StructSP: Efficient Fine-tuning of Task-Oriented Dialog System by Using Structure-aware Boosting and Grammar Constraints

Dinh-Truong Do* and **Minh-Phuong Nguyen*** and **Le-Minh Nguyen**+
Japan Advanced Institute of Science and Technology, Japan
{truongdo,phuongnm,nguyenml}@jaist.ac.jp

Abstract

We have investigated methods utilizing hierarchical structure information representation in the semantic parsing task and have devised a method that reinforces the semantic awareness of a pre-trained language model via a two-step fine-tuning mechanism: hierarchical structure information strengthening and a final specific task. The model used is better than existing ones at learning the contextual representations of utterances embedded within its hierarchical semantic structure and thereby improves system performance. In addition, we created a mechanism using inductive grammar to dynamically prune the unpromising directions in the semantic structure parsing process. Finally, through experiments[1] on the TOP and TOPv2 (low-resource setting) datasets, we achieved state-of-the-art (SOTA) performance, confirming the effectiveness of our proposed model.

1 Introduction

Task-oriented dialog systems are computer systems designed to perform a specific task (Bai et al., 2022). These systems have a wide range of applications in modern business (Zhang et al., 2020b) and daily lives (Yan et al., 2017), and so on. The semantic parsing model at the core of these systems plays an essential role in converting user utterances into machine-understandable representations for use in capturing the semantic meaning and returning appropriate responses. The introduction of hierarchical representation by Gupta et al. (2018) demonstrated the importance of nested sublogic composition in a task-oriented dialog system. Although this representation is flexible enough to capture the meaning of complicated queries, it also challenges models to identify labels and select the corresponding spans of constituent semantics in the natural sentence.

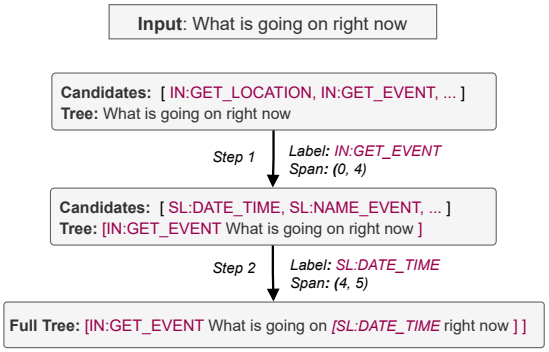

Figure 1: Example of the parsing process in our framework. Each tree is a linearized representation, where *IN:*, *SL:* represent the intent and slot, respectively. Logical tokens are highlighted in red.

Pre-trained language models (e.g., BERT (Devlin et al., 2019)) have recently achieved impressive results on many tasks, including semantic parsing (Ziai, 2019; Herzig and Berant, 2021; Rubin and Berant, 2021). Among them, the Recursive INsertion-based Encoding (RINE) model (Mansimov and Zhang, 2022) produces SOTA results by using *recursive insertion-based* mechanism that directly injects the decoded tree from the previous steps as input to the current step. Although many semantic parsing systems have shown impressive results, there is still much room for improvement in their overall performance.

A promising way to improve the performance of a task-oriented dialog system is to use a pre-trained language model based on the characteristics of this task. The language models is pre-trained on large-scale unstructured text, which means that it does not explicitly learn the logical structure of any sentences. Since information about sentence structures is crucial in the semantic parsing task. Besides, the hierarchical semantic representation in a task-oriented dialog system usually follows a grammar based on a specific task or domain. For

*Equal contribution
+Corresponding author
[1] The source code of this work is released at https://github.com/truongdo619/StructSP

Findings of the Association for Computational Linguistics: ACL 2023, pages 10206–10220

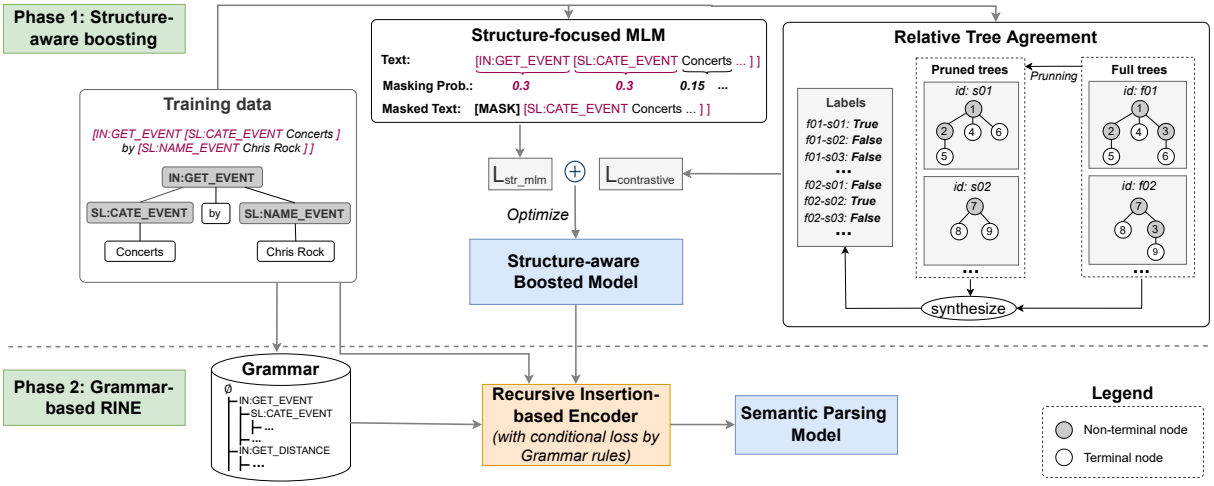

Figure 2: System architecture of StructSP framework.

example, the parent node of slot SL:NAME_EVENT is always the intent IN:GET_EVENT, etc. However, existing methods ignore this information (Mansimov and Zhang, 2022). We thus focused on making two improvements: strengthening the hierarchical structure awareness of the pre-trained language model and using dynamically pruning the unpromising decoding directions by using inductive grammar (Figure 2).

In this paper, we introduce StructSP, a framework for effectively embedding the structured logic information of utterances into a pre-trained language model and thereby enhancing the performance of the semantic parsing system. Our method is generalized and can be adapted to any logical structure, such as a logical form, by using λ-calculus or Prolog syntax (Dong and Lapata, 2016). In particular, we exploit the hierarchical representation, a deep fine-grained structure widely used in semantic parsing (Zhao et al., 2022b; Desai and Aly, 2021; Louvan and Magnini, 2020). Our method consists of two phases: structure-aware boosting and grammar-based RINE. The **structure-aware boosting** phase aims to enhance the structured information of the pre-trained language models and includes two subtasks. The first is structure-focused mask language modeling (MLM), which extends the standard MLM task (Devlin et al., 2019) by focusing more on the logical units in the linearized hierarchical representation. The second subtask is *relative tree agreement* where relative trees are the trees parsed in the middle steps of the parsing process (Figure 1). This subtask enables the model to represent the relative trees with closer hidden vectors and learn the hidden relationship between them. The second phase, **grammar-based RINE** uses the grammar extracted from annotated data to support node label prediction. Incorporating structure information into the model enabled our StructSP to achieve an exact match score on the TOP dataset 0.61 points better than with a SOTA method and demonstrated similar potential results on the TOPv2 datasets (low-resource setting).

This work makes three contributions: (1) an effective fine-tuning approach is introduced for incorporating hierarchical semantic structured information into a pre-trained language model; (2) the use of grammar is introduced in the parsing process to reduce unpromising node label prediction; (3) the StructSP framework is shown to outperform existing models in task-oriented semantic parsing on two datasets, TOP and TOPv2.

2 Related Work

Overview Task-oriented parsing (TOP) (Gupta et al., 2018) and its variant TOPv2 (Chen et al., 2020) were created as benchmarks for assessing the performance of task-oriented semantic parsing models. Various approaches have been proposed for tackling the semantic parsing task on these datasets. Zhu et al. (2020) introduced a non-autoregressive sequence-to-sequence semantic parsing model, which is based on the Insertion Transformer model (Stern et al., 2019). Aghajanyan et al. (2020) introduced an extension of the hierarchical representation, *"decoupled representation"*, and used sequence-to-sequence (S2S) models based on the pointer generator architecture to parse this representation. Rongali et al. (2020) introduced a unified architecture based on S2S mod-

els and a pointer-generator network for semantic parsing. Einolghozati et al. (2019) introduced a shift-reduce parser based on recurrent neural network grammars (RNNGs) with three improvements to the base RNNG model: incorporating contextualized embeddings, ensembling, and pairwise reranking based on a language model. Additionally, Zhao et al. (2022a) showed that compositional TOP could be formulated as abstractive question answering (QA), where the parse tree nodes are generated by posing queries to a QA model. The RINE model (Mansimov and Zhang, 2022) splits the work of parsing an utterance into multiple steps, where the input of each step is the output of the previous step. However, even SOTA methods ignored the hierarchical structure information during the parsing process. We thus focused on utilizing the hierarchical structure information by using inductive grammar extracted from annotated data to guide the node label predictions.

Pre-trained Language Model Adaptation Several recent studies have demonstrated the value of adapting pre-trained language models to specific tasks with different training objectives, such as in summarization (Zhang et al., 2020a) and knowledge inference (Sun et al., 2019; Liu et al., 2020). In the realm of semantic parsing, the SCORE pre-training method (Yu et al., 2020b) focuses on inducing representations that capture the alignment between dialogue flow and structural context in conversational semantic parsing tasks. Grappa (Yu et al., 2020a) is a pre-training approach designed for table semantic parsing and seeks to learn an inductive compositional bias through the joint representation of textual and tabular data. Bai et al. (2022) introduced a semantic-based pre-training approach that uses abstract meaning representation (AMR) as explicit semantic knowledge to capture the core semantic information of utterances in dialogues. In comparison, to strengthening hierarchical structure information, we continue training the model by using only annotated data for the fine-tuning task instead of generating (or augmenting) a large artificial dataset to learn structure information (Yu et al., 2020b,a). This enables the model to effectively performs the hierarchical semantic parsing task using fewer computational resources.

Grammar-Constrained Neural Network Models Incorporating constraint grammar in text generation tasks using neural networks is a topic of interest to researchers. Yin and Neubig (2017) proposed an approach using a grammar model to generate an abstract syntax tree by a series of actions. Krishnamurthy et al. (2017) presented a type-constrained semantic parsing model, which ensures that the decoder only generates well-typed logical forms. Shin et al. (2021) demonstrated that the constrained decoding process with an intermediate sub-language mapped by a large language model can support parsing the target language. More recently, Baranowski and Hochgeschwender (2021) extracted context-free grammar from the target logical form of the semantic parsing task. These grammar constraints were then enforced with an LR parser to maintain syntactically valid sequences throughout decoding. In comparison, our approach utilizes grammar as additional structured information during training with a conditional loss function. This key difference sets our approach apart from previous works.

3 Method

3.1 Overview

Our StructSP framework consists of two fine-tuning phases: *structure-aware boosting* and *grammar-based RINE* (Figure 2). In the first phase, Structure-Aware Boosting, we use a pre-trained RoBERTa model (Liu et al., 2019) as a backbone and continue to train it in a multi-task setting. This improves the model's ability to express structured information of hierarchical representation by using two sub-tasks: *structure-focused MLM* (section 3.2.1) and *relative tree agreement* (section 3.2.2).

In the next phase, we use grammar-based RINE, which uses the RINE approach (Mansimov and Zhang, 2022) augmented by grammar rules to tackle the problem of the hierarchical semantic parsing task. The parsing process is split into multiple steps, where the input of each step is the output of the previous one. The structure-aware model from the previous phase is used as an encoder to make predictions. Especially, an inductive grammar synthesized using training data is used to prune the unpromising decoding directions. The grammar rules not only correct the parsing process but also reduce resource usage by eliminating unnecessary predictions.

3.2 Structure-aware Boosting

3.2.1 Structure-focused MLM

Before introducing our proposed structure-focused masking strategy, we describe the original MLM (Devlin et al., 2019).

Original MLM Given a sequence of tokens $\mathcal{X} = [x_1, x_2, ..., x_m]$, probability $T = [t_1, t_2, ..., t_m]$ corresponding to the input sequence is generated, where t_i indicates the probability that token x_i will be chosen for masking. Following Devlin et al. (2019), all t_i are equal to 15%, meaning that every token has the same probability of being chosen in the masking process. Let $\mathcal{D}$ be the set of tokens randomly chosen for masking (e.g., $\mathcal{D} = \{x_2\}$), and let $\mathcal{X}'$ be the sentence with masked tokens (e.g., $\mathcal{X}' = [x_1, \texttt{[MASK]}, ...x_m]$). The cross-entropy loss function is computed for all masked tokens:

$$\mathcal{L}_{mlm} = -\frac{1}{|\mathcal{D}|} \sum_{i=1}^{|\mathcal{D}|} \sum_{j=1}^{|\mathcal{V}|} x_{ij} \log p(x_{ij} \mid \mathcal{X}') \quad (1)$$

where $|\mathcal{D}|$ is the total number of masked tokens in a sentence, $|\mathcal{V}|$ is the size of vocabulary of language model, and $p(x_{ij} \mid \mathcal{X}')$ is the probability that the model assigns to the token x_i belong to token j^{th} in vocabulary given the context provided by the masked sequence. The model is trained to minimize this loss function, which measures how well it can predict the masked tokens on the basis of the context provided by the unmasked ones

Structure-focused MLM In the linearized hierarchical representation of an utterance (Table 1), there are two types of tokens: normal and logical. The logical tokens can be divided into two sub-categories: bracket tokens ("[" or "]") indicate the span of a label, and label tokens represent the label itself (e.g., `IN:GET_EVENT`). The correct prediction of a masked bracket token demonstrates the model's understanding of the label span. Similarly, the correct prediction of a masked label token demonstrates the model's understanding of the label type. Therefore, masking logical tokens to train the model for learning the hierarchical structure is a reasonable approach (Bai et al., 2022). However, the original MLM treats all tokens equally, regardless of whether they are normal or logical.

In our approach, we modify the original MLM approach by assigning a higher masking probability to logical tokens, thereby pushing the model to pay more attention to the hierarchical structure (the logical token) information. This enables the model to learn the structural characteristic of natural sentences while retaining the knowledge learned from the original MLM (Yu et al., 2020a). More specifically, our *structure-focused MLM* strategy gives logical tokens a higher masking probability, defined as α (with $\alpha > 15\%$). Let $T' = [t'_1, t'_2, ..., t'_m]$ be the revised sequence of masking probabilities, where $t'_i = \alpha$ if the corresponding i^{th} token is a logical one. The *structured-focused MLM* loss ($\mathcal{L}_{str\_mlm}$) function of masked tokens is computed similarly to the original MLM loss.

3.2.2 Relative Tree Agreement

In hierarchical representation, leaf nodes are words, while non-terminal nodes are semantic tokens: either intents or slots. We define a *non-terminal list* as an ordered list of non-terminal nodes by using a breadth-first search. For example, using the tree shown in Figure 2, we obtain the non-terminal list [`IN:GET_EVENT`, `SL:CATE_EVENT`, `SL:NAME_EVENT`]. This list is then used to form positive training samples.

The goal of this task is to improve the ability of the model to produce similar representations for trees in the same parsing process, enabling it to effectively perform the recursive insertion-based semantic parsing task. To achieve this goal, a *pruned tree* is generated at each training iteration by randomly selecting a node from the non-terminal list of the *full tree* (i.e., the annotated tree) and pruning all non-terminal nodes to the right of it in the list. This pruned tree is *relative* to the full tree. Formally, we denote the linearized representation of the full tree as $\mathcal{P}_{full}$ and that of the pruned tree as $\mathcal{P}_{pruned}$. The hidden vector representations of these two trees (h_{full} and h_{pruned}) are encoded using a pre-trained language model:

$$\begin{aligned} h_{full} &= \text{RoBERTa}(\mathcal{P}_{full}) \\ h_{pruned} &= \text{RoBERTa}(\mathcal{P}_{pruned}) \end{aligned} \quad (2)$$

We use contrastive learning (Frosst et al., 2019; Gao et al., 2021; Bai et al., 2022; Luo et al., 2022) to train our model for this task. The aim is to align the representations of positive pairs in the latent space by minimizing the distance between them while simultaneously maximizing the distance between negative pairs. Specifically, we define a contrastive loss function that measures the similarity between the hidden states h_{full} and h_{pruned}. This information is then used to update the model's parameters. Minimizing this loss function enables

our model to learn how to produce more similar representations for the full and pruned trees. In particular, given a training batch B, the positive pair at the i^{th} position is $(h^{(i)}_{full}, h^{(i)}_{pruned})$, and the negative pairs are the other samples in the batch. The training objective is given as follows:

$$\mathcal{L}_{contrastive} = -\log \frac{\exp(\text{sim}(h^{(i)}_{full}, h^{(i)}_{pruned})/\tau)}{\sum_{j \in B} \exp(\text{sim}(h^{(i)}_{full}, h^{(j)}_{pruned})/\tau)} \quad (3)$$

where sim is the similarity function based on cosine distance, $h^{(i)}_{full}$ and $h^{(i)}_{pruned}$ are the hidden states for the full and pruned trees, respectively, at the i^{th} position in the batch indexes. τ is a temperature hyperparameter (Gao et al., 2021), and B is the set of samples in the mini-batch.

3.2.3 Objective Function

We combine the objectives of the two sub-tasks to formulate the *structure-aware boosting* loss ($\mathcal{L}_{sab}$).

$$\mathcal{L}_{sab} = \mathcal{L}_{contrastive} + \lambda * \mathcal{L}_{str\_mlm} \quad (4)$$

where λ is a hyperparameter that balances the contributions of the two objectives. As proposed elsewhere (Lee et al., 2020; Nandy et al., 2021; Bai et al., 2022), we do not train our model from scratch. Instead, we use a pre-trained language model to initialize its parameters. This enables us to leverage the knowledge contained in the language model.

3.3 Grammar-based RINE

RINE In this phase, we use a recursive insertion-based approach (Mansimov and Zhang, 2022), with our structure-aware boosted model created in the previous phase serving as the backbone. The parsing process can be formally represented as a chain of incremental sub-parsed trees, denoted as $\mathcal{P} = [\mathcal{P}_0, \mathcal{P}_1, ..., \mathcal{P}_{gold}]$ (Table 1).

Table 1: Example chain of incremental trees in parsing process using RINE.

Tree	Linearized representation
$\mathcal{P}_0$	Concerts by Chris Rock
$\mathcal{P}_1$	[IN:GET_EVENT Concerts by Chris Rock]
$\mathcal{P}_2$	[IN:GET_EVENT [SL:CATE_EVENT Concerts] by Chris Rock]
$\mathcal{P}_3$	[IN:GET_EVENT [SL:CATE_EVENT Concerts] by [SL:NAME_EVENT Chris Rock]]

At the i^{th} step, the model receives the tree in the previous step $\mathcal{P}_{i-1}$ as input and predicts the current node label and its span to decode the tree $\mathcal{P}_i$. The tree $\mathcal{P}_i$ is updated to include a new prediction node and processing recursively moves to the next step $i+1$ until a special end-of-prediction signal (EOP) is found.

Grammar constraint Observation revealed that the relationships between nodes in the semantic tree are important information in the parsing process. For example, the intent IN:GET_EVENT typically comes with slots for event information, SL:CATE_EVENT or SL:NAME_EVENT. Therefore, we introduce a grammar integration mechanism into RINE to reduce movement in unpromising decoding directions. Particularly, we construct grammar $\mathcal{G} = \{A \rightarrow B \mid A, B \text{ are non-terminal nodes}\}$ on the basis of training data (e.g., IN:GET_EVENT $\rightarrow$ SL:CATE_EVENT). Integrating grammar into the RINE model means that the parser considers only the potential nodes in a candidate set (e.g., $\mathcal{C} = \mathcal{G}(A)$) on the basis of the parent node (e.g., $B \in \mathcal{C}$) at each step of node label prediction.

Modeling For mathematical modeling, given subtree $\mathcal{P}_i$ at the i^{th} step, our model first encodes this tree as a sequence and obtains the corresponding hidden states fused by context. Following Mansimov and Zhang (2022), we use hidden states of the [CLS] token in the last transformer encoder layer to predict the label of the next node. Furthermore, to predict the range of this node, we use all hidden states in the last two layers to compute the probabilities of the start and end positions.

$$p^{nodeLb} = \text{softmax}(W_{lb} h^{(l)}_{[CLS]} + b_{lb}) \quad (5)$$

$$p^{start}_k = \text{softmax}(W_s h^{(l)}_{w_k} + b_s) \quad (6)$$

$$p^{end}_k = \text{softmax}(W_e h^{(l-1)}_{w_k} + b_e) \quad (7)$$

where $W\cdot$ and $b\cdot$ are the learnable parameters, l is the number of transformer encoder layers in the pre-trained language model, and w_k is the k^{th} token (word) index of the input sequence. We use a special penalty score ($s^{penalty}$) injected into the loss function to cause the unpromising label node predictions to be ignored:

$$s^{penalty} = \begin{cases} 0 & \text{if} \quad nodeLb \in \mathcal{C} \\ +\infty & \text{otherwise} \end{cases} \quad (8)$$

where $\mathcal{C}$ is the candidates generated from the grammar and the parent node. Finally, the cross-entropy (CE) losses are synthesized in accordance with the

gold label (g^{nodeLb}) and these output probabilities:

$$\mathcal{L}_{nodeLb} = \text{CE}(g^{nodeLb}, p^{nodeLb}) + s^{penalty} \quad (9)$$

$$\mathcal{L}_{start} = \sum_k \text{CE}(g_k^{start}, p_k^{start}) \quad (10)$$

$$\mathcal{L}_{end} = \sum_k \text{CE}(g_k^{end}, p_k^{end}) \quad (11)$$

$$\mathcal{L}_{final} = \mathcal{L}_{nodeLb} + \mathcal{L}_{start} + \mathcal{L}_{end} \quad (12)$$

Inference challenge In the inference step, the challenge with the approach is a lack of information about which node is the parent node at the current step to be used to generate the candidate set. We propose a simple strategy to meet this challenge: use the result of span prediction to determine the parent node. For example, considering the 3^{rd} step in Table 1 and given input $\mathcal{P}_2$, we have no information about the next node to be predicted and do not know whether the parent node is `IN:GET_EVENT` or `SL:CATE_EVENT`. We predict the span first and obtain the position of *"Chris Rock"*. We thus determine that the parent of the current node is `IN:GET_EVENT`, and can generate candidates $\mathcal{C}$ by using this node.

4 Experiment

To verify the effectiveness of our proposed method, we conducted experiments on datasets TOP[2] and TOPv2[3] (low-resource setting). Following Mansimov and Zhang (2022), we conducted three random seeds for each experimental setting and reported the average results with standard deviation.

4.1 Datasets and Evaluation Metric

The TOP dataset (Gupta et al., 2018) is a collection of utterances divided into two domains: **navigation** and **event**. Twenty-fine intents and 36 slots are used to represent the utterances. The mean depth of the trees in the dataset is 2.54, and the mean length of the utterances is 8.93 tokens. Following previous works (Einolghozati et al., 2019; Rongali et al., 2020; Zhu et al., 2020; Mansimov and Zhang, 2022), we removed utterances with the `UNSUPPORTED` intent, resulting in 28,414 training, 4,032 valid, and 8,241 test utterances.

The TOPv2 dataset (Chen et al., 2020) focuses on scenarios with low resources. We followed Mansimov and Zhang (2022) by using low-resource versions of the **reminder** and **weather** domains. The reminder domain contains 19 intents and 32 slots while the weather domain contains 7 intents and 11 slots. The low-resource data were created by selecting a fixed number of training *samples per intent and slot* (SPIS) from the original dataset. If an intent or slot occurred fewer times than the specified number of samples, all of the parse trees containing that intent or slot were included in the low-resource data. Specifically, we used the training, validation, and test data at 500 and 25 SPIS. At 500 SPIS, the reminder domain has 4788 training and 1871 validation samples, and the weather domain has 2372 training and 1202 validation samples. At 25 SPIS, the reminder domain has 493 training and 337 validation samples, and the weather domain has 176 training and 147 validation samples. For both SPIS settings, the test sets for the reminder and weather domains contained 5767 and 5682 test samples, respectively.

Identical to previous studies (Einolghozati et al., 2019; Rongali et al., 2020; Zhu et al., 2020; Mansimov and Zhang, 2022), we used exact match (EM) as performance metric. The EM score was calculated as the number of utterances for which fully parsed trees were correctly predicted.

4.2 Experimental setting

Structure-aware boosting: We used the RoBERTa (Liu et al., 2019) model as our backbone. With the TOP dataset, we used a peak learning rate of 1e-05 and continued training for ten epochs using the Adam optimizer (Kingma and Ba, 2014) with epsilon 1e-08 and a batch size of 16 sequences with 512 tokens. For the hyperparameters, we set $\tau = 1.0$ and selected λ from {**0.5**, 1.0}[4] . With the TOPv2 dataset, we used the same settings as for the TOP dataset except we increased the number of training epochs to 50 for the 25 SPIS setting. The training process on a single NVIDIA A100 GPU.

Grammar-based RINE: we used the model trained from the structure-aware boosting phase as an encoder. For the TOP dataset, we set the number of warmup steps to 1000, selected the learning rate from {**1e-05**, 2e-05}, and performed training for 50 epochs. The best checkpoint was chosen on the basis of the model's performance on the validation set. We utilized the Adam optimizer and a batch size of 32 sequences containing 512 tokens. For the TOPv2 dataset, we used the same settings

[2] Provided under the CC-BY-SA license

[3] Provided under the CC BY-NC 4.0 license

[4] The values resulting in the best performance are **bold**.

except for adjusting the batch size to 16 for the 500 SPIS setting and 8 for the 25 SPIS setting.

Baseline: We reproduced the RINE model (Mansimov and Zhang, 2022) as a strong baseline model and kept the same hyperparameter values as the original paper (Mansimov and Zhang, 2022).

4.3 Main Results

TOP dataset We evaluated the performance of our proposed StructSP method on the TOP dataset and compared it with that of the other works. The results are shown in Table 2.

Table 2: Performance comparison for TOP test set.

Method	Pre-trained	EM
Seq2seq (Bai et al., 2022)	SARA-ROBERTA	82.78
RNNG ensem. + SVMRank (Einolghozati et al., 2019)	ELMo	87.25
Non-AR S2S-Ptr (Shrivastava et al., 2021)	RoBERTa$_{base}$	85.07
S2S-Ptr (Rongali et al., 2020)	RoBERTa$_{base}$	86.67
Decoupled S2S-Ptr (Aghajanyan et al., 2020)	RoBERTa$_{large}$	87.10
Insertion Transformer + S2S-Ptr (Zhu et al., 2020)	RoBERTa$_{base}$	86.74
RINE (Mansimov and Zhang, 2022)	RoBERTa$_{large}$	87.57 ± 0.03
Non-grammar StructSP (ours)	RoBERTa$_{large}$	87.89 ± 0.08
StructSP (ours)	RoBERTa$_{large}$	**88.18 ± 0.24**

Our model outperformed all the other models. Specifically, it achieved a higher score than the autoregressive seq2seq model with pointer (Rongali et al., 2020) by 1.51 EM and outperformed the best method of previous works, RINE (Mansimov and Zhang, 2022) by 0.61 EM scores. These results show the effectiveness of our proposed method in injecting prior semantic hierarchical structure information of a natural sentence into the pre-trained language model. Besides, our result outperformed the result of Bai et al. (2022) by a large margin, although the authors also strengthened the structure information of pre-trained models using a general AMR structure. Additionally, using grammar improved the performance of our model by 0.29 points compared with a version of the model trained without using grammar. This demonstrates the value of incorporating grammar into our model when using the TOP dataset.

TOPv2 dataset We evaluated the performance of our proposed StructSP method on the low-resource data of the reminder and weather domains of the TOPv2 dataset and compared it with that of the other methods. The results are shown in Table 3. Our model outperformed the others for all SPIS settings. At the 25 SPIS setting, our StructSP outperforms the baseline by 0.93 EM and 1.02 EM in the weather and reminder domains. Notice that, with these low-resource settings, in the structure-aware boosting phase, the model also only continues to train with limited data but still achieves impressive improvements.

In addition, at the 25 SPIS setting of the weather domain, we observed that the performance of the StructSP was improved when it was trained without using grammar. We argue that this was due to the size of the training set at this setting being extremely small (176 samples), and the extracted grammar from this training set is not expressive enough to cover the grammar in the validation and the test set.

5 Analysis

5.1 Ablation Study

To evaluate the effect of using each component in our framework, we compared the model's performance for each combination of component settings with that of the RINE baseline model on the validation set of the TOP dataset (Table 4).

We found that using grammar in the second phase grammar-based RINE led to improved performance on the TOP dataset (EM score 0.12 points higher than with the baseline model). Additionally, using the structure-aware boosting phase substantially improved the EM score compared with the baseline (0.52 points higher). To further understand the contributions of each training subtask in the structure-aware boosting phase, we conducted two additional experiments. The results show that using the structure-focused mask language modeling subtask improves EM performance by 0.36 points compared with the baseline while using the relative tree agreement subtask leads to only a 0.05 improvement.

Furthermore, we conducted a t-test (Lehmann and Romano, 1986) with the null hypothesis that the expected values of our full-setting model (StructSP) and the baseline model (RINE) are identical (Table 4). Based on the experimental outcomes, the p-value was found to be below 0.05, which suggests that the proposed method outperforms the baseline model significantly.

Table 3: Performance comparison using exact match score for our StructSP method and previous works on TOPv2 test set.

Method	Pre-trained Model	Exact Match			
		Weather		Reminder	
		25 SPIS	500 SPIS	25 SPIS	500 SPIS
LSTM Seq2Seq-Ptr (Chen et al., 2020)	_	46.2	78.6	21.5	65.9
Seq2seq-Ptr (Chen et al., 2020)	RoBERTa$_{base}$	_	83.5	_	71.9
Seq2seq-Ptr (Chen et al., 2020)	BART$_{large}$	71.6	84.9	55.7	71.9
RINE (Mansimov and Zhang, 2022)	RoBERTa$_{base}$	74.53 ± 0.86	87.80 ± 0.04	68.71 ± 0.46	80.30 ± 0.04
RINE (Mansimov and Zhang, 2022)	RoBERTa$_{large}$	77.03 ± 0.16	87.50 ± 0.28	71.10 ± 0.63	81.31 ± 0.22
Non-grammar StructSP	RoBERTa$_{large}$	**78.24 ± 0.47**	88.00 ± 0.47	72.07 ± 1.24	81.57 ± 0.27
StructSP	RoBERTa$_{large}$	77.96 ± 0.92	**88.08 ± 0.11**	**72.12 ± 1.13**	**82.28 ± 0.24**

Table 4: Results of ablation study on validation set of TOP dataset. Denotations ✓ and ✗ indicate whether corresponding component was used or not, respectively. Δ denotes difference in EM scores between the full-setting model with other models.

Method	Settings			EM	Δ	T-test (Significantly better at 95%?)
	Structure-aware boosting		Grammar			
	Structure focused MLM	Relative tree agreement				
StructSP	✓	✓	✓	**88.26**	_	**yes**
	✗	✗	✓	87.69	-0.57	no
	✓	✓	✗	88.09	-0.17	**yes**
	✓	✗	✗	87.93	-0.33	**yes**
	✗	✓	✗	87.62	-0.64	no
Baseline	✗	✗	✗	87.57	-0.69	_

Table 5: Comparison of outputs[5] of baseline (RINE) and our StructSP model on the validation set of TOP dataset.

Type	Ouput
Input	Where is the nearest Tom Thumb
Ground-Truth	[IN:GET_LOCATION Where is the [SL:LOCATION_MODIFIER nearest] [SL:POINT_ON_MAP Tom Thumb]]
Baseline	[IN:GET_LOCATION Where is the [SL:LOCATION_MODIFIER nearest] [SL:NAME_EVENT Tom Thumb]] ✗
StructSP	[IN:GET_LOCATION Where is the [SL:LOCATION_MODIFIER nearest] [SL:POINT_ON_MAP Tom Thumb]] ✓
Input	What to do after a Pacers game
Ground-Truth	[IN:GET_EVENT What to do [SL:DATE_TIME [IN:GET_EVENT after a [SL:NAME_EVENT Pacers] [SL:CATEGORY_EVENT game]]]]
Baseline	[IN:GET_EVENT What to do after a [SL:NAME_EVENT Pacers] [SL:CATEGORY_EVENT game]] ✗
StructSP	[IN:GET_EVENT What to do [SL:DATE_TIME [IN:GET_EVENT after a [SL:NAME_EVENT Pacers] [SL:CATEGORY_EVENT game]]]] ✓
Input	traffic near me right now
Ground-Truth	[IN:GET_INFO_TRAFFIC traffic [SL:LOCATION [IN:GET_LOCATION [SL:LOCATION_MODIFIER [IN:GET_LOCATION [SL:SEARCH_RADIUS near] [SL:LOCATION_USER me]]]]] [SL:DATE_TIME right now]]
Baseline	[IN:GET_INFO_TRAFFIC traffic [SL:LOCATION [IN:GET_LOCATION [SL:SEARCH_RADIUS near] [SL:LOCATION_USER me]]] [SL:DATE_TIME right now]] ✗
StructSP	[IN:GET_INFO_TRAFFIC traffic [SL:LOCATION [IN:GET_LOCATION [SL:SEARCH_RADIUS near][SL:LOCATION_USER me]]] [SL:DATE_TIME right now]] ✗

5.2 Effect of masking probability α

In another experiment, we analyzed the effect of the logical-token masking probability (α in Section 3.2.1) in the *structured-aware boosting* phase on overall performance (Figure 3). High performance was achieved when α was set to 0.3 or 0.4. We attribute this to our mechanism pushing the model to pay more attention to the logical tokens, helping it to better capture the structure. In summary, these results show that our StructSP method achieved better performance than the baseline for all values of α, which demonstrates the robustness of our approach.

5.3 Case Study

Table 5 presents several example utterances from the TOP dataset. In the first example, our model

[5] Please refer to Appendix C for tree representation.

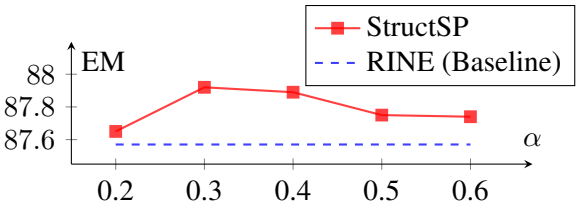

Figure 3: Effect of logical-token masking probability (α) on system performance.

predicted the slot SL:POINT_ON_MAP after the intent IN:GET_LOCATION, whereas the baseline model predicted the slot SL:NAME_EVENT. This difference occurred because the extracted grammar does not contain the constraint (IN:GET_LOCATION => SL:NAME_EVENT). This demonstrates the effectiveness of incorporating grammar into our model. The second example presents a ground-truth tree with a complex structure, requiring the model to identify the span "after a Pacers game" as a date time slot (SL:DATE_TIME) and then correctly parse the structure within this span. Our model was able to correctly return the tree, whereas the baseline model was not. The final example is a particularly challenging one, the tree has a depth of 5, indicating that the tree structure is highly complex. Both models failed to return the correct predictions, suggesting that learning to handle such complicated queries is an interesting topic for future work.

6 Conclusion

We have presented a novel approach to improving the performance of SOTA semantic parsing models on hierarchical representation datasets. In this approach, a model is created that incorporates knowledge of the utterance structures into the semantic parsing process. This is achieved by learning contextual representations from the hierarchical representation of utterances with objective functions targeted at the semantic parsing task as well as by using grammar rules containing knowledge about the structure of the data for training and label prediction. In experiments on the TOP and TOPv2 datasets, our model outperforms previous SOTA approaches.

7 Limitations

There are two main limitations to our works.

(1) Grammar constraint: The results of the StructSP method at the 25 SPIS setting in the TOPv2 dataset (Table 3) suggest that the results of using grammar with low-resource data can be uncertain. The reason is that the extracted grammar from training data for low-resource setting is not general enough to capture the grammar of the new coming data (validation or test set). Therefore, for our StructSP method to work effectively, the provided grammar should cover all grammar rules if possible.

(2) Prediction time: A recursive insertion-based strategy is used for prediction. This means that the output of the previous parsing step is used as input for the current parsing step, and this process continues until a terminal signal is encountered. As a result, parsing a complex tree with multiple intents/slots (labels) can be a lengthy process due to the recursive nature of this method. Future work includes improving parsing prediction time by predicting all labels at the same level in the parsed tree rather than predicting them one by one.

Acknowledgement

This work is partly supported by AOARD grant FA23862214039

References

Armen Aghajanyan, Jean Maillard, Akshat Shrivastava, Keith Diedrick, Michael Haeger, Haoran Li, Yashar Mehdad, Veselin Stoyanov, Anuj Kumar, Mike Lewis, and Sonal Gupta. 2020. Conversational semantic parsing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5026–5035, Online. Association for Computational Linguistics.

Xuefeng Bai, Linfeng Song, and Yue Zhang. 2022. Semantic-based pre-training for dialogue understanding. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 592–607, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Artur Baranowski and Nico Hochgeschwender. 2021. Grammar-constrained neural semantic parsing with LR parsers. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1275–1279, Online. Association for Computational Linguistics.

Xilun Chen, Asish Ghoshal, Yashar Mehdad, Luke Zettlemoyer, and Sonal Gupta. 2020. Low-resource domain adaptation for compositional task-oriented semantic parsing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5090–5100, Online. Association for Computational Linguistics.

Shrey Desai and Ahmed Aly. 2021. Diagnosing transformers in task-oriented semantic parsing. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 57–62, Online. Association for Computational Linguistics.

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.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 33–43, Berlin, Germany. Association for Computational Linguistics.

Arash Einolghozati, Panupong Pasupat, Sonal Gupta, Rushin Shah, Mrinal Mohit, Mike Lewis, and Luke Zettlemoyer. 2019. Improving semantic parsing for task oriented dialog. *arXiv preprint arXiv:1902.06000*.

Nicholas Frosst, Nicolas Papernot, and Geoffrey Hinton. 2019. Analyzing and improving representations with the soft nearest neighbor loss. In *International conference on machine learning*, pages 2012–2020. PMLR.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6894–6910, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Sonal Gupta, Rushin Shah, Mrinal Mohit, Anuj Kumar, and Mike Lewis. 2018. Semantic parsing for task oriented dialog using hierarchical representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2787–2792, Brussels, Belgium. Association for Computational Linguistics.

Jonathan Herzig and Jonathan Berant. 2021. Span-based semantic parsing for compositional generalization. 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 908–921, Online. Association for Computational Linguistics.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Jayant Krishnamurthy, Pradeep Dasigi, and Matt Gardner. 2017. Neural semantic parsing with type constraints for semi-structured tables. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1516–1526, Copenhagen, Denmark. Association for Computational Linguistics.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2020. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240.

Erich Leo Lehmann and Joseph P Romano. 1986. *Testing statistical hypotheses*, volume 3. Springer.

Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Qi Ju, Haotang Deng, and Ping Wang. 2020. K-bert: Enabling language representation with knowledge graph. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 2901–2908.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Samuel Louvan and Bernardo Magnini. 2020. Recent neural methods on slot filling and intent classification for task-oriented dialogue systems: A survey. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 480–496, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Yun Luo, Fang Guo, Zihan Liu, and Yue Zhang. 2022. Mere contrastive learning for cross-domain sentiment analysis. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 7099–7111, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Elman Mansimov and Yi Zhang. 2022. Semantic parsing in task-oriented dialog with recursive insertion-based encoder. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 11067–11075.

Abhilash Nandy, Soumya Sharma, Shubham Maddhashiya, Kapil Sachdeva, Pawan Goyal, and NIloy Ganguly. 2021. Question answering over electronic devices: A new benchmark dataset and a multi-task learning based QA framework. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4600–4609, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Subendhu Rongali, Luca Soldaini, Emilio Monti, and Wael Hamza. 2020. Don't parse, generate! a sequence to sequence architecture for task-oriented semantic parsing. In *Proceedings of The Web Conference 2020*, pages 2962–2968.

Ohad Rubin and Jonathan Berant. 2021. SmBoP: Semi-autoregressive bottom-up semantic parsing. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 311–324, Online. Association for Computational Linguistics.

Richard Shin, Christopher Lin, Sam Thomson, Charles Chen, Subhro Roy, Emmanouil Antonios Platanios, Adam Pauls, Dan Klein, Jason Eisner, and Benjamin Van Durme. 2021. Constrained language models yield few-shot semantic parsers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7699–7715, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Akshat Shrivastava, Pierce Chuang, Arun Babu, Shrey Desai, Abhinav Arora, Alexander Zotov, and Ahmed Aly. 2021. Span pointer networks for non-autoregressive task-oriented semantic parsing. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 1873–1886, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Mitchell Stern, William Chan, Jamie Kiros, and Jakob Uszkoreit. 2019. Insertion transformer: Flexible sequence generation via insertion operations. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5976–5985. PMLR.

Yu Sun, Shuohuan Wang, Yu-Kun Li, Shikun Feng, Xuyi Chen, Han Zhang, Xin Tian, Danxiang Zhu, Hao Tian, and Hua Wu. 2019. ERNIE: enhanced representation through knowledge integration. *CoRR*, abs/1904.09223.

Zhao Yan, Nan Duan, Peng Chen, Ming Zhou, Jianshe Zhou, and Zhoujun Li. 2017. Building task-oriented dialogue systems for online shopping. In *Thirty-first AAAI conference on artificial intelligence*.

Pengcheng Yin and Graham Neubig. 2017. A syntactic neural model for general-purpose code generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 440–450, Vancouver, Canada. Association for Computational Linguistics.

Tao Yu, Chien-Sheng Wu, Xi Victoria Lin, Bailin Wang, Yi Chern Tan, Xinyi Yang, Dragomir Radev, Richard Socher, and Caiming Xiong. 2020a. Grappa: grammar-augmented pre-training for table semantic parsing. *arXiv preprint arXiv:2009.13845*.

Tao Yu, Rui Zhang, Alex Polozov, Christopher Meek, and Ahmed Hassan Awadallah. 2020b. Score: Pre-training for context representation in conversational semantic parsing. In *International Conference on Learning Representations*.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. 2020a. PEGASUS: Pre-training with extracted gap-sentences for abstractive summarization. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 11328–11339. PMLR.

Zheng Zhang, Ryuichi Takanobu, Qi Zhu, MinLie Huang, and XiaoYan Zhu. 2020b. Recent advances and challenges in task-oriented dialog systems. *Science China Technological Sciences*, 63(10):2011–2027.

Wenting Zhao, Konstantine Arkoudas, Weiqi Sun, and Claire Cardie. 2022a. Compositional task-oriented parsing as abstractive question answering. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4418–4427, Seattle, United States. Association for Computational Linguistics.

Yingxiu Zhao, Yinhe Zheng, Zhiliang Tian, Chang Gao, Bowen Yu, Haiyang Yu, Yongbin Li, Jian Sun, and Nevin L Zhang. 2022b. Prompt conditioned vae: Enhancing generative replay for lifelong learning in task-oriented dialogue. *arXiv preprint arXiv:2210.07783*.

Qile Zhu, Haidar Khan, Saleh Soltan, Stephen Rawls, and Wael Hamza. 2020. Don't parse, insert: Multilingual semantic parsing with insertion based decoding. In *Proceedings of the 24th Conference on Computational Natural Language Learning*, pages 496–506, Online. Association for Computational Linguistics.

Amir Ziai. 2019. Compositional pre-training for neural semantic parsing. *arXiv preprint arXiv:1905.11531*.

A Data Pre-processing

Before training our models, we perform the following data processing steps:

- **Step 1:** Extract the grammar using the full trees of the training samples, with the constraint *"parent node -> child node"* (Eg. `IN:GET_LOCATION -> SL:NAME_EVENT`).
- **Step 2:** Convert the ground-truth full trees into multiple sub-parsed trees, each represented as a triple *(Current parsed tree, Label, Label span)* (Figure 1).
- **Step 3:** Extract the set of label types from the training samples.

With the above data, we then proceed to train our models in two phases.

B Model Hyper-Parameters

The hyper-parameters used in our models in the TOP dataset are shown in Table 6. In the TOPv2 dataset, we use the same hyper-parameters with the following exceptions: for the 25 SPIS setting in the structure-aware boosting phase, the training epoch was adjusted to 50; for the 500 SPIS and 25 SPIS settings in the label prediction phase, the batch size was adjusted to 16 and 8, respectively. The hyper-parameters used in our models in the TOP dataset Our framework is implemented using Pytorch[6] and HuggingFace Transformers.[7] . Our source code and extracted grammar can be found at: **[masked]** (access will be granted upon acceptance).

Table 6: Hyper-parameters of our models in TOP dataset

Phase	Hyper-Parameter	Value
Structure Enhancement	Batch size	16
	Learning Rate	1e-5
	Sequence length	300
	MLM Weight (θ)	0.5
	Training Epoch	10
Label Prediction	Batch Size	32
	Optimizer	Adam
	Learning Rate	1e-5
	Warmup Step	1000
	Max Training Epoch	50
	Max length	512
	Logical Token Masking Prob. (α)	0.3
	RoBERTa Attention Dropout	0.2
	MLP Dropout	0.5

C Case study outputs with tree representation

Figure 5 presents the example of the parsing process described in Section 1 using tree representation. Additionally, Figure 5 displays the outputs of examples discussed in Section 5.3 using tree representation.

[6] https://github.com/pytorch/pytorch
[7] https://github.com/huggingface/transformers

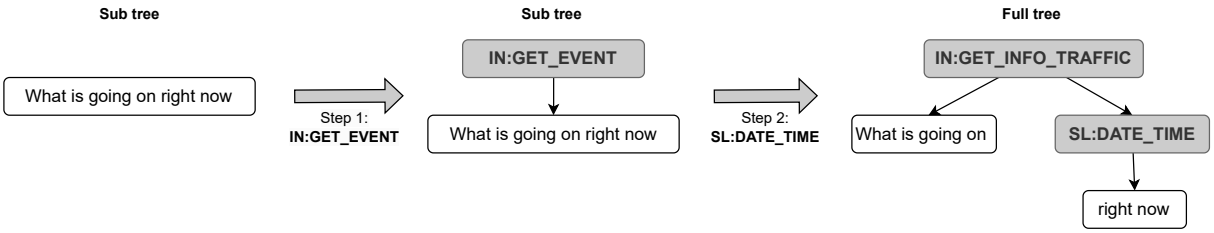

Figure 4: Example of parsing process with tree representation.

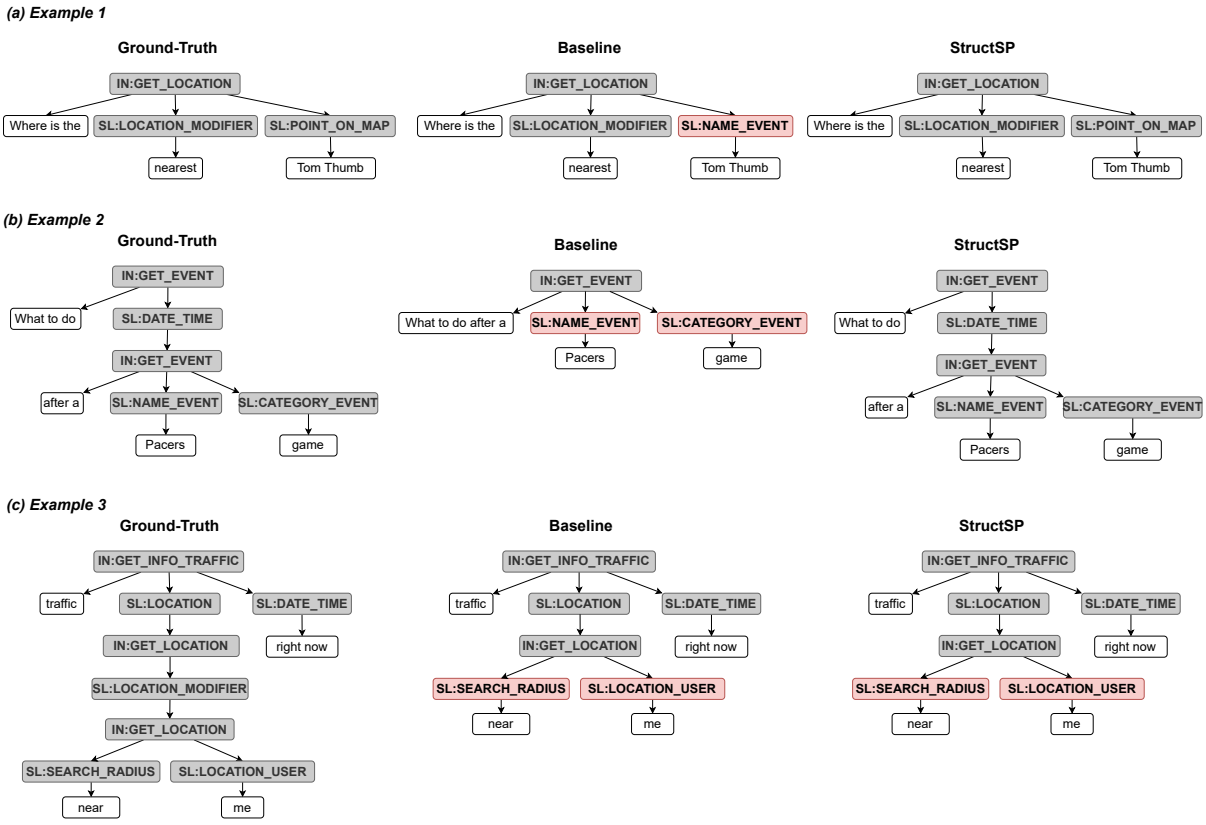

Figure 5: Case study outputs with tree representation.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
7

☒ A2. Did you discuss any potential risks of your work?
Our research focuses on improving the performance of the semantic parsing task, which is experimented on well-known published datasets.

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Left blank.

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

4

☑ B1. Did you cite the creators of artifacts you used?
4

☑ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
4

☑ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
4

☒ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
This data is used by previous works.

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
4

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
4

C ☑ Did you run computational experiments?

4

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
4

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
4

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
5

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
4

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.