang, and Jinsong Su. 2020. Modeling discourse structure for document-level neural machine translation. In *Proceedings of the First Workshop on Automatic Simultaneous Translation*, pages 30–36, Seattle, Washington. Association for Computational Linguistics.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. *CoRR*, abs/2305.14314.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. 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.

Grigorii Guz and Giuseppe Carenini. 2020. Coreference for discourse parsing: A neural approach. In *Proceedings of the First Workshop on Computational Approaches to Discourse*, pages 160–167, Online. Association for Computational Linguistics.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. DeBERTa: Decoding-enhanced BERT with disentangled attention. In *Proceedings of the International Conference on Learning Representations*.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Xinyu Hu and Xiaojun Wan. 2023. Rst discourse parsing as text-to-text generation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 31:3278–3289.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2020. SpanBERT: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.

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 AAAI Conference on Artificial Intelligence*, pages 8099–8106.

Naoki Kobayashi, Tsutomu Hirao, Hidetaka Kamigaito, Manabu Okumura, and Masaaki Nagata. 2022. A simple and strong baseline for end-to-end neural RST-style discourse parsing. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 6725–6737, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Jingun Kwon, Naoki Kobayashi, Hidetaka Kamigaito, and Manabu Okumura. 2021. Considering nested tree structure in sentence extractive summarization with pre-trained transformer. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4039–4044, Online and Punta Cana, Dominican Republic. 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.

Yang Janet Liu and Amir Zeldes. 2023. Why can't discourse parsing generalize? a thorough investigation of the impact of data diversity. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 3112–3130, Dubrovnik, Croatia. Association for Computational Linguistics.

Zhengyuan Liu, Ke Shi, and Nancy Chen. 2021. DMRST: A joint framework for document-level multilingual RST discourse segmentation and parsing. In *Proceedings of the 2nd Workshop on Computational Approaches to Discourse*, pages 154–164,

Punta Cana, Dominican Republic and Online. Association for Computational Linguistics.

Mary Ellen Okurowski Lynn Carison, Daniel Marcu. 2002. *RST Discourse Treebank*. Philadelphia: Linguistic Data Consortium.

W.C. Mann and S.A Thompson. 1987. Rhetorical structure theory: A theory of text organization. Technical Report ISI/RS-87-190, USC/ISI.

Daniel Marcu. 1998. Improving summarization through rhetorical parsing tuning. In *Sixth Workshop on Very Large Corpora*.

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.

Thanh-Tung Nguyen, Xuan-Phi Nguyen, Shafiq Joty, and Xiaoli Li. 2021. RST parsing from scratch. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1613–1625, Online. Association for Computational Linguistics.

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, Doha, Qatar. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Kenji Sagae and Alon Lavie. 2005. A classifier-based parser with linear run-time complexity. In *Proceedings of the Ninth International Workshop on Parsing Technology*, pages 125–132, Vancouver, British Columbia. Association for Computational Linguistics.

Ke Shi, Zhengyuan Liu, and Nancy F. Chen. 2020. An end-to-end document-level neural discourse parser exploiting multi-granularity representations. *CoRR*, abs/2012.11169.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2020. Mpnet: Masked and permuted pre-training for language understanding. In *Advances in Neural Information Processing Systems*, volume 33, pages 16857–16867. Curran Associates, Inc.

Rajen Subba and Barbara Di Eugenio. 2009. An effective discourse parser that uses rich linguistic information. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 566–574, Boulder, Colorado. Association for Computational Linguistics.

Xin Tan, Longyin Zhang, Fang Kong, and Guodong Zhou. 2022. Towards discourse-aware document-level neural machine translation. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 4383–4389. International Joint Conferences on Artificial Intelligence Organization. Main Track.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models.

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.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V Le. 2022. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame,

Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Zihao Wu, Lu Zhang, Chao Cao, Xiaowei Yu, Haixing Dai, Chong Ma, Zhengliang Liu, Lin Zhao, Gang Li, Wei Liu, Quanzheng Li, Dinggang Shen, Xiang Li, Dajiang Zhu, and Tianming Liu. 2023. Exploring the trade-offs: Unified large language models vs local fine-tuned models for highly-specific radiology nli task.

Jiacheng Xu, Zhe Gan, Yu Cheng, and Jingjing Liu. 2020. Discourse-aware neural extractive text summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5021–5031, Online. 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. 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.

Nan Yu, Meishan Zhang, Guohong Fu, and Min Zhang. 2022. RST discourse parsing with second-stage EDU-level pre-training. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4269–4280, Dublin, Ireland. Association for Computational Linguistics.

Amir Zeldes. 2017. The GUM corpus: Creating multilayer resources in the classroom. *Language Resources and Evaluation*, 51(3):581–612.

Longyin Zhang, Fang Kong, and Guodong Zhou. 2021a. Adversarial learning for discourse rhetorical structure parsing. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3946–3957, Online. Association for Computational Linguistics.

Ying Zhang, Hidetaka Kamigaito, and Manabu Okumura. 2021b. A language model-based generative classifier for sentence-level discourse parsing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2432–2446, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Computing Interface	
Experiments for Llama 2 (7B)	NVIDIA RTX 3090
Experiments for Llama 2 (13B)	NVIDIA RTX A6000
Experiments for Llama 2 (70B)	NVIDIA A100 (80 GB)
Hyperparameters	
number of training epochs	5
batch size	16
optimizer	Adam
learning rate	2e-4
learning rate scheduler	Linear warm-up and cosine annealing
warm-up ratio	0.03
gradient clipping	1.0
lora r	64
lora α	16
lora dropout ratio	0.1
lora target modules	All linear layers in transformer-blocks
quantization for Llama 2	4-bit NormalFloat and double quantization

Table 4: Hyperparameters in the experiments

A Hyperparameters

Table 4 shows the hyperparameters and computing interfaces used in our experiments.