Don't Parse, Insert: Multilingual Semantic Parsing with Insertion Based Decoding

Qile Zhu[1*], **Haidar Khan**[2], **Saleh Soltan**[2], **Stephen Rawls**[2], **and Wael Hamza**[2]
[1]University of Florida, [2]Amazon Alexa AI
valder@ufl.edu
{khhaida,ssoltan,sterawls,waelhamz}@amazon.com

Abstract

Semantic parsing is one of the key components of natural language understanding systems. A successful parse transforms an input utterance to an action that is easily understood by the system. Many algorithms have been proposed to solve this problem, from conventional rule-based or statistical slot-filling systems to shift-reduce based neural parsers. For complex parsing tasks, the state-of-the-art method is based on autoregressive sequence to sequence models to generate the parse directly. This model is slow at inference time, generating parses in $O(n)$ decoding steps (n is the length of the target sequence). In addition, we demonstrate that this method performs poorly in zero-shot cross-lingual transfer learning settings. In this paper, we propose a non-autoregressive parser which is based on the insertion transformer to overcome these two issues. Our approach 1) speeds up decoding by 3x while outperforming the autoregressive model and 2) significantly improves cross-lingual transfer in the low-resource setting by 37% compared to autoregressive baseline. We test our approach on three well-known monolingual datasets: ATIS, SNIPS and TOP. For cross lingual semantic parsing, we use the MultiATIS++ and the multilingual TOP datasets.

1 Introduction

Given a query, a semantic parsing module identifies not only the *intent* (play music, book a flight) of the query but also extracts necessary *slots* (entities) that further refines the action to perform (which song to play? Where or when to go?). A traditional rule-based or slot-filling system classifies a query with one intent and tags each input token (Mesnil et al., 2013). However, supporting more complex queries that are composed of multiple intents and nested slots is a challenging problem (Gupta et al., 2018). Gupta et al. (2018) and Einolghozati et al. (2019) propose to use a Shift-Reduce parser based on Recurrent Neural Network for these complex queries. Recently, Rongali et al. (2020) propose directly generating the parse as a formatted sequence and design a unified model based on sequence to sequence generation and pointer networks. Their approach formulates the tagging problem into a generation task in which the target is constructed by combining all the necessary intents and slots in a flat sequence with no restriction on the semantic parse schema.

A relatively unexplored direction is the cross-lingual transfer problem (Duong et al., 2017; Susanto and Lu, 2017), where the parsing system is trained in a high-resource language and transfered directly to a low-resource language (zero-shot).

The state-of-the-art model leverages the autoregressive decoder such as Transformer (Vaswani et al., 2017) and Long-Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) to generate the target sequence (representing the parse) from left to right. The left to right autoregressive generation constraint has two drawbacks: 1) generating a parse takes $O(n)$ decoding time, where n is the length of the target sequence. This is further exacerbated when paired with standard search algorithms such as beam search. 2) In the cross-lingual setting, autoregressive parsers have difficulty transferring between languages.

A recent direction in machine translation and natural language generation to speed up sequence to sequence models is non-autoregressive decoding (Stern et al., 2019; Gu et al., 2018, 2019). Since the parsing task in the sequence to sequence framework only requires inserting tags rather than generating the whole sequence, an insertion based parser is both faster and more natural for language transfer than an autoregressive parser.

In this paper, we leverage insertion based se-

[*] Work done while interning at Amazon Alexa

Proceedings of the 24th Conference on Computational Natural Language Learning, pages 496–506
Online, November 19-20, 2020.
https://doi.org/10.18653/v1/P17

quence to sequence models for the semantic parsing problem that require only O(log(n)) decoding time to generate a parse. We enhance the insertion transformer (Stern et al., 2019) with the pointer mechanism, since the entities in the source sequence are ensured to appear in the target sequence. Our non-autoregressive based model can also boost the performance on the zero-shot and few-shot cross-lingual setting, in which the model is trained on a high-resource language and tested on low-resource languages. We also introduce a copy source mechanism for the decoder to further improve the cross lingual transfer performance. In this way, the pointer embedding will be replaced by the corresponding outputs from the encoder. We test our proposed model on several well known datasets, TOP (Gupta et al., 2018), ATIS (Price, 1990), SNIPS (Coucke et al., 2018), MultiATIS++ (Xu et al., 2020) and multilingual TOP (Xia and Monti, 2021).

In summary, the main contributions of our work include:

- To our knowledge, we are the first to apply the non-autoregressive framework to the semantic parsing task. Experiments show that our approach can reduce the decoding steps by 66.7%. By starting generation with the whole source sequence, we can further reduce the number of decoding steps by 82.4%.

- We achieve new state-of-the-art Exact Match (EM) scores on ATIS (89.14), SNIPS (91.00) and TOP (86.74, single model) datasets.

- We introduce a copy encoder outputs mechanism and achieve a significant improvement compared to the autoregressive decoder and sequence labeling on the zero-shot and few-shot setting in cross lingual transfer semantic parsing. Our approach surpasses the autoregressive baseline by 9 EM points on average over both simple (MultiATIS++) and complex (multilingual TOP) queries and matches the performance of the sequence labeling baseline on MultiATIS++.

2 Background

In this section, we introduce the sequence generation via insertion operations and the pretrained models we leverage in our work.

2.1 Sequence Generation Via Insertion

We begin by briefly describing sequence generation via insertion, for a more complete description see (Stern et al., 2019).

Let $x_1, x_2, ..., x_m$ be the source sequence with length m and $y_1, y_2, ..., y_n$ denotes the target sequence with length n. We define the generated sequence h_t at decoding step t. In the autoregressive setting, $h_t = y_{1,2,...,t-1}$. In insertion based decoding, h_t is a subsequence of the target sequence y that preserves order. For example, if the final sequence $y = [A, B, C, D, E]$, then $h_t = [B, E]$ is a valid intermediate subsequence while $h_t = [C, A]$ is an invalid intermediate subsequence.

During decoding step $t + 1$, we insert tokens into h_t. In the previous example, there are three available insertion slots: before token B, between B and E and after E. We always add special tokens such as bos (begin of the sequence) and eos (end of the sequence) to the subsequences. The number of available insertion slots will be $T - 1$ where T is the length of h_t including bos and eos. If we insert one token in all available slots, multiple tokens can be generated in one time step.

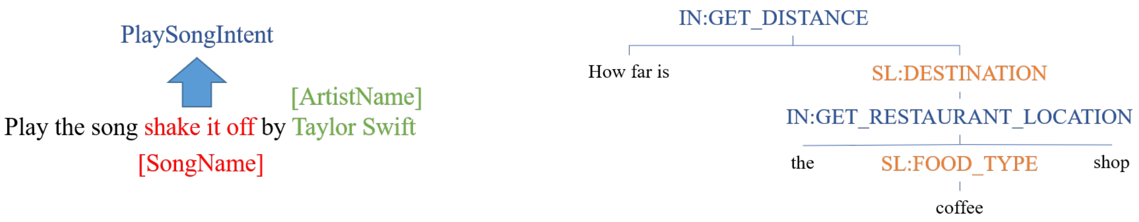
In order to predict the token to insert in a slot, we form the representation for each insertion slot by pooling the representations of adjacent tokens. We have $T - 1$ slots for a sequence with length T. Let $r \in \mathbb{R}^{T \times d}$, where T is the sequence length and d denotes the hidden size of the transformer decoder layer. All slots $s \in \mathbb{R}^{(T-1) \times d}$ can be computed as:

$$s = concat(r[1:], r[:-1]) \cdot W_s, \qquad (1)$$

where $r[1:]$ is the entire sequence representation excluding the first token, $r[:-1]$ is the entire sequence representation excluding the last token and $W_s \in \mathbb{R}^{2d \times d}$ is a trainable projection matrix. We apply $softmax$ to the slot representations to obtain the token probabilities to insert at each slot.

2.2 Pretrained Models

Pretrained language models (Devlin et al., 2019; Liu et al., 2019; Lan et al., 2020; Dong et al., 2019; Peters et al., 2018) have sparked significant progress in a wide variety of natural language processing tasks. The basic idea of these models is to leverage the knowledge from large-scale corpora by using a language modeling objective to learn a representation for tokens and sentences. For downstream tasks, the learned representations are



Left: [IN:PS @0 @1 @2 [SL:SName @3 @4 @5 SL:SName] @6 [SL:AName @7 @8 SL:ArName] IN:PS]
Right: [IN:GD @0 @1 @2 [SL:DEST [IN:GRL @3 [SL:FT @4 SL:FT] @5 IN:GRL] SL:DEST] IN:GD]

Figure 1: Example of a simple query (left) and complex query (right). The complex query contains multiple intents and nested slots and can be represented as a tree structure. The two queries are represented as formatted sequences that are treated as the target sequence in the parsing task. IN is the intent, SL is the slot. Source tokens that appear in the target sequence are replaced by pointers with the form @*n* where *n* denotes its location in the source sequence. For complex queries, we can build the parse from top to bottom and left to right.

fine-tuned for the task. This improvement is even more significant when the downstream task has few labeled examples.

We also follow this trend, and use the Transformer (Vaswani et al., 2017) based pretrained language model. We use the RoBERTa base (Liu et al., 2019) (we refer to this model as RoBERTa) as our query encoder to fairly compare with the previous method. This model has the same architecture as BERT base (Devlin et al., 2019) with several modifications during pretraining. It uses a dynamic masking scheme and removes the next sentence prediction task. RoBERTa is also trained with longer sentences and larger batch sizes with more training samples. For the multilingual zero-shot and few-shot semantic parsing task, we use XLM-R (Conneau et al., 2020) and multilingual BERT (Devlin et al., 2019) which are trained on text for more than 100 languages.

3 Methodology

In this section, we introduce our non-autoregressive sequence to sequence model for the semantic parsing problem.

3.1 Query Formulation

To train a sequence to sequence model, we prepare a source sequence and a target sequence. For the task of semantic parsing, the source sequence is the query in natural language. We construct the target sequence following Rongali et al. (2020) and Einolghozati et al. (2019). Tokens in the source sequence that are present in the target sequence are replaced with the special pointer token *ptr-n*, where *n* is the position of that token in the source sequence. By using pointers in the decoder, we can drastically reduce the vocabulary size. We follow previous work and use symmetrical tags for *intents* and *slots*. Fig. 1 shows two examples, a simple query and a complex query with the corresponding target sequences. This formulation is also able to express other tagging problems like named entity recognition (NER).

3.2 Insertion Transformer

We use the insertion transformer (Stern et al., 2019) as the base framework for the decoder. The insertion transformer is a modification of the original transformer decoder architecture (Vaswani et al., 2017). The original transformer decoder predicts the next token based on the previously generated sequence while the insertion transformer can predict tokens for all the available slots. In this setup, tokens in the decoder side can attend to the entire sequence instead of only their left side. This means we remove the causal self-attention mask in the original decoder.

3.2.1 Pointer Network with Copy

Pointer Network: In the normal sequence to sequence model, target tokens are generated by feeding the final representations (decoder hidden states) through a feed-forward layer and applying a softmax function over the whole target vocabulary. This is slow when the vocabulary size is large (Yang et al., 2018). In parsing, the entities in the source sequence will always appear in the target sequence. We can leverage the pointer mechanism (Vinyals et al., 2015) to reduce the target vocabulary size by dividing the vocabulary into two types: *tokens* that are the parsing symbols like intent and slot names, and *pointers* to words in the source sequence.

Since we have two kinds of target tokens, we use two slightly different ways to obtain unnormalized probabilities for each type. For the tokens in the tagging vocabulary, we feed the hidden states generated by the insertion transformer and slot pooling to a dense layer to produce the logits of size V (tagging vocabulary). The tagging vocabulary contains only the parse symbols like intents and slots together with several special tokens such as *bos*, *eos*, the padding and unknown token. For the pointers, we compute the scaled dot product attention scores between the slot representation and the encoder output. The attention scores will be computed as

$$a(Q,K) = \frac{QK^T}{\sqrt{h}}, \tag{2}$$

where query (Q) is the slot representation, the encoder outputs would be the key (K) and h is the hidden size of the query. Since the hidden size of encoder and decoder may be different, we also do a projection of query and key to the same dimension with two dense layers. Notice that the length of attention scores follows the length of the source sequence. Concatenating the attention scores with size n and the logits for the tagging vocabulary (V), we get the unnormalized distribution over $V + n$ tokens. We apply the $softmax$ function to obtain the final distribution over these tokens.

Copy Mechanism: Rongali et al. (2020) use a set of special embeddings to represent pointer tokens. This is a problem because the pointer embedding cannot encode semantic information since it points to different words across examples. Instead, we reuse the encoder output that the pointer token points to. Without copying, the special pointer embedding would learn a special position based representation for the source language that is hard to transfer to other languages.

3.3 Training and Loss

Training the insertion decoder requires sampling source and target sequences from the training data. We randomly sample valid subsequences from the target sequence to mimic intermediate insertion steps. We first sample a length $k \in [0,n]$ for the subsequence, where n is the length of the target sequence (here n excludes the bos and eos tokens). We select k tokens from the target sequence and maintain the original ordering. This sampling helps the model learn to insert tokens from the initial generation state as well as intermediate generation.

The insertion transformer can do parallel decoding since we can insert tokens in all available insertion slots. However, for each insertion slot, there may be multiple candidate tokens that can be inserted. For example, given a target sequence $[A,B,C,D,E]$ and a valid subsequence $[A,E]$, the candidates for the slot between token A and E are B,C,D. We use the two different weighting schemes proposed in Stern et al. (2019): uniform weights and balanced binary tree weights.

Binary Tree Weights: The motivation for applying binary tree weighting is to make the decoding time nearly O(log(n)). Consider the example of sequence A,B,C,D,E again, the desired order of generation would be $[bos,eos] \rightarrow [bos,C,eos] \rightarrow [bos,A,C,E,eos] \rightarrow [bos,A,B,C,D,E,eos]$. To achieve this goal, we weight the candidates according to their positions. For the sequence above, candidates in the span of $[bos,eos]$ are A,B,C,D,E. We assign token C the highest weight, then lower weights for B,D and the lowest weights for A,E.

Given a sampled subsequence with length $k+1$, we have k insertion slots at location $l = (0,1,...,k-1)$. Let $c_{l_0},...c_{l_i}$ be the candidates for one location l. We can define a distance function d_j for each token j in the candidates of l:

$$d_l(j) = |j - \frac{i}{2}|, \tag{3}$$

where i is the number of candidates in the location l. We then use the negative distance to compute the softmax based weighting (Rusu et al., 2016; Norouzi et al., 2016):

$$w_l(j) = \frac{exp(-d_l(j)/\tau)}{\sum_{m=0}^{i} exp(-d_l(m)/\tau)}. \tag{4}$$

Where τ is the temperature hyperparameter which allows us to control the sharpness of the weight distribution.

Uniform Weights: Instead of encouraging the model to follow a tree structure generation order, we can also treat the candidates equally. This performs better than the binary tree weights when we input the whole source sequence to the decoder as the initial sequence. In this case, we only need to insert the tagging tokens; the number of candidates is not as large as from scratch ($[bos,eos]$). This uniform weighting can be easily done by taking $\tau \rightarrow \infty$.

Loss Function: The autoregressive sequence to sequence model uses the negative log-likelihood loss since in each decoding step, there is only one ground-truth label. However, in our approach, we have multiple candidates for each insertion slot. Therefore, we use the KL-divergence between the predicted token distribution and the ground truth distribution. Then the loss for insertion slot l is:

$$L_{slot}(x, h_t, l) = D_{KL}((p_l|(x, h_t))||g_l), \qquad (5)$$

where p_l is the distribution output by the decoder and g_l is the target distribution where we set the probability to 0 for tokens that are not candidates. Note that the ground truth distribution depends on the weighting scheme for generation.

Finally, we have the complete loss averaged over all the insertion slots:

$$L(x, h_t) = \frac{1}{k+1} \sum_{l=0}^{k} L_{slot}(x, h_t, l) \qquad (6)$$

3.4 Termination Strategy

Terminating generation for insertion based decoding is not as straightforward as autoregressive decoding, which only needs the no-insertion token to be predicted. Insertion decoding requires a similar mechanism for every insertion slot. When computing the slot-loss above, if there are no candidates for the slot we set the ground truth label as the no-insertion token. At inference time, we can stop decoding when all available slots predict the no-insertion token. However, there is a problem when combining the sampling method and this termination strategy. The no-insertion token is more frequent compared with other tokens. The same situation is also encountered in (Stern et al., 2019). This is solved by adding a penalty hyperparameter to control the sequence length generated by the decoder. The hyperparameter is simply a scalar subtracted from the log probability of the no-insertion token for each insertion slot during inference. By doing this, we set a threshold for the difference between the no-insertion token and the second-best choice.

4 Experiments

In this section, we introduce the datasets and baseline models we experiment with. Then we report the results of monolingual experiments and cross lingual transfer learning experiments.

4.1 Datasets

4.1.1 SNIPS

The SNIPS dataset (Coucke et al., 2018) is a public dataset aimed to improve the semantic parsing models. It contains seven different intents: SearchCreativeWork, GetWeather, BookRestaurant, PlayMusic, AddToPlaylist, RateBook, and SearchScreeningEvent. For each intent, there are about 2000 training samples and 100 test samples. The SNIPS dataset consists of only simple queries.

4.1.2 ATIS

The Airline Travel Information System (ATIS) (Price, 1990) dataset was originally collected in the early 90s. The utterances are transcribed from the audio recordings of flight reservation calls. Similar to SNIPS, it consists of only simple queries. ATIS contains seventeen different intents. However, nearly 70% of the queries are the FLIGHT intent.

Recently, a multilingual version of ATIS called MultiATIS++ is introduced by Xu (2020). It is an extension of the Multilingual ATIS (Upadhyay et al., 2018). Besides the original three languages (English, Hindi and Turkish), MultiATIS++ adds six new languages including Spanish, German, Chinese, Japanese, Portuguese and French annotated by human experts and consists of a total of 37,084 training samples and 7,859 test samples. We exclude Turkish in our experiments as the test set size is limited.

4.1.3 TOP

Since ATIS and SNIPS contain only simple queries, the Facebook Task Oriented Parsing (TOP) dataset (Gupta et al., 2018) was introduced for complex hierarchical and nested queries that are more challenging. The dataset contains around 45,000 annotated queries with 25 intents and 36 slots. They further split them into training (31,000), validation (5,000) and test (9,000). As shown in Fig. 1, the nested slots make it harder to parse using a simple sequence tagging model. We also do experiments on multilingual TOP (Xia and Monti, 2021) with Italian and Japanese data. In this dataset, the training and validation set is machine translated while the test set is annotated by human experts.

4.2 Baseline Models

Monolingual Baselines: For monolingual experiments, we select the algorithms reported in Rongali et al. (2020) as baselines for ATIS and SNIPS.

Method	TOP		ATIS		SNIPS	
	EM	IC	EM	IC	EM	IC
Joint BiRNN (Hakkani-Tür et al., 2016)	-	-	80.70	92.60	73.20	96.90
Attention BiRNN (Liu and Lane, 2016)	-	-	78.90	91.10	74.10	96.70
Slot Gated Full Attention (Goo et al., 2018)	-	-	82.20	93.60	75.50	97.00
CapsuleNlU (Zhang et al., 2019)	-	-	83.40	95.00	80.90	97.30
SR(*S*)+ELMO+SVMRank (Gupta et al., 2018)	83.93	-	-	-	-	-
SR(*E*)+ELMO+SVMRank (Gupta et al., 2018)	**87.25**	-	-	-	-	-
AR-S2S-PTR (paper) (Rongali et al., 2020)	86.67	98.13	87.12	**97.42**	87.14	98.00
AR-S2S-PTR (reproduce) (Rongali et al., 2020)	85.67	98.17	88.91	97.09	90.71	**98.43**
IT-S2S-PTR ($\tau = 1$)	**86.74**	98.47	**89.14**	97.31	**91.00**	**98.43**
IT-S2S-PTR (input-src, uniform)	85.41	**98.71**	-	-	-	-

Table 1: Exact Match and Intent Classification scores for on the test set. Input-src means the initial input of the decoder is the whole source sequence. For the shift reduce parsing models, *E* denotes the ensemble model and *S* is the single model.

Model	Avg. steps	# tokens generated per step								
		1	2	3	4	5	6	7	8	9
AR-S2S-PTR	17.7	1	1	1	1	1	1	1	1	1
IT-S2S-PTR	5.9	1.0	2.0	3.96	6.66	6.24	3.17	1.6	1.4	1.2
IT-S2S-PTR(input-src)	**3.1**	4.99	2.92	1.37	1.00	0.54	0.27	0.25	1.0	1.0

Table 2: Decoding statistics on the TOP dataset. Average target sequence length of TOP is 17.7 tokens, we see that the insertion based parser can fully utilize binary tree decoding. ”input-src” means we set the whole source sequence as the initial decoder state.

Two of them leverage the power of RNNs: with attention (Liu and Lane, 2016) and without attention (Hakkani-Tür et al., 2016). Another model works completely with attention (Goo et al., 2018). A Capsule Networks based model is also included (Zhang et al., 2019). Finally, we compare with the autoregressive sequence to sequence with pointer model which is most recent (Rongali et al., 2020). Simple tagging based models cannot easily handle the complex queries in the TOP dataset. For the TOP dataset, we compare with two previous models, a shift reduce parsing model (Gupta et al., 2018) and the autoregressive sequence to sequence model (Rongali et al., 2020). For all monolingual experiments, we use RoBERTa as our pretrained encoder (Liu et al., 2019).

Cross lingual Baselines: For multilingual experiments (zero-shot and few-shot), we use a sequence labelling model based on multilingual BERT and an autoregressive sequence to sequence model (Rongali et al., 2020) as our baseline. To make fair comparasion, we also use the copy source mechanism in the AR model. For sequence labeling, instead of using F1 score, we also use the exact match (EM) which requires all intents and slots are labeled correctly by the model.

4.3 Results

4.3.1 Model Configuration

We use the pretrained RoBERTa and mBERT as the encoder for our model. For the decoder side, we use 4 layers with 12 heads transformer decoder. The hidden size of the decoder is the same as the embedding size of the pretrained encoder. For optimization, we use Adam (Kingma and Ba, 2015) with $\beta_1 = 0.9$ and $\beta_2 = 0.98$, paired with the Noam learning rate (initialized with 0.15) scheduler (Vaswani et al., 2017) with 500 warmup steps. For cross-lingual experiments, we freeze the encoder's embedding layer.

4.3.2 Monolingual Results

We use the exact match (EM) accuracy as the main metric to measure the performance of different models. By using EM, the entire parsing sequence predicted by the model has to match the reference sequence, since it's not easy to apply the F1 score or semantic error rate (Thomson et al., 2012) to

	en	es	pt	de	fr	hi	zh	ja*	avg
IT-S2S-PTR	**87.23**	**50.06**	**39.30**	**39.46**	**46.78**	11.42	**28.72**	12.60	32.69
AR-S2S-PTR	86.83	40.72	33.38	34.00	17.22	7.45	23.74	10.04	23.77
mBERT	86.33	48.46	38.56	39.12	42.98	**15.22**	21.89	**23.29**	32.78

Table 3: Zero-shot cross lingual EM scores by our approach (IT), autoregressive baseline (AR) and sequence labeling baseline (mBERT). Results are averaged over four random seeds. For our approach, we initialize the decoder with source sequences. * indicates that the data format for the language is not consistent with the S2S model tokenizer.

	en	it	ja
IT-S2S-PTR	84.61	50.07	3.64
AR-S2S-PTR	**85.4**	41.06	0.64

Table 4: Zero-shot EM scores on multilingual TOP dataset. Model is trained on English only.

complex queries. It's better to use the EM here for both simple and complex queries. We also report the intent classification accuracy for our models.

Main Result: Table 1 shows the results from monolingual experiments on three datasets: TOP, ATIS and SNIPS. Our insertion transformer with pointer achieves new state-of-the-art performance on ATIS and SNIPS under EM metric. For TOP dataset, our model matches the best performance reported for single models (AR-S2S-PTR) despite being 3x faster.

We also experiment with starting generation with the entire source sequence as the initial state of the decoder. The performance degrades slightly in this case, possibly due to a training/inference mismatch in this setting. This degradation is likely due to training the model to generate the entire target sequence but only asking the model to generate tags during inference.

Decoding Steps: Since our approach can do parallel decoding, the number of decoding steps is only O(log(n)). Table 2 shows the statistics for the average decoding steps for the TOP dataset and the number of generated tokens per step. The insertion transformer with pointer only needs 5.9 steps while the autoregressive needs 17.7, resulting in a 3x speedup with insertion decoding. The decoding steps can be further reduced to 3.1 when we start decoding with the source sequence as the initial sequence for the decoder. Theoretically, a perfect binary tree based insertion model should generate 2^{n-1} tokens for the n_{th} decoding step. We can see that our approach can make full use of the parallel decoding during the first three steps, since the average length of TOP's test samples is only 17.7.

Weighting Strategy: We do experiments on both binary tree weighting and uniform weighting for the TOP dataset. We set $\tau \in [0.5, 1.0, 1.5, 2.0]$ and find 1.0 performs best. Binary tree weights are better than uniform in the setting of decoding from scratch. However, uniform performs better when we decode from the whole source sequence.

4.3.3 Cross Lingual Transfer Results

For MultiATIS++, we train on English training data and test on all languages. Table 3 shows the results of our approach compared to the autoregressive and sequence labeling baselines. We find that:

- Our approach outperforms the baseline on most of the languages except Hindi and Japanese. For Japanese, we found inconsistencies in the tokenizer that is the likely cause of the degradation [1].

- The autoregressive baseline performs poorly on cross lingual experiments. For example, it only achieves 17.22 EM on the French test set while the other two systems achieve >40 EM. This highlights the weakness of autoregressive parsers that cannot produce parses directly from the encoded representations of the source sequence.

- The order of the sentence in Hindi and Japanese is different from others, this may limit the performance of transfer learning for S2S parsers.

We also test on the multilingual TOP dataset (Xia and Monti, 2021), which extends the TOP datasets

[1] Chinese is tokenized at the character level in mBERT, while Katakana/Hiragana are tokenized with whitespace. Data in MultiATIS++ is mixed in these two fashions.

	IT-S2S-PTR				AR-S2S-PTR			
# samples	0	10	50	100	0	10	50	100
it	**50.07**	50.13	52.69	**56.42**	41.06	42.23	44.98	46.96
ja	**3.64**	4.7	18.01	**18.96**	0.64	1.73	10.78	18.56

Table 5: Few-shot EM scores on multilingual TOP dataset with model pretrained on English. Training samples used in few-shot are sampled from the test set and excluded during testing.

to other languages providing human annotated Italian and Japanese test sets. TOP contains a much larger test set compared to ATIS. Table 4 shows the zero-shot results and Table 5 shows the few-shot results.

In the zero-shot setting, our approach achieves 50.07 EM score for Italian while AR only achieves 41.06. Both models are unable to achieve good performance in the zero-shot setting for Japanese. We speculate on this behavior in the few-shot experiment results.

In the few shot setting, we finetune the model in two stages, first on the entire English data and then with 10, 50, 100 training samples from other languages. Our approach outperforms the AR baseline in all few shot settings. For Italian, increasing training samples from 10 to 100 does not result in much gain, since the knowledge from English can readily be transferred to Italian, probably due to the similarity of the languages. To further improve the performance on Italian, the model may need many more training samples. However, for Japanese little knowledge (like word order) can be transferred from English so both models can perform as if training from scratch. There may be two reasons here: 1) the order of a sentence is different from English. 2) the annotated target is aligned with the original words in the multilingual TOP so the order of pointers are mixed. Thus, we see the EM scores improves drastically as the number of training samples increases.

4.4 Ablation Study

For ablation study, we separate the experiments to monolingual and multilingual as above. For multilingual experiments, we use the Italian from multilingual TOP dataset.

From Table 6, we observe that the copy mechanism improves performance in the monolingual setting. For the hyperparameter τ, recall that a higher value for τ would result in flatter (more uniform) weights for the candidates. $\tau = 1.0$ provides the best balance between equally weighting the candidates and weighting the next token to be inserted heavily. However, we find that when initializing the decoder with source sequences, uniform weights performs better than binary tree weights.

	EM
IT-S2S-PTR	86.74
$\tau = 0.1$	74.84
$\tau = 0.5$	85.47
$\tau = 1.0$	**86.74**
$\tau = 1.5$	86.33
no copy	86.09

Table 6: The ablation study for the τ parameter and copy source embedding vector vs. no copy in the monolingual setting. Results on the TOP dataset show the importance of copying source embeddings. We also observe that small values of τ (i.e. weighting the central token for insertion heavily) degrade performance.

Models	EM
IT-S2S-PTR-Best	50.07
- copy	47.00
- input-src	42.03
AR-S2S-PTR-BEST	41.06
- copy	30.87

Table 7: The ablation study for source embedding copying and starting generation from source tokens in the cross-lingual setting. Results are zero-shot in Italian. For the IT-S2S model, both copying and starting generation with source tokens contribute to zero-shot performance

For cross-lingual experiments, we introduce two components to improve the performance. Table 7 shows that both of them help in the zero-shot transfer setting. From the results, we can observe that initializing the decoder with the source sequence plays an important role in zero-shot transfer, which is impossible for the autoregressive based models. The copy mechanism is again beneficial for both the sequence to sequence models, improving the performance of even the autoregressive model from 30.87 EM to 41.06 EM in the zero-shot Italian experiment.

5 Related Work

Monolingual Semantic Parsing: The task oriented semantic parsing for intent classification and slot detection is usually achieved by sequence labeling. Normally, the system will first classify the query based on the sentence level semantic and then label each word in the query. Conditional Random Fields (CRFs) (Peters et al., 2018; Lan et al., 2020; Jiao et al., 2006) is one of the most successful algorithms applied to this task before deep learning dominated the area. Deep learning algorithms boost the performance of semantic parsing, especially using recurrent neural networks (Liu and Lane, 2016; Hakkani-Tür et al., 2016). Other architectures are also explored, such as convolutional neural networks (Kim, 2014) and capsule networks (Zhang et al., 2019).

Cross Lingual Transfer Semantic Parsing: Multilingual natural language understanding has been studied in a variety of tasks including part-of-speech (POS) tagging (Plank and Agić, 2018; Yarowsky et al., 2001; Täckström et al., 2013), named entity recognition (Zirikly and Hagiwara, 2015; Tsai et al., 2016; Xie et al., 2018) and semantic parsing (Xu et al., 2020). Before the advent of pretrained cross-lingual language models, researchers leveraged the representations learned by multilingual neural machine translation (NMT). Another approach is to use NMT to translate between the source language and the target language. However, it is challenging for the sequence tagging tasks: labels on the source language need to be projected on the translated sentences (Xu et al., 2020). Pretrained cross-lingual language models (Devlin et al., 2019; CONNEAU and Lample, 2019) achieve great success in various multilingual natural language tasks.

6 Conclusion

In this paper, we tackle two shortcomings of the autoregressive sequence to sequence semantic parsing models: 1) expensive decoding and 2) poor cross-lingual performance.

We propose 1) insertion transformer with pointers and 2) a copy mechanism which replaces the pointer embeding with corresponding encoder outputs, to mitigate these two problems. Our model can achieve O(log(n)) decoding time with parallel decoding. For the specific task of semantic parsing, we can further reduce the decoding steps by initializing the decoder sequence with the whole source sequence. Our model achieves new state-of-the-art performance on both simple queries (ATIS and SNIPS) and complex queries (TOP). In cross-lingual transfer, our approach surpasses the baselines in the zero-shot setting by 9 EM points on average across 9 languages.

Acknowledgements

We thank the reviewers for their excellent feedback. Special thanks also to Emilio Monti and Menglin Xia for their help with the multilingual TOP dataset.

References

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Alexis CONNEAU and Guillaume Lample. 2019. Cross-lingual language model pretraining. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 7059–7069. Curran Associates, Inc.

Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibault Lavril, Maël Primet, and Joseph Dureau. 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *CoRR*, abs/1805.10190.

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, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. In *Advances in Neural Information Processing Systems*, pages 13042–13054.

Long Duong, Hadi Afshar, Dominique Estival, Glen Pink, Philip R Cohen, and Mark Johnson. 2017. Multilingual semantic parsing and code-switching. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 379–389.

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

Chih-Wen Goo, Guang Gao, Yun-Kai Hsu, Chih-Li Huo, Tsung-Chieh Chen, Keng-Wei Hsu, and Yun-Nung Chen. 2018. Slot-gated modeling for joint slot filling and intent prediction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 753–757.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor O.K. Li, and Richard Socher. 2018. Non-autoregressive neural machine translation. In *International Conference on Learning Representations*.

Jiatao Gu, Changhan Wang, and Junbo Zhao. 2019. Levenshtein transformer. In *Advances in Neural Information Processing Systems*, pages 11181–11191.

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.

D. Hakkani-Tür, G. Tür, A. Çelikyilmaz, Yun-Nung Chen, Jianfeng Gao, L. Deng, and Ye-Yi Wang. 2016. Multi-domain joint semantic frame parsing using bi-directional rnn-lstm. In *INTERSPEECH*.

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

Feng Jiao, Shaojun Wang, Chi-Hoon Lee, Russell Greiner, and Dale Schuurmans. 2006. Semi-supervised conditional random fields for improved sequence segmentation and labeling. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 209–216, Sydney, Australia. Association for Computational Linguistics.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

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

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: a lite BERT for self-supervised learning of language representations. In *International Conference on Learning Representations*.

Bing Liu and Ian Lane. 2016. Attention-based recurrent neural network models for joint intent detection and slot filling. In *Interspeech 2016, 17th Annual Conference of the International Speech Communication Association, San Francisco, CA, USA, September 8-12, 2016*, pages 685–689. ISCA.

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. *CoRR*, abs/1907.11692.

Grégoire Mesnil, Xiaodong He, Li Deng, and Yoshua Bengio. 2013. Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding. In *Interspeech*, pages 3771–3775.

Mohammad Norouzi, Samy Bengio, Navdeep Jaitly, Mike Schuster, Yonghui Wu, Dale Schuurmans, et al. 2016. Reward augmented maximum likelihood for neural structured prediction. In *Advances In Neural Information Processing Systems*, pages 1723–1731.

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

Barbara Plank and Željko Agić. 2018. Distant supervision from disparate sources for low-resource part-of-speech tagging. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 614–620, Brussels, Belgium. Association for Computational Linguistics.

Patti Price. 1990. Evaluation of spoken language systems: The ATIS domain. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27, 1990*.

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.

Andrei A. Rusu, Sergio Gomez Colmenarejo, Çaglar Gülçehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. 2016. Policy distillation. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Mitchell Stern, William Chan, Jamie Kiros, and Jakob Uszkoreit. 2019. Insertion transformer: Flexible sequence generation via insertion operations. In *ICML*.

Raymond Hendy Susanto and Wei Lu. 2017. Neural architectures for multilingual semantic parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 38–44.

Oscar Täckström, Dipanjan Das, Slav Petrov, Ryan McDonald, and Joakim Nivre. 2013. Token and type constraints for cross-lingual part-of-speech tagging. *Transactions of the Association for Computational Linguistics*, 1:1–12.

Blaise Thomson, Milica Gasic, Matthew Henderson, Pirros Tsiakoulis, and Steve Young. 2012. N-best error simulation for training spoken dialogue systems. In *2012 IEEE Spoken Language Technology Workshop (SLT)*, pages 37–42. IEEE.

Chen-Tse Tsai, Stephen Mayhew, and Dan Roth. 2016. Cross-lingual named entity recognition via wikification. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 219–228, Berlin, Germany. Association for Computational Linguistics.

Shyam Upadhyay, Manaal Faruqui, Gokhan Tür, Hakkani-Tür Dilek, and Larry Heck. 2018. (almost) zero-shot cross-lingual spoken language understanding. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6034–6038. IEEE.

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

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in neural information processing systems*, pages 2692–2700.

Menglin Xia and Emilio Monti. 2021. Multilingual neural semantic parsing with pretrained encoders. In *Proceedings of the 16th European Chapter of the Association for Computational Linguistics*. Submitted.

Jiateng Xie, Zhilin Yang, Graham Neubig, Noah A. Smith, and Jaime Carbonell. 2018. Neural cross-lingual named entity recognition with minimal resources. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 369–379, Brussels, Belgium. Association for Computational Linguistics.

Weijia Xu, Batool Haider, and Saab Mansour. 2020. End-to-end slot alignment and recognition for cross-lingual NLU. *CoRR*, abs/2004.14353.

Zhilin Yang, Zihang Dai, Ruslan Salakhutdinov, and William W. Cohen. 2018. Breaking the softmax bottleneck: A high-rank RNN language model. In *International Conference on Learning Representations*.

David Yarowsky, Grace Ngai, and Richard Wicentowski. 2001. Inducing multilingual text analysis tools via robust projection across aligned corpora. In *Proceedings of the First International Conference on Human Language Technology Research*.

Chenwei Zhang, Yaliang Li, Nan Du, Wei Fan, and Philip Yu. 2019. Joint slot filling and intent detection via capsule neural networks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5259–5267, Florence, Italy. Association for Computational Linguistics.

Ayah Zirikly and Masato Hagiwara. 2015. Cross-lingual transfer of named entity recognizers without parallel corpora. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 390–396, Beijing, China. Association for Computational Linguistics.