Semantic Parsing in Task-Oriented Dialog with Recursive Insertion-Based Encoder

Elman Mansimov and Yi Zhang
AWS AI Labs
{mansimov, yizhngn}@amazon.com

Abstract

We introduce **R**ecursive **IN**sertion-based **E**ncoder (RINE), a novel approach for semantic parsing in task-oriented dialog. Our model consists of an encoder network that incrementally builds the semantic parse tree by predicting the non-terminal label and its positions in the linearized tree. At the generation time, the model constructs the semantic parse tree by recursively inserting the predicted non-terminal labels at the predicted positions until termination. RINE achieves state-of-the-art exact match accuracy on low- and high-resource versions of the conversational semantic parsing benchmark `TOP`, outperforming strong sequence-to-sequence models and transition-based parsers. We also show that our model design is applicable to nested named entity recognition task, where it performs on par with state-of-the-art approach designed for that task. Finally, we demonstrate that our approach is $2-3.5\times$ faster than the sequence-to-sequence model at inference time.

Introduction

Task-oriented dialog systems are playing an increasingly important role in modern business and social lives of people by facilitating information access and automation of routine tasks through natural language conversations. At the core of such dialog systems, a natural language understanding component interprets user input utterances into a meaning representation. While the traditional intent-slot based approach can go a long way, such flat meaning representation falls short of capturing the nuances of natural languages, where phenomena such as conjunction, negation, co-reference, quantification and modification call for a hierarchically structured representation, as illustrated by recent work (Gupta et al. 2018; Bonial et al. 2020; Cheng et al. 2020; Andreas et al. 2020). Commonly adopted tree- or directed acyclic graph-based structures resemble traditional frameworks for syntactic or semantic parsing of natural language sentences.

The hierarchical representation of Gupta et al. (2018) motivated the extension of neural shift-reduce parsers (Dyer et al. 2016; Einolghozati et al. 2019), neural span-based parsers (Stern, Andreas, and Klein 2017; Pasupat et al. 2019) and sequence-to-sequence (seq2seq) (Sutskever, Vinyals, and Le 2014; Vaswani et al. 2017; Rongali et al. 2020) models for handing *compositional* queries in task-oriented dialog. Due to the state-of-the-art performance of these models, there has been limited work designing structured prediction models that have a stronger inductive bias for semantic parsing of task-oriented dialog utterances.

In this paper, we propose the **R**ecursive **In**sertion-based **E**ncoder) (RINE; pronounced "Ryan"), that incrementally builds the semantic parse tree by inserting the non-terminal intent/slot labels into the utterance. The model is trained as a discriminative model that predicts labels with their corresponding positions in the input. At generation time, the model constructs the semantic parse tree by recursively inserting the predicted label at the predicting position until the termination. Unlike seq2seq models (Rongali et al. 2020; Zhu et al. 2020; Aghajanyan et al. 2020; Babu et al. 2021), our approach does not contain a separate decoder which generates the linearized semantic parse tree.

We extensively evaluate our proposed approach on low-resource and high-resource versions of the popular conversational semantic parsing dataset `TOP` (Gupta et al. 2018; Chen et al. 2020). We compare our model against a state-of-the-art transition-based parser RNNG (Gupta et al. 2018; Einolghozati et al. 2019) and seq2seq models (Rongali et al. 2020; Zhu et al. 2020; Aghajanyan et al. 2020; Babu et al. 2021) adapted to this task. We show that our approach achieves the state-of-the-art performance on both low-resource and high-resource settings `TOP`. In particular, RINE achieves **up to an** 13% **absolute improvement in exact match** in the low-resource setting. We also demonstrate that our approach is $2-3.5\times$ faster than strong sequence-to-sequence model at inference time.

While we focus on semantic parsing in task-oriented dialog, we demonstrate that our model design is applicable to other structured prediction tasks, such as nested named entity recognition (nested NER). We empirically show that our model with no specific tuning performs on par with state-of-the-art machine reading comprehension approach for nested NER (Li et al. 2020) that was explicitly designed for that task.

Proposed Approach

First, we introduce the problem of semantic parsing in task-oriented dialog and give a general description of our approach in Section . Then in Section , we give a detailed description of the forward pass and loss calculation in our model. Finally

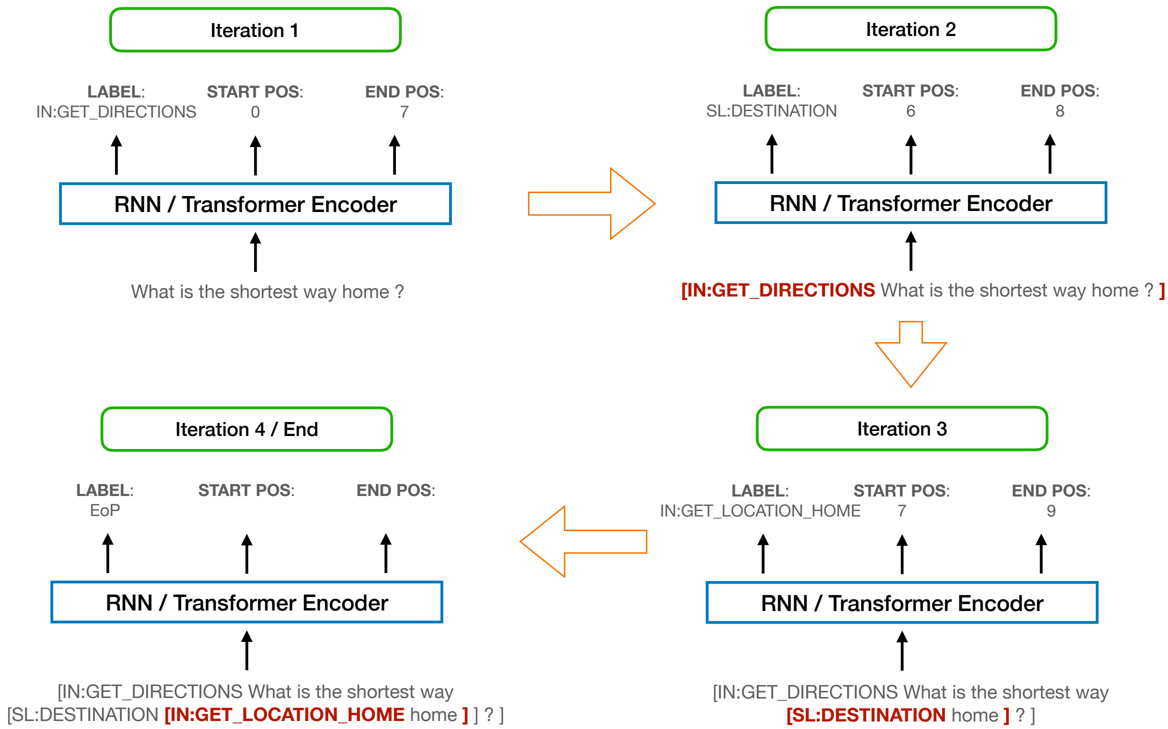

Figure 1: Overview of the top-down generation of the semantic parse tree corresponding to the utterance *What is the shortest way home?* from the TOP dataset (Gupta et al. 2018) using our proposed model. The inserted labels at each generation step are highlighted in red.

in Section we describe the generation procedure of semantic parse tree given the input utterance.

Overview

Given the utterance $X = (x_0, ..., x_{n-1})$ with n tokens, our goal is to predict the semantic parse tree Y. Each leaf node in the tree Y corresponds to a token $x_i \in X$, while each non-terminal node covers some span (i, j) with tokens $x_{i:j} = (x_i, ..., x_{j-1})$. The label l of each non-terminal node is either an intent (prefixed with `IN:`) or a slot (prefixed with `SL:`). The root node of the TOP tree covering the span $(0, n)$ must be an intent. Intents can be nested inside the slots and vice versa resulting in the composite tree structures. It should be noted that this formulation of semantic parse tree resembles the constituent structures commonly adopted in syntactic parsing. The key difference is that the non-terminal nodes in TOP tree are semantic entities in a dialog frame representation (i.e. intents and slots). Therefore they are not syntactic units and their corresponding leaf sub-sequences do not necessarily pass the constituency test. Instead, these non-terminal semantic nodes *govern* the linguistic expressions where the meaning is derived: slot nodes dominate over the string spans denoting their values; intent nodes dominate over both the span of utterance signaling the intent and slot nodes as arguments of each intent.

For our approach, it is helpful to view the target tree Y as the result of the incremental insertions of the elements in the set $S = \{(l_1, i_1, j_1), ..., (l_T, i_T, j_T)\}$ into the utterance X. The t^{th} element (l_t, i_t, j_t) in the set S consists of the intent/slot label l_t, the start position i_t and the end position j_t. The label l_t covers the span $(i_t, j_t - 1)$ in the partially build tree Y_{t-1}. The result of the sequence of consecutive insertions of all elements in set S is the target tree $Y_T = Y$. The utterance X is used as the input for the first insertion step.

You can see the example of the semantic tree generation using our model in Figure 1. At the first iteration, the label `IN:GET_DIRECTIONS` is inserted at the start position 0 and end position 7 into the utterance *What is the shortest way home ?*. The result of the insertion operation is the tree `[IN:GET_DIRECTIONS` *What is the shortest way home ?* `]`. This tree is fed back into the model to output the tuple (`SL:DESTINATION`, 6, 8). The updated tree with inserted label `SL:DESTINATION` is fed back into the model to output the (`IN:GET_LOCATION_HOME`, 7, 9). Finally, the model predicts a special end of prediction `EoP` label that indicates the termination of the generation process.

Training

For an input sequence $[w_1, w_2, ..., w_m]$ consisting of the tokens from utterance X and intent/slot labels from parse tree Y, our model first encodes the input into a sequence of hidden vectors $[e_1, e_2, ..., e_m]$. This model consists of an encoder that can have an RNN (Elman 1990), Transformer (Vaswani et al. 2017) or any other architecture. In practice, we use the pretrained Transformer model RoBERTa (Liu et al. 2019) due to its significant improvement in performance across natural language understanding tasks (Wang et al. 2018) and state-of-the-art performance on task-oriented semantic parsing (Rongali et al. 2020).

The hidden vector e_1 corresponding to the special start of the sentence symbol is passed to a multilayer perceptron (MLP) to predict the probability of the output label $l_t = \text{softmax}(\text{MLP}(e_1))$. We use the attention probabilities from the first attention head of the last layer's multi-head attention layer to predict the begin position i_t. Similarly, we use the attention probabilities from the second attention head of the last layer's multi-head attention layer to predict the end position j_t. This choice allows the model to extrapolate to start and end positions larger than the ones encountered during training.

After getting the outputs from the model, we train it by combining three objectives: *label loss* $\mathcal{L}_{\text{label}} = -\log p(l_t^*|Y_{t-1}^*)$, *start position loss* $\mathcal{L}_{\text{start}} = -\log p(i_t^*|Y_{t-1}^*)$ and *end position loss* $\mathcal{L}_{\text{end}} = -\log p(j_t^*|Y_{t-1}^*)$. As a result, we minimize the joint negative log likelihood of the the ground-truth labels (l_t^*, i_t^*, j_t^*) given the ground-truth partial tree Y_{t-1}^*:

$$\mathcal{L} = \mathcal{L}_{\text{label}} + \mathcal{L}_{\text{start}} + \mathcal{L}_{\text{end}}$$

During training, we batch the predictions across the generation time-steps $t = 1, ..., T$. We follow a top-down generation ordering to create the training set of pairs of partially constructed ground-truth trees Y_{t-1}^* and outputs (l_t^*, i_t^*, j_t^*)). When using top-down ordering, we first generate the root node and then work down the tree to generate remaining nodes. In principle, we can use other generation orderings (such as bottom-up ordering where we start from generating nodes at the lowest level of the tree and work up, which we empirically compare against in Section).

Generation

When evaluating the trained model, we use greedy decoding. We start from the input utterance X and predict the most likely label $\hat{l}_1 = \arg\max_l p(l_1|Y_0)$ and most likely start and end positions $\hat{i}_1 = \arg\max_i p(i_1|Y_0)$ and $\hat{j}_1 = \arg\max_j p(j_1|Y_0)$. We then insert the label $\hat{l}_1$ into position $\hat{i}_1$, followed by inserting the closing bracket into the position $\hat{j}_1 + 1$. We feed the resulting tree $\hat{Y}_1$ back into the model to predict the next triplet of ($\hat{l}_2$, $\hat{i}_2$, $\hat{j}_2$) following the same procedure. The entire process is repeated until the special end of prediction symbol `EoP` is predicted as the most likely label by the model.

Related Work

Parsing the meaning of utterances in task-oriented dialogue has been a prevalent problem in a research community since the advent of the `ATIS` dataset (Hemphill, Godfrey, and Doddington 1990). Traditionally, this task is formulated as a joint intent classification and slot tagging problem. Sequence labeling models based on recurrent neural networks (RNNs) (Mesnil et al. 2013; Liu and Lane 2016) and pre-trained Transformer models (Devlin et al. 2019; Chen, Zhuo, and Wang 2019) have been successfully used for joint intent classification and slot tagging. These sequence models can only parse *flat* utterances which contain a single intent class and single slot label per each token in the utterance. To deal with this limitation, recent studies investigated structured prediction models based on neural shift-reduce parsers (Dyer et al. 2016; Gupta et al. 2018; Einolghozati et al. 2019), neural span-based parsers (Stern, Andreas, and Klein 2017; Pasupat et al. 2019), autoregressive sequence-to-sequence models (Rongali et al. 2020; Aghajanyan et al. 2020), and non-autoregressive sequence-to-sequence models (Zhu et al. 2020; Shrivastava et al. 2021; Babu et al. 2021) for handling *compositional* queries. All of these approaches have been adapted from constituency parsing, dependency parsing and machine translation. Among these, the approach for task-oriented dialogue by Zhu et al. (2020) bears the most similarity to our model.

Zhu et al. (2020) adapted the Insertion Transformer (Stern et al. 2019) into the seq2seq-ptr model (Rongali et al. 2020) for conversational semantic parsing. The Insertion Transformer generates the linearized parse tree in balanced binary tree order by predicting labels in the insertion slots at each generation step (there are $T - 1$ insertion slots for sentence of length T). Unlike traditional seq2seq models which scale linearly with length of target sequence, the Insertion Transformer only requires a logarithmic number of decoding steps. Despite sharing the insertion operation, there are several key differences between our approach and the Insertion Transformer seq2seq-ptr approach illustrated in Figure 2. Unlike Insertion Transformer: 1) our model does not have separate decoder, 2) our model generates a parse tree in a top-down fashion with the number of decoding steps equivalent to the number of intent/slot labels in the tree. Additionally, the termination strategy in our model is as simple as termination in vanilla seq2seq models. To terminate generation the Insertion Transformer requires predicting `EoS` token for each insertion slot. This makes the `EoS` token more frequent than other tokens which leads to generation of short target sequences. To avoid this issue, Zhu et al. (2020) add a special penalty hyperparameter to control the sequence length.

In parallel to the design of neural architectures, there has been a research effort on improving neural conversational semantic parsers in low-resource setting using meta-learning (Chen et al. 2020) and label semantics (Athiwaratkun et al. 2020; Paolini et al. 2021; Desai et al. 2021). These approaches are architecture agnostic and can be easily combined with our model to further improve performance.

Experiments

Datasets

We use the `TOP` (Gupta et al. 2018) and `TOPv2` (Chen et al. 2020) conversational semantic parsing datasets as well as `ACE2005` nested named entity recognition dataset in our experiments.

The `TOP` dataset[1] (Gupta et al. 2018) consists of natural language utterances in two domains: *navigation* and *event*.

[1] http://fb.me/semanticparsingdialog

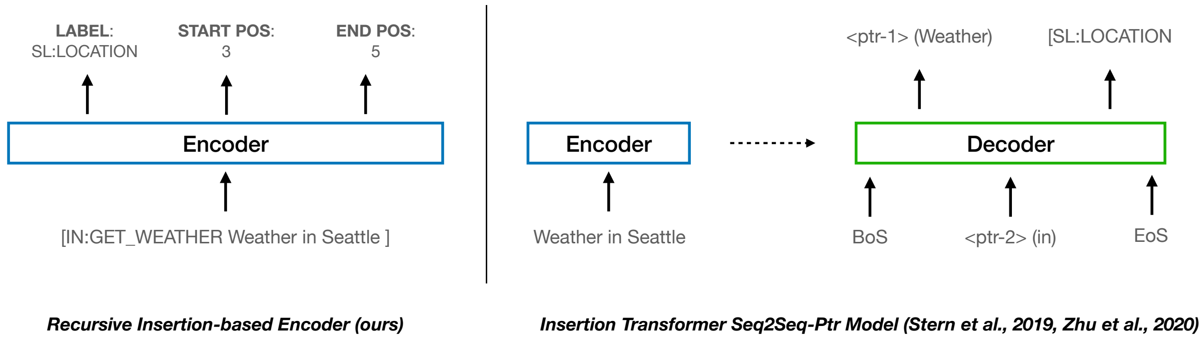

Figure 2: Side-by-side comparison of two closely related architectures for semantic parsing in task-oriented dialog. On the left, we show the forward pass of our model. On the right, we show the forward pass of Insertion Transformer Seq2seq-Ptr Model (Stern et al. 2019; Zhu et al. 2020). Tokens $<$ *ptr-i*$>$ denote the pointers to the utterance. The Insertion Transformer follows the balanced binary tree ordering by predicting labels for each insertion slot (there are $T-1$ insertion slots for sentence of length T). Our model follows the top-down generation ordering by predicting the single intent/slot label with start and end positions in the linearized tree.

The dataset consists of 25 intents and 36 slots. Following previous work (Einolghozati et al. 2019; Rongali et al. 2020; Zhu et al. 2020), we remove the utterances that contain the UNSUPPORTED intent from the dataset. This results in 28,414 train, 4,032 valid and 8,241 test utterances[2]. 39% of queries in the TOP dataset are hierarchical.

The TOPv2 dataset (Chen et al. 2020) is an extension of TOP dataset (Gupta et al. 2018) that was collected by following the same guidelines. Following the experimental setup of Chen et al. (2020) we use low-resource versions of *reminder* and *weather* domains. The *reminder* domain consists of 19 intents and 32 slots. 21% of queries in *reminder* domain are hierarchical. The *weather* domain consists of 7 intents and 11 slots. All queries in the *weather* domain are flat. The low-resource data was created by taking a fixed number of training *samples per intent and slot label (SPIS)* from the original dataset. If a particular intent or slot occurred less than the specified number of times then all the parse trees containing that intent or slot are selected. We use the same train, validation and test data at 25 and 500 SPIS for *reminder* and *weather* prepared by Chen et al. (2020)[3]. The *reminder* domain at 500 SPIS contains 4,788 train and 2,526 valid samples, *weather* 500 SPIS contains 2,372 train and 2,667 valid samples, *reminder* 25 SPIS contains 493 train and 337 valid samples, and *weather* 25 SPIS contains 176 train and 147 valid samples. For both SPIS settings the test splits of *reminder* and *weather* contain 5,767 and 5,682 test samples respectively.

The ACE2005 nested named entity recognition dataset is derived from the ACE2005 corpus (Walker et al. 2006) and consists of sentences from a variety of domains, including news and online forums. We use the same processing and splits of Li et al. (2020), resulting in 7,299 sentences for training, 971 for validation, and 1,060 for testing. The dataset has seven entity types: *location, organization, person, vehicle, geographical entity, weapon, facility*. 38% of queries in the ACE2005 dataset are hierarchical. The design of the semantic parse trees in ACE2005 dataset is similar to the design of semantic parse trees in TOP dataset. The entities in the ACE2005 dataset are represented as slots. Slots in ACE2005 can be nested inside the slots resulting in nested entity structures.

Following previous work (Gupta et al. 2018; Einolghozati et al. 2019; Rongali et al. 2020; Zhu et al. 2020; Aghajanyan et al. 2020) we use exact match (EM) accuracy as the metric for evaluating approaches on TOP and TOPv2 datasets. The exact match measures the number of utterances where complete trees are correctly predicted by the model. On ACE2005 we report the span-level micro-averaged precision, recall and F1 scores.

Hyperparameters

We follow the experimental settings of previous conversational semantic parsing work (Rongali et al. 2020; Zhu et al. 2020; Chen et al. 2020) and use a pre-trained RoBERTa (Liu et al. 2019) model as the backbone of our RINE model. We experiment with both RoBERTa$_{\text{BASE}}$ and RoBERTa$_{\text{LARGE}}$ architectures. The [CLS] representation is passed into the MLP with 1 hidden layer to predict the intent/slot label. The probabilities from the second and third heads of the last self-attention layer are used to predict start and end positions.

We train a sequence-to-sequence pointer network (seq2seq-ptr) that combines a Transformer (Vaswani et al. 2017) and pointer-generator network (See, Liu, and Manning 2017). Rongali et al. (2020) proposed this model for the task of conversational semantic parsing. The seq2seq-ptr model generates the linearized semantic parse tree by alternating between generating intent/slot tags from a fixed vocabulary and copying a token from the source query using a pointer network (Vinyals, Fortunato, and Jaitly 2015; See, Liu, and Manning 2017). The encoder of seq2seq-ptr is initialized using a pre-trained RoBERTa (Liu et al. 2019) architecture. The decoder is initialized with random weights. The decoder contains 6

[2]The dataset statistics were verified with authors of (Zhu et al. 2020)

[3]https://fb.me/TOPv2Dataset

layers, 4 attention heads, 512-dimensional embeddings, and $1{,}024$ hidden units.

We use the Adam optimizer (Kingma and Ba 2014) with the following hyperparameters: $\beta_1 = 0.9$, $\beta_2 = 0.98$, $\epsilon = 1e-6$ and L_2 weight decay of $1e-4$. When using RoBERTa$_{\text{BASE}}$, we warm-up the learning rate for 500 steps up to a peak value of $5e-4$ and then decay it based on the inverse square root of the update number. When using RoBERTa$_{\text{LARGE}}$, we warm-up the learning rate for $1{,}000$ steps up to a peak value of $1e-5$ and then decay it based on the inverse number of update steps. The same hyperparameters of the optimizer are used for training both RINE and seq2seq models. We use a dropout (Srivastava et al. 2014) rate of 0.3 and an attention dropout rate of 0.1 in both our proposed models and seq2seq baseline. The choices of hyperparameters were made based on preliminary experiments on `TOP`. For all datasets we use 4 Tesla V100 GPUs to train both baseline and proposed model. We use 3 random seeds to train all models and report the average and standard deviation. For fair comparison of results, we train both baseline and proposed model for the same number of iterations on all datasets. We implement all models on top of the *fairseq* framework (Ott et al. 2019).

Results

TOP dataset We present the results on `TOP` (Gupta et al. 2018) in Table 1. Our proposed RINE model that uses RoBERTa$_{\text{LARGE}}$ as the backbone outperforms all previously published results on the `TOP` dataset. In particular, our RINE model initialized with the RoBERTa$_{\text{LARGE}}$ outperforms the non-autoregressive seq2seq-ptr model (Shrivastava et al. 2021) initialized with RoBERTa$_{\text{BASE}}$ by 2.5 EM, autoregressive seq2seq-ptr models (Rongali et al. 2020; Zhu et al. 2020) initialized with RoBERTa$_{\text{BASE}}$ by 0.9 EM, decoupled seq2seq-ptr model initialized with BART$_{\text{LARGE}}$ by 0.47 EM, and RNNG ensemble with SVM reranking (Einolghozati et al. 2019) by 0.32 EM. Our model initialized with the RoBERTa$_{\text{BASE}}$ outperforms the non-autoregressive seq2seq-ptr model (Shrivastava et al. 2021) initialized with RoBERTa$_{\text{BASE}}$ by 2.0 EM, autoregressive seq2seq-ptr models (Rongali et al. 2020; Zhu et al. 2020) initialized with RoBERTa$_{\text{BASE}}$ by 0.4 EM, and performs on par with RNNG ensemble with SVM reranking (Einolghozati et al. 2019) and decoupled seq2seq-ptr (Aghajanyan et al. 2020) initialized with BART$_{\text{LARGE}}$. Unlike RNNG, we do not use ensembling and do not rerank outputs of our model. Unlike decoupled seq2seq-ptr, we don't use stochastic weight averaging (Izmailov et al. 2018) to improve results.

We also re-implemented and trained our seq2seq-ptr model by Rongali et al. (2020). We obtain exact match of 85.73 ± 0.21 and 86.67 ± 0.21 with RoBERTa$_{\text{BASE}}$ and RoBERTa$_{\text{LARGE}}$ backbones respectively. Compared to our replication of autoregressive seq2seq-ptr model by Rongali et al. (2020), the proposed RINE model achieves 1.41 and 0.9 improvement in exact match using RoBERTa$_{\text{BASE}}$ and RoBERTa$_{\text{LARGE}}$ respectively. The performance of RINE is $5-7$ standard deviations higher than performance of our seq2seq-ptr model.

The decomposition of the parse tree leads to multiple training passes over the same source sentence in our model. This is unlike seq2seq-ptr model that processes each pair of source and target once during the epoch. One could argue that increasing the number of training iterations in baseline seq2seq-ptr approach can lead to the same performance as our model. However despite training baseline seq2seq-ptr for the same iterations as RINE, seq2seq-ptr stops improving validation exact match after 150 epochs and starts overfitting after.

TOPv2 dataset We present the results on low-resource versions of *reminder* and *weather* domains in `TOPv2` (Chen et al. 2020) in Table 2. There are several observations on this dataset.

First, our proposed RINE model outperforms the baseline autoregressive seq2seq-ptr model on all evaluated scenarios of this dataset. In the 500 SPIS setting with RoBERTa$_{\text{BASE}}$, our RINE model achieves 8.4 and 2.9 exact match improvement on *reminder* and *weather* domains over the best published seq2seq-ptr baseline. In the 25 SPIS setting with RoBERTa$_{\text{BASE}}$, our RINE model achieves 13.0 and 2.9 exact match improvement on *reminder* and *weather* domains over the best published autoregressive seq2seq-ptr baseline. In the *reminder* domain the improvement in performance is higher for the 25 SPIS setting, whereas in the *weather* domain the improvement in performance is comparable for both 25 and 500 SPIS. We hypothesize that this is due to the *reminder* domain being more challenging that *weather* domain since it contains composite utterances and have a larger number of intent and slot types.

Second, we trained our re-implementation of seq2seq-ptr model by Rongali et al. (2020) on 500 SPIS setting. We find that our replication of seq2seq-ptr approach with RoBERTa$_{\text{BASE}}$ backbone achieves 78.46 ± 0.3 and 85.41 ± 0.04 exact match on *reminder* and *weather* domains. The performance of our replication of seq2seq-ptr model with RoBERTa$_{\text{BASE}}$ performs better than seq2seq-ptr with RoBERTa$_{\text{BASE}}$ reported by Chen et al. (2020). We believe it is due to the number of epochs we used to train seq2seq-ptr models. Chen et al. (2020) report using 100 epochs to train all models without meta-learning, whereas we train seq2seq-ptr for larger number of epochs to match the number of iterations used to train RINE. Despite training our seq2seq-ptr for larger number of iterations, RINE outperforms our replication of seq2seq-ptr approach by 1.84 and 2.39 exact match on 500 SPIS setting of the *reminder* and *weather* domains respectively. We make a similar observation to `TOP` dataset and find that seq2seq-ptr approach underperforms RINE despite increasing number of training iterations to match the number of training iterations of RINE. In particular, seq2seq-ptr converges to the highest validation exact match after 300 epochs and starts overfitting after.

ACE2005 We present the results on `ACE2005` dataset in Table 3. Our model outperforms all previously published approaches designed for nested named entity recognition task, except for BERT-MRC (Li et al. 2020) approach. In particular, RINE with RoBERTa$_{\text{LARGE}}$ encoder underperforms BERT-MRC with BERT$_{\text{LARGE}}$ encoder in terms of precision, while achieving higher recall and comparable F1 score. Compared to our model, BERT-MRC uses questions constructed from

Method	Pretrained model	Exact Match
RNNG (Einolghozati et al. 2019)	-	80.86
RNNG (Einolghozati et al. 2019)	ELMo	86.26
RNNG ensemble + SVMRank (Einolghozati et al. 2019)	ELMo	87.25
Non-AR Seq2seq-Ptr (Shrivastava et al. 2021)	$\text{RoBERTa}_{\text{BASE}}$	85.07
Seq2seq-Ptr (Rongali et al. 2020)	$\text{RoBERTa}_{\text{BASE}}$	86.67
Insertion Transformer + Seq2seq-Ptr (Zhu et al. 2020)	$\text{RoBERTa}_{\text{BASE}}$	86.74
Decoupled Seq2seq-Ptr (Aghajanyan et al. 2020)	$\text{BART}_{\text{LARGE}}$	87.10
RINE (ours)	$\text{RoBERTa}_{\text{BASE}}$	87.14±0.06
RINE (ours)	$\text{RoBERTa}_{\text{LARGE}}$	**87.57±0.03**

Table 1: Accuracy (exact match ↑) on the test split of `TOP` dataset (Gupta et al. 2018). Non-AR stands for non-autoregressive. Pretrained model stands for the type of pretrained architecture used in the corresponding method.

Method	Pretrained model	Exact Match			
		Reminder		Weather	
		25 SPIS	500 SPIS	25 SPIS	500 SPIS
LSTM Seq2Seq-Ptr (Chen et al. 2020)	-	21.5	65.9	46.2	78.6
Seq2seq-Ptr (Chen et al. 2020)	$\text{RoBERTa}_{\text{BASE}}$	-	71.9	-	83.5
Seq2seq-Ptr (Chen et al. 2020)	$\text{BART}_{\text{LARGE}}$	55.7	71.9	71.6	84.9
RINE (ours)	$\text{RoBERTa}_{\text{BASE}}$	68.71±0.46	80.30±0.04	74.53±0.86	**87.80±0.04**
RINE (ours)	$\text{RoBERTa}_{\text{LARGE}}$	**71.10±0.63**	**81.31±0.22**	**77.03±0.16**	87.50±0.28

Table 2: Accuracy (exact match ↑) on the test split of *reminder* and *weather* domains of `TOPv2` dataset (Chen et al. 2020). SPIS stands for *samples for each intent and slot label*.

Method	Pretrained Model	Precision	Recall	F1
Hyper-Graph LSTM (Katiyar and Cardie 2018)	-	70.6	70.4	70.5
Seg-Graph (Wang and Lu 2018)	GLoVE	76.8	72.3	74.5
ARN (Lin et al. 2019)	GLoVE	76.2	73.6	74.9
Path-BERT (Shibuya and Hovy 2019)	$\text{BERT}_{\text{LARGE}}$	82.98	82.42	82.7
Merge-BERT (Fisher and Vlachos 2019)	$\text{BERT}_{\text{LARGE}}$	82.7	82.1	82.4
DYGIE (Luan et al. 2019)	GLoVE + ELMo	-	-	82.9
Seq2seq-BERT (Straková, Straka, and Hajic 2019)	$\text{BERT}_{\text{LARGE}}$	-	-	84.33
TANL (Paolini et al. 2021)	T5_{BASE}	-	-	84.9
BERT-MRC (Li et al. 2020)	$\text{BERT}_{\text{LARGE}}$	**87.16**	86.59	**86.88**
RINE (ours)	$\text{RoBERTa}_{\text{BASE}}$	84.13±0.03	87.06±0.19	85.57±0.1
RINE (ours)	$\text{RoBERTa}_{\text{LARGE}}$	84.62±0.05	**88.33±0.07**	86.44±0.04

Table 3: Precision (↑), recall (↑) and F1 score (↑) on the test split of `ACE2005` dataset.

the annotation guideline notes used for collecting ACE2005 dataset. These questions contain the ground-truth label semantics (example for ORG label the question is ”find organizations including companies, agencies and institutions“). Li et al. (2020) show that label semantics improve results by 1.5 F1 (Table 5 of BERT-MRC (Li et al. 2020)) and use scores obtained with label semantics in the main results. We believe using some form of label semantics in the output can further improve RINE on ACE2005. Despite the lack of label semantics in our approach and no additional tuning, RINE achieves comparable performance to state-of-the-art BERT-MRC which further demonstrates the strong performance of our model.

Analysis

Does Generation Order Matter?

In this section we analyze whether generation order matters for our model. In our default setting, we follow the top-down ordering of the labels when training and generating trees. We experiment with a bottom-up ordering and present the comparison with top-down ordering in Table 4. We find that the choice of ordering makes no difference in exact match accuracy on the validation split of `TOP` suggesting that the model is agnostic to the particular order in which it was trained.

Architecture	Order	Exact Match
RoBERTa$_{\text{BASE}}$	Top-down	87.07±0.09
	Bottom-up	87.01±0.04
RoBERTa$_{\text{LARGE}}$	Top-down	87.72±0.07
	Bottom-up	87.70±0.04

Table 4: Exact match of proposed RINE model on validation split of `TOP` dataset with top-down and bottom-up generation orderings.

Flat vs Composite Queries

Our initial motivation of the proposed approach stems from the argument that the seq2seq-ptr models are not ideally suited for parsing the utterance into hierarchically structured representations. In this section, we empirically validate this argument by breaking down the performance of both models on flat and hierarchical trees. We find that a larger improvement of our approach over the baseline model comes from composite trees. In particular, on the validation split of `TOP`, the RINE model achieves 3.4% relative improvement on the composite queries over the seq2seq-ptr model. On the same dataset, the RINE model achieves 1.6% improvement on the flat queries over the seq2seq-ptr model. We make a similar empirical observation on the validation split of the *reminder* domain on the `TOPv2` dataset. In the 25 SPIS setting, the proposed model achieves a 22.4% relative improvement on composite queries, while it achieves only a 6% relative improvement on flat queries. In the 500 SPIS setting, the proposed model achieves a 3.5% and 1.3% relative improvement on composite and flat queries. The relative improvement in performance on composite queries becomes larger in the low-resource 25 SPIS setting. This shows that the proposed approach is better suited for hierarchically structured meaning representations compared to the seq2seq-ptr model.

Validity of Generated Trees

In this section we compare the validity of the semantic parse trees generated by our and baseline approaches. Unlike our approach which generates perfectly valid trees when trained on both low- and high-resource settings of `TOP` dataset, seq2seq-ptr struggles to achieve perfect validity when trained in the low-resource setting. In particular in the 25 SPIS setting of the *reminder* domain, 93% of the generated trees by seq2seq-ptr model are valid. When we increase the training dataset size and train the seq2seq-ptr model on the 500 SPIS setting of the *reminder* domain, the validity of generated trees becomes close to 100%. This demonstrates that baseline seq2seq-ptr model requires larger amount of training data to learn the structure of semantic parse trees in order to generate valid trees.

Generation Efficiency

In this section we compare the generation latency of the seq2seq-ptr and RINE approaches. We generate parse trees by processing 1 sentence at a time using Tesla V100 GPU. As the measure of generation efficiency we use number of sentences per second (↑) processed by each approach. We show

Architecture	Seq2seq-ptr	RINE
RoBERTa$_{\text{BASE}}$	3.70	**12.95**
RoBERTa$_{\text{LARGE}}$	3.42	**7.09**

Table 5: Generation efficiency (sentences per second ↑) of seq2seq-ptr and RINE approaches with RoBERTa$_{\text{BASE}}$ and RoBERTa$_{\text{LARGE}}$ architectures on validation split of `TOP` (Gupta et al. 2018) dataset.

results in Table 5. We find that our approach is $3.5\times$ faster with RoBERTa$_{\text{BASE}}$ and $2\times$ faster with RoBERTa$_{\text{LARGE}}$ than seq2seq-ptr approach. We notice that the decoding efficiency of our approach relative to baseline drops when using large encoder. Seq2seq-ptr scales better with larger encoders due to caching of the encoder representations that are later used by the decoder.

Conclusions and Future Work

Following on the exciting and recent development of hierarchically structured meaning representations for task-oriented dialog, we proposed a recursive insertion-based encoder approach that achieves state-of-the-art results on low- and high-resource versions of the conversational semantic parsing benchmark `TOP` (Gupta et al. 2018; Chen et al. 2020). We also showed that the proposed model design is applicable to nested NER, where it achieves comparable results to state-of-the-art with no additional tuning. Analysis of the results demonstrates that proposed approach achieves higher relative improvement on hierarchical trees compared to baselines and does not require large amount of training data to learn the structure of the trees.

Despite achieving strong empirical results, there are some limitations with the proposed approach. The insertion-based approach is generally limited to generate *anchored* and well-nested tree structures that are addition to the input with no deletion or reordering. While such assumption is commonly adopted in many linguistic representations, well-known exceptions do exist (e.g. non-projective dependencies (Hall and Nivre 2008), discontiguous constituents (Vijay-Shanker, Weir, and Joshi 1987; Müller 2004), or unanchored meaning representations such as AMR (Banarescu et al. 2013)). While the authors of `TOP` (Gupta et al. 2018) found that a very small fraction of English queries (0.3%) require a more general unanchored graph-based meaning representation, we believe it is important to address such issues for non-configurational languages as well as the broader range of application-specific structural representations. We plan to extend the model with additional actions such as token insertion and swap in order to support parsing of such non-anchored representation in multilingual semantic parsing.

Overall, we hope that our paper inspires research community to further extend and apply our model for a variety of structured prediction tasks.

Acknowledgements

We would like to thank Daniele Bonadiman, Arshit Gupta, James Gung, Yassine Benajiba, Haidar Khan, Saleh Soltan and other members of Amazon AWS AI for helpful suggestions.

References

Aghajanyan, A.; Maillard, J.; Shrivastava, A.; Diedrick, K.; Haeger, M.; Li, H.; Mehdad, Y.; Stoyanov, V.; Kumar, A.; Lewis, M.; and Gupta, S. 2020. Conversational Semantic Parsing. *ArXiv*, abs/2009.13655.

Andreas, J.; Bufe, J.; Burkett, D.; Chen, C.; Clausman, J.; Crawford, J.; Crim, K.; DeLoach, J.; Dorner, L.; Eisner, J.; Fang, H.; Guo, A.; Hall, D.; Hayes, K.; Hill, K.; Ho, D.; Iwaszuk, W.; Jha, S.; Klein, D.; Krishnamurthy, J.; Lanman, T.; Liang, P.; Lin, C. H.; Lintsbakh, I.; McGovern, A.; Nisnevich, A.; Pauls, A.; Petters, D.; Read, B.; Roth, D.; Roy, S.; Rusak, J.; Short, B.; Slomin, D.; Snyder, B.; Striplin, S.; Su, Y.; Tellman, Z.; Thomson, S.; Vorobev, A.; Witoszko, I.; Wolfe, J.; Wray, A.; Zhang, Y.; and Zotov, A. 2020. Task-Oriented Dialogue as Dataflow Synthesis. *Transactions of the Association for Computational Linguistics*, 8: 556–571.

Athiwaratkun, B.; Santos, C. D.; Krone, J.; and Xiang, B. 2020. Augmented Natural Language for Generative Sequence Labeling. In *EMNLP*.

Babu, A.; Shrivastava, A.; Aghajanyan, A.; Aly, A.; Fan, A.; and Ghazvininejad, M. 2021. Non-Autoregressive Semantic Parsing for Compositional Task-Oriented Dialog. In *NAACL*.

Banarescu, L.; Bonial, C.; Cai, S.; Georgescu, M.; Griffitt, K.; Hermjakob, U.; Knight, K.; Koehn, P.; Palmer, M.; and Schneider, N. 2013. Abstract Meaning Representation for Sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, 178–186. Sofia, Bulgaria: Association for Computational Linguistics.

Bonial, C.; Donatelli, L.; Abrams, M.; Lukin, S. M.; Tratz, S.; Marge, M.; Artstein, R.; Traum, D.; and Voss, C. 2020. Dialogue-AMR: Abstract Meaning Representation for Dialogue. In *Proceedings of the 12th Language Resources and Evaluation Conference*, 684–695. ISBN 979-10-95546-34-4.

Chen, Q.; Zhuo, Z.; and Wang, W. 2019. BERT for Joint Intent Classification and Slot Filling. *ArXiv*, abs/1902.10909.

Chen, X.; Ghoshal, A.; Mehdad, Y.; Zettlemoyer, L.; and Gupta, S. 2020. Low-Resource Domain Adaptation for Compositional Task-Oriented Semantic Parsing. In *EMNLP*.

Cheng, J.; Agrawal, D.; Alonso, H. M.; Bhargava, S.; Driesen, J.; Flego, F.; Kaplan, D.; Kartsaklis, D.; Li, L.; Piraviperumal, D.; Williams, J.; Yu, H.; Séaghdha, D. Ó.; and Johannsen, A. 2020. Conversational Semantic Parsing for Dialog State Tracking. In *EMNLP*.

Desai, S.; Shrivastava, A.; Zotov, A.; and Aly, A. 2021. Low-Resource Task-Oriented Semantic Parsing via Intrinsic Modeling. *ArXiv*, abs/2104.07224.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT*.

Dyer, C.; Kuncoro, A.; Ballesteros, M.; and Smith, N. A. 2016. Recurrent Neural Network Grammars. In *HLT-NAACL*.

Einolghozati, A.; Pasupat, P.; Gupta, S.; Shah, R.; Mohit, M.; Lewis, M.; and Zettlemoyer, L. 2019. Improving Semantic Parsing for Task Oriented Dialog. *ArXiv*, abs/1902.06000.

Elman, J. 1990. Finding Structure in Time. *Cogn. Sci.*, 14: 179–211.

Fisher, J.; and Vlachos, A. 2019. Merge and Label: A novel neural network architecture for nested NER. *ArXiv*, abs/1907.00464.

Gupta, S.; Shah, R.; Mohit, M.; Kumar, A.; and Lewis, M. 2018. Semantic Parsing for Task Oriented Dialog using Hierarchical Representations. In *EMNLP*.

Hall, J.; and Nivre, J. 2008. Parsing Discontinuous Phrase Structure with Grammatical Functions. In Nordström, B.; and Ranta, A., eds., *Advances in Natural Language Processing*, 169–180. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN 978-3-540-85287-2.

Hemphill, C. T.; Godfrey, J. J.; and Doddington, G. 1990. The ATIS Spoken Language Systems Pilot Corpus. In *HLT*.

Izmailov, P.; Podoprikhin, D.; Garipov, T.; Vetrov, D.; and Wilson, A. 2018. Averaging Weights Leads to Wider Optima and Better Generalization. *ArXiv*, abs/1803.05407.

Katiyar, A.; and Cardie, C. 2018. Nested Named Entity Recognition Revisited. In *NAACL-HLT*.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Li, X.; Feng, J.; Meng, Y.; Han, Q.; Wu, F.; and Li, J. 2020. A Unified MRC Framework for Named Entity Recognition. In *ACL*.

Lin, H.; Lu, Y.; Han, X.; and Sun, L. 2019. Sequence-to-Nuggets: Nested Entity Mention Detection via Anchor-Region Networks. In *ACL*.

Liu, B.; and Lane, I. 2016. Attention-Based Recurrent Neural Network Models for Joint Intent Detection and Slot Filling. In *INTERSPEECH*.

Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; and Stoyanov, V. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *ArXiv*, abs/1907.11692.

Luan, Y.; Wadden, D.; He, L.; Shah, A.; Ostendorf, M.; and Hajishirzi, H. 2019. A General Framework for Information Extraction using Dynamic Span Graphs. *ArXiv*, abs/1904.03296.

Mesnil, G.; He, X.; Deng, L.; and Bengio, Y. 2013. Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding. In *INTERSPEECH*.

Müller, S. 2004. Continuous or Discontinuous Constituents? A Comparison between Syntactic Analyses for Constituent Order and Their Processing Systems. *Research on Language and Computation*, 2: 209–257.

Ott, M.; Edunov, S.; Baevski, A.; Fan, A.; Gross, S.; Ng, N.; Grangier, D.; and Auli, M. 2019. fairseq: A Fast, Extensible Toolkit for Sequence Modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

Paolini, G.; Athiwaratkun, B.; Krone, J.; Ma, J.; Achille, A.; Anubhai, R.; Santos, C. D.; Xiang, B.; and Soatto, S. 2021. Structured Prediction as Translation between Augmented Natural Languages. *ArXiv*, abs/2101.05779.

Pasupat, P.; Gupta, S.; Mandyam, K.; Shah, R.; Lewis, M.; and Zettlemoyer, L. 2019. Span-based Hierarchical Semantic Parsing for Task-Oriented Dialog. In *EMNLP/IJCNLP*.

Rongali, S.; Soldaini, L.; Monti, E.; and Hamza, W. 2020. Don't Parse, Generate! A Sequence to Sequence Architecture for Task-Oriented Semantic Parsing. *Proceedings of The Web Conference 2020*.

See, A.; Liu, P. J.; and Manning, C. D. 2017. Get To The Point: Summarization with Pointer-Generator Networks. In *ACL*.

Shibuya, T.; and Hovy, E. 2019. Nested Named Entity Recognition via Second-best Sequence Learning and Decoding. *Transactions of the Association for Computational Linguistics*, 8: 605–620.

Shrivastava, A.; Chuang, P.-H.; Babu, A.; Desai, S.; Arora, A.; Zotov, A.; and Aly, A. 2021. Span Pointer Networks for Non-Autoregressive Task-Oriented Semantic Parsing. In *EMNLP*.

Srivastava, N.; Hinton, G. E.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15: 1929–1958.

Stern, M.; Andreas, J.; and Klein, D. 2017. A Minimal Span-Based Neural Constituency Parser. *ArXiv*, abs/1705.03919.

Stern, M.; Chan, W.; Kiros, J.; and Uszkoreit, J. 2019. Insertion Transformer: Flexible Sequence Generation via Insertion Operations. In *ICML*.

Straková, J.; Straka, M.; and Hajic, J. 2019. Neural Architectures for Nested NER through Linearization. In *ACL*.

Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to Sequence Learning with Neural Networks. In *NIPS*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *NIPS*.

Vijay-Shanker, K.; Weir, D. J.; and Joshi, A. K. 1987. Characterizing Structural Descriptions Produced by Various Grammatical Formalisms. In *25th Annual Meeting of the Association for Computational Linguistics*, 104–111. Stanford, California, USA: Association for Computational Linguistics.

Vinyals, O.; Fortunato, M.; and Jaitly, N. 2015. Pointer Networks. In *NIPS*.

Walker, C.; Strassel, S.; Medero, J.; and Maeda, K. 2006. Ace 2005 multilingual training corpus. *Linguistic Data Consortium, Philadelphia*.

Wang, A.; Singh, A.; Michael, J.; Hill, F.; Levy, O.; and Bowman, S. R. 2018. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *BlackboxNLP@EMNLP*.

Wang, B.; and Lu, W. 2018. Neural Segmental Hypergraphs for Overlapping Mention Recognition. In *EMNLP*.

Zhu, Q.; Khan, H.; Soltan, S.; Rawls, S.; and Hamza, W. 2020. Don't Parse, Insert: Multilingual Semantic Parsing with Insertion Based Decoding. *ArXiv*, abs/2010.03714.