Non-Autoregressive Semantic Parsing for Compositional Task-Oriented Dialog

Arun Babu **Akshat Shrivastava** **Armen Aghajanyan**
Ahmed Aly **Angela Fan** **Marjan Ghazvininejad**
Facebook
{arbabu, akshats, armenag, ahhegazy, angelafan, ghazvini}@fb.com

Abstract

Semantic parsing using sequence-to-sequence models allows parsing of deeper representations compared to traditional word tagging based models. In spite of these advantages, widespread adoption of these models for real-time conversational use cases has been stymied by higher compute requirements and thus higher latency. In this work, we propose a non-autoregressive approach to predict semantic parse trees with an efficient seq2seq model architecture. By combining non-autoregressive prediction with convolutional neural networks, we achieve significant latency gains and parameter size reduction compared to traditional RNN models. Our novel architecture achieves up to an 81% reduction in latency on TOP dataset and retains competitive performance to non-pretrained models on three different semantic parsing datasets. Our code is available at https://github.com/facebookresearch/pytext.

1 Introduction

Advances in conversational assistants have helped to improve the usability of smart speakers and consumer wearables for different tasks. Semantic parsing is one of the fundamental components of these assistants and it helps to convert the user input in natural language to a structure representation that can be understood by downstream systems. Majority of the semantic parsing systems deployed on various devices, rely on server-side inference because of the lower compute/memory available on these edge devices. This poses a few drawbacks such as flaky user experience with spotty internet connectivity and compromised user data privacy due to the dependence on a centralized server to which all user interactions are sent to. Thus, semantic parsing on-device has numerous advantages.

For the semantic parsing task, the meaning representation used decides the capabilities of the system built. Limitations of the representation with one intent and slot labels were studied in the context of nested queries and multi turn utterances in Aghajanyan et al. (2020) and Gupta et al. (2018). New representations were proposed to overcome these limitations and sequence-to-sequence models were proposed as the solution to model these complex forms. But using these new models in real-time conversational assistants still remains a challenge due to higher latency requirements. In our work, we propose a novel architecture and generation scheme to significantly improve the end2end latency of sequence-to-sequence models for the semantic parsing task.

Due to the autoregressive nature of generation in sequence-to-sequence semantic parsing models, the recurrence relationship between target tokens creates a limitation that decoding cannot be parallelized.

There are multiple works in machine translation which try to solve this problem. These approaches relax the decoder token-by-token generation by allowing multiple target tokens to be generated at once. Fully non-autoregressive models (Gu et al., 2017; Ma et al., 2019; Ghazvininejad et al., 2020a; Saharia et al., 2020) and conditional masked language models with iterative decoding (Ghazvininejad et al., 2019; Gu et al., 2019; Ghazvininejad et al., 2020b) are some of them.

To enable non-autoregressive generation in semantic parsing, we modify the objective of the standard seq2seq model to predict the entire target structure at once. We build upon the CMLM (Conditional Masked Language Model) (Ghazvininejad et al., 2019) and condition the generation of the full target structure on the encoder representation. By eliminating the recurrent relationship between individual target tokens, the decoding process can be parallelized. While this drastically improves latency, the representation of each token is still dependent on previous tokens if we continue to use an RNN architecture. Thus, we propose a novel

Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 2969–2978
June 6–11, 2021.

model architecture for semantic parsing based on convolutional networks (Wu et al., 2019b) to solve this issue.

Our non-autoregressive model achieves up to an 81% reduction in latency on the TOP dataset (Gupta et al., 2018), while achieving 80.23% exact match accuracy. We also achieve 88.16% exact match accuracy on DSTC2 (Henderson et al., 2014) and 80.86% on SNIPS (Coucke et al., 2018) which is competitive to prior work without pretraining.

To summarize, our two main contributions are:

- We propose a novel alternative to the traditional autoregressive generation scheme for semantic parsing using sequence-to-sequence models. With a new model training strategy and generation approach, the semantic parse structure is predicted in one step improving parallelization and thus leading to significant reduction in model latency with minimal accuracy impact. We also study the limitations of original CMLM (Ghazvininejad et al., 2019) when applied for conversational semantic parsing task and provide motivations for our simple yet critical modifications.
- We propose LightConv Pointer, a model architecture for non-autoregressive semantic parsing, using convolutional neural networks which provides significant latency and model size improvements over RNN models. Our novel model architecture is particularly suitable for limited compute use-cases like on-device conversational assistants.

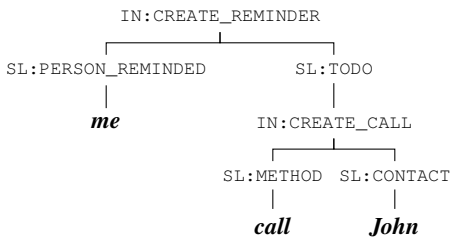


Figure 1: Decoupled semantic representation for the single utterance *"Please remind me to call John"*.

2 Method

In this section, we propose a novel, convolutional, non-autoregressive architecture for semantic parsing. While non-autoregressive decoding has been previously explored in machine translation, we describe how it can be applied to semantic parsing with several critical modifications to retain performance. We then describe our convolutional architecture. By incorporating these advances together, our approach achieves both high accuracy and efficient decoding.

The task is to predict the semantic parse tree given the raw text. We use the decoupled representation (Aghajanyan et al., 2020), an extension of the compositional form proposed in Gupta et al. (2018) for task oriented semantic parsing. Decoupled representation is obtained by removing all text in the compositional form that does not appear in a leaf slot. Efficient models require representations which are compact, with least number of tokens, to reduce number of floating point operations during inference. Decoupled representation was found to be suitable due to this.

Figure 1 shows the semantic parse for a sample utterance. Our model predicts the serialized representation of this tree which is

```
[IN:CREATE_REMINDER [SL:PERSON_REMINDED me ]
[SL:TODO [IN:CREATE_CALL [SL:METHOD call ]
[SL:CONTACT John ] ] ] ]
```

2.1 Non-Autoregressive Decoding

While autoregressive models (Figure 2), which predict a sequence token by token, have achieved strong results in various tasks including semantic parsing, they have a large downside. The main challenge in practical applications is the slow decoding time. We investigate how to incorporate recent advances in non-autoregressive decoding for efficient semantic parsing models.

We build upon the Conditional Masked Language Model (CMLM) proposed in Ghazvininejad et al. (2019) by applying it to the structured prediction task of semantic parsing for task-oriented dialog. Ghazvininejad et al. (2019) uses CMLM to first predict a token-level representation for each source token and a target sequence length; then the model predicts and iterates on the target sequence prediction in a non-autoregressive fashion. We describe our changes and the motivations for these changes below.

One of the main differences between our work and Ghazvininejad et al. (2019) is that target length prediction plays a more important role in semantic parsing. For the translation task, if the target length is off by one or more, the model can slightly rephrase the sentence to still return a high quality translation. In our case, if the length prediction is

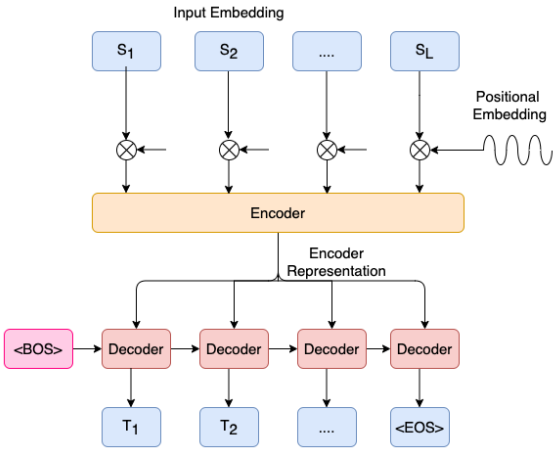


Figure 2: Traditional Sequence to Sequence architecture which uses autoregressive generation scheme for decoder.

off by even one, it will lead to an incorrect semantic parse.

To resolve this important challenge, we propose a specialized length prediction module that more accurately predicts the target sequence length. While Ghazvininejad et al. (2019) uses a special CLS token in the source sequence to predict the target length, we have a separate module of multiple layers of CNNs with gated linear units to predict the target sequence length (Wu et al., 2019b). We also use label smoothing and differently weighing losses as explained in section 2.3, to avoid the easy overfitting in semantic parsing compared to translation.

As shown in Aghajanyan et al. (2020), transformers without pre-training perform poorly on TOP dataset. The architectural changes that we propose to solve the data efficiency can be found in the section 2.2.1.

Further, we find that the random masking strategy proposed in Ghazvininejad et al. (2019) works poorly for semantic parsing. When we use the same strategy for the semantic parsing task where the output has a structure, model is highly likely to see invalid trees during training as masking random tokens in the linearized representation of a tree mostly gives invalid tree representations. This makes it hard for the model to learn the structure especially when the structure is complicated (in the case of trees, deep trees were harder to learn). To remedy this problem, we propose a different strategy for model training where all the tokens in the target sequence are masked during training.

Ablation experiments for all the above changes can be found in section 4.3.

2.2 LightConv Pointer Model

2.2.1 Model Architecture

Our model architecture (Figure 3) is based on the classical seq2seq model (Sutskever et al., 2014) and follows the encoder-decoder architecture. In order to optimize for efficient encoding and decoding, we look to leverage a fully parallel model architecture. While transformer models are fully parallel and popular in machine translation (Vaswani et al., 2017), they are known to perform poorly in low resource settings and require careful tuning using techniques like Neural Architecture Search to get good performance (van Biljon et al., 2020; Murray et al., 2019). Similarly, randomly initialized transformers performed poorly on TOP dataset achieving only 64.5 % accuracy when SOTA was above 80% (Aghajanyan et al., 2020). We overcome this limitation by augmenting Transformers with Convolutional Neural Networks. Details of our architecture are explained below.

For token representations, we use word embeddings concatenated with the sinusoidal positional embeddings (Vaswani et al., 2017). Encoder and decoder consist of multiple layers with residual connections as shown in Figure 4.

First sub-block in each layer consists of MHA (Vaswani et al., 2017). In decoder, we do not do masking of future tokens during model training. This is needed for non-autoregressive generation of target tokens during inference.

Second sub-block consists of multiple convolutional layers. We use depthwise convolutions with weight sharing (Wu et al., 2019b). Convolution layer helps in learning representation for tokens for a fixed context size and multiple layers helps with bigger receptive fields. We use non-causal convolutions for both encoder as well as decoder.

Third sub-block is the FFN (Vaswani et al., 2017; Wu et al., 2019b) which consists of two linear layers and relu. The decoder has source-target attention after the convolution layer.

Pointer-Generator Projection layer The decoder has a final projection layer which generates the target tokens from the decoder/encoder representations. Rongali et al. (2020) proposes an idea based Pointer Generator Network (See et al., 2017) to convert the decoder representation to target tokens using the encoder output. Similarly, we use a pointer based projection head, which decides whether to copy tokens from the source-sequence or generate from the pre-defined ontology at every

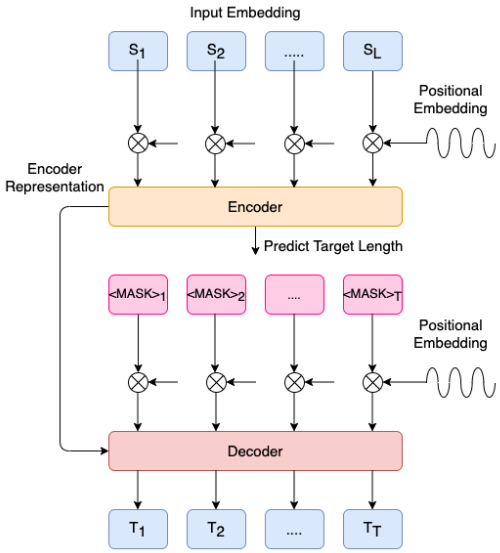


Figure 3: Sequence to Sequence model architecture which uses Non-Autoregressive strategy for generation

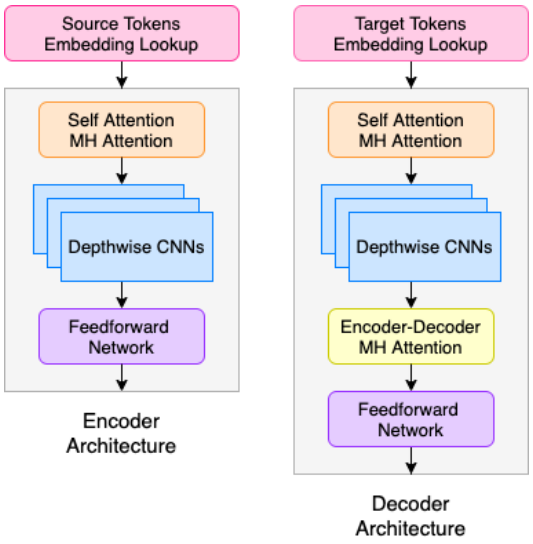


Figure 4: Different layers in LightConv Pointer Model Architecture

decoding step (Aghajanyan et al., 2020).

Length Prediction Module Length prediction Module receives token level representations from the encoder as input. It uses stacked CNNs with gated linear units and mean pooling to generation the length prediction.

2.2.2 Inference

Suppose the source sequence is of length L and source tokens in the raw text are $s_1, s_2, s_3 \dots s_L$. Encoder generates a representation of for each token in the source sequence.

$$e_1, \dots, e_L = \text{Encoder}(s_1, \dots, s_L) \quad (1)$$

The length prediction module predicts the target sequence length using the token level encoder representation.

$$T = \text{PredictLength}(e_1, \dots, e_L) \quad (2)$$

Using the predicted length T, we create a target sequence of length T consisting of identical MASK tokens. This sequence is passed through possibly multiple decoder layers and generates a representation for each token in the masked target sequence.

$$x_1, ..., x_T = \text{Dec}(\text{MASK}_1, \dots, \text{MASK}_T; e_1, ..., e_L) \quad (3)$$

We use **Pointer-Generator Projection layer** explained in 2.2.1 to predict target tokens.

$$y_1, ..., y_T = \text{PtrProj}(x_1, .., x_T; e_1, .., e_L) \quad (4)$$

We make a strong assumption that each token in the target sentence is conditionally independent of each other given the source and the target length. Thus, the individual probabilities for each token is $P(y_i|X, T)$ where X is the input sequence and T is the length of target sequence.

Beam Search During inference, length prediction module explained in 2.2.1 predicts top k lengths. For each predicted length, we create a decoder input sequence of all masked tokens. This is similar to the beam search with beam size k in autoregressive systems. The main difference in our model architecture is that we expect only one candidate for each predicted length. These all masked sequences are given as input to the model and the model predicts target tokens for each masked token. Once we have predicted target sequences for k different lengths, they are ranked based on the ranking algorithm described in (5), where X is the input sequence and Y is the predicted output sequence, note the predicted token y_i is conditioned on both the sequence (X) and the predicted target length T.

$$S(X, Y) = \sum_{i}^{T} P(y_i \mid X, T) \cdot P(T) \quad (5)$$

2.3 Training

During training, we jointly optimize for two weighted losses. The first loss is calculated for the predicted target tokens against the real target and the second loss is calculated for predicted target length against real target length.

During forward-pass, we replace all the tokens in the target sequence with a special <MASK> token and give this as an input to the decoder. Decoder predicts the token for each masked token and the cross-entropy loss is calculated for each predicted token.

The length prediction module in the model predicts the target length using the encoder representation. Similar to CMLMs in (Ghazvininejad et al., 2019), length prediction is modeled as a classification task with class labels for each possible length. Cross entropy loss is calculated for length prediction. For our semantic parsing task, label smoothing (Szegedy et al., 2015) was found to be very critical as the length prediction module tends to easily overfit and strong regularization methods are needed. This was because length prediction was a much well-defined task compared to predicting all the tokens in the sequence.

Total loss was calculated by taking a weighted sum of cross entropy loss for labels and length, with lower weight for length loss.

As training progresses through different epochs, the best model is picked by comparing the exact match (EM) accuracy of different snapshots on validation set.

3 Experiments

3.1 Datasets

We use 3 datasets across various domains to evaluate our semantic parsing approach. Length distribution of each dataset is described using median, 90th percentile and 99th percentile lengths.

TOP Dataset Task Oriented Parsing (Gupta et al., 2018) is a dataset for compositional utterances in the navigation and events domains. The training set consists of $31,279$ instances and the test set consists of $9,042$. The test set has a median target length of 15, P90 27 and P99 39.

SNIPS The SNIPS (Coucke et al., 2018) dataset is a public dataset used for benchmarking semantic parsing intent slot models. This dataset is considered flat, since it does not contain compositional queries and can be solved with word-tagging models. Recently, however seq2seq models have started to out perform word-tagging models (Rongali et al., 2020; Aghajanyan et al., 2020). The training set consists of $13,084$ instances, the test set consists of 700 instances. The test set has a median target length of 11, P90 17, P99 21.

DSTC2 Dialogue State Tracking Challenge 2 (Henderson et al., 2014), is a dataset for conversational understanding. The dataset involves users searching for restaurants, by specifying constraints such as cuisine type and price range, we encode these constraints as slots and use this to formulate the decoupled representation. The training set consists of $12,611$ instances and a test set of 9890. The test set has a median target length of 6, P90 9 and P99 10.

3.2 Evaluation

Semantic Parsing Performance For all our datasets, we convert the representation of either the compositional form or flat intent slot form to the decoupled representation (Aghajanyan et al., 2020) . We compare the model prediction with the serialized structure representation and look for exact match (EM).

Benchmarking Latency For the latency analysis for the models trained from scratch: AR LightConv Pointer, NAR LightConv Pointer, and BiLSTM. We chose these 3 architectures, to compare NAR vs AR variants of LightConv Pointer, as well as the best performant baseline: Pointer BiLSTM (Aghajanyan et al., 2020). We use Samsung Galaxy S8 with Android OS and Octa-core processor. We chose to benchmark latency to be consistent with prior work on on-device modeling (Wu et al., 2019a; Howard et al., 2019). All models are trained in PyTorch (Paszke et al., 2019) and exported using Torchscript. We measure wall clock time as it is preferred instead of other options because it relates more to real world inference. [1] Latency results can be found in section 4.2.

3.3 Baselines

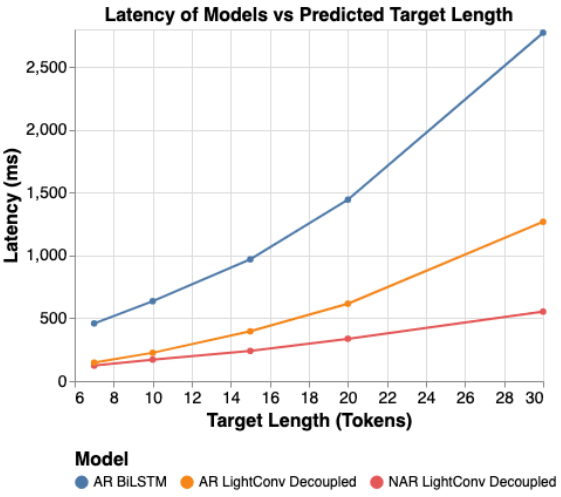
For each of our datasets, we report accuracy metrics on the following models:

AR LightConv Pointer: Autoregressive (AR) LightConv Pointer model to establish an autoregressive baseline of our proposed architecture.

[1] We use the open source framework https://github.com/facebook/FAI-PEP for latency benchmarking.

Model	Exact Match Accuracy TOP	DSTC2	SNIPS
RNNG (Einolghozati et al., 2018)	80.86	-	-
Ptr Transformer (Rongali et al., 2020)	79.25	-	85.43
Ptr BiLSTM (Aghajanyan et al., 2020)	79.51	88.33	-
GLAD (Zhong et al., 2018)	-	79.4	-
JointBiRNN (Hakkani-Tür et al., 2016)	-	-	73.20
Slot Gated (Goo et al., 2018)	-	-	75.50
Capsule NLU (Zhang et al., 2018)	-	-	80.90
	Ours		
NAR LightConv Pointer	80.20	88.16	80.86
AR LightConv Pointer	80.23	88.58	76.43

(a) Exact Match Accuracy on TOP, DSTC2, and SNIPS



(b) Median latency on TOP dataset

Figure 5: **(a)**: Exact match (EM) accuracy is shown on the test set across 3 different datasets. We compare our proposed LightConv Pointer variants (AR and NAR) against various baselines that do not include pre-trained representations. **(b)**: Median latency of the NAR LightConv Pointer, AR LightConv Pointer, and the Seq2Seq Pointer BiLSTM baseline (termed Ptr BiLSTM in figure 5a) (Aghajanyan et al., 2020) varying over increasing target sequence length on the TOP dataset.

Model			EM	Mean Length Bucket Exact Match Accuracy (%) < 10 EM	10-20 EM	20-30 EM	> 30 EM
CMLM Transformer	+ CLS	+ Random Masking	70.9	79.3	74.5	35.1	0.4
CMLM LightConv	+ CLS	+ Random Masking	78.3	82.9	81.8	53.5	3.9
CMLM LightConv	+ Conv Length	+ Random Masking	79.4	83.3	82.5	58.2	5.1
CMLM LightConv	+ CLS	+ Mask Everything	78.6	82.7	81.8	56.0	9.4
CMLM LightConv	+ Conv Length	+ Mask Everything	79.6	82.2	82.8	61.4	14.9

Table 1: Ablation experiments reporting EM in different buckets based on the target sequence length. Bucket sizes are 2798, 5167, 992 and 85 respectively. It can be seen the our model setup works significantly better, especially for longer sequences.

NAR LightConv Pointer: A non-autoregressive (NAR) variant of the above model to allow for parallel decoding.

We compare against the best reported numbers across datasets where the models don't use pre-training.

3.4 Model Training Details

During training of our model we use the same base model across all datasets and sweep over hyper parameters for the length module and the batch size and learning rate, an equivalent sweep was done for the AR variant as well. The base model we use for NAR LightConv Pointer model uses 5 encoder layers with convolutional kernel sizes [3,7,15,21,27], where each encoder layer has embedding and convolutional dimensions of 160, 1 self attenion head, and 2 decoder layers with kernel sizes [7,27], and embedding dimension of 160, 1 self-attention head and 2 encoder-attention heads. Our length prediction module leverages a two convolution layers of 512 embedding dimensions and kernel sizes of 3 and 9. and uses hidden dimension in [128,256,512] determined by hyper parameter sweeps. We also use 8 attention heads for the decoupled projection head. For the convolutional layer, we use lightweight convolutions (Wu et al., 2019b) with number of heads set to 2. We train with the Adam (Kingma and Ba, 2014) optimizer, learning rate is selected to be between [0.00007, 0.0004]. If our evaluation accuracy has not increased in 10 epochs, we also reduce our learning rate by a factor of 10, and we employ early stopping if the accuracy has not changed in 20 epochs. We train with our batch size fixed to be 8. We optimize a joint loss for label prediction and length prediction. Both losses consist of label smoothed cross entropy loss (β is the weight of the uniform distribution) (Pereyra et al., 2017), our label loss has $\beta = 0.1$ and our length loss has $\beta = 0.5$, we

also weight our length loss lower, $\lambda = 0.25$. For inference, we use a length beam size of $k = 5$. Our AR variant follows the same parameters however it does not have length prediction and self-attention in encoder and decoder.

4 Results

We show that our proposed non-autoregressive convolutional architecture for semantic parsing is competitive with auto-regressive baselines and word tagging baselines without pre-training on three different benchmarks and reduces latency up to 81% on the TOP dataset. We first compare accuracy and latency, then discuss model performance by analyzing errors by length, and the importance of knowledge distillation. We do our analysis on the TOP dataset, due to its inherent compositional nature, however we expect our analysis to hold for other datasets as well. Non-compositional datasets like DSTC2 and SNIPS can be modeled by word tagging models making seq2seq models more relevant in the case of compositional datasets.

4.1 Accuracy

In table 5a we show our NAR and AR variants for LightConv Pointer perform quite similarly across all datasets. We can see that our proposed NAR LightConv Pointer is also competitive with state of the art models without pre-training: -0.66% TOP, -0.17% DSTC2, -4.57% SNIPS (-0.04% compared to word tagging models). Following the prior work on Non-Autoregressive models, we also report our experiments with sequence-level knowledge distillation in subsection Knowledge Distillation under section. 4.3.

4.2 Latency

In figure 5b we show the latency of our model with different generation approaches (NAR vs AR) over increasing target sequence lengths on the TOP dataset. Firstly, we show that our LightConv Pointer is significantly faster than the BiLSTM baseline (Aghajanyan et al., 2020), achieving up to a 54% reduction in median latency. BiLSTM was used as baseline as that was the SOTA without pretraining for TOP and Transformers performed poorly. By comparing our model with AR and NAR generation strategy, it can be seen that increase in latency with increase in target length is much smaller for NAR due to better parallelization of decoder, resulting in up to an 81% reduction in median latency compared to the BiLSTM model. Also note that both the LightConv Pointer models are able to achieve parity in terms of EM Accuracy compared to the baseline BiLSTM model, while using many fewer parameters, the BiLSTM model uses 20M parameters, while the NAR LightConv Pointer uses 12M and the AR LightConv Pointer uses 10M.

Length Bucket	NAR (%)	AR (%)	Bucket Size
< 10	82.80	83.13	2798
10-20	84.18	84.36	5167
20-30	62.50	65.72	992
30-40	21.25	41.25	80
> 40	0.00	20.00	5

Table 2: EM accuracy of the NAR LightConv Pointer (distilled) vs AR LightConv Pointer distilled across different target length buckets along with the number of instances in each bucket on the TOP dataset.

4.3 Analysis

Ablation experiments We compare the modifications proposed by this work (LightConv, Conv length prediction module and Mask everything strategy) with the original model proposed in Ghazvininejad et al. (2019) in table 1. The motivations for each modification was already discussed in sub-section 2.1. Our mean EM accuracy results based on 3 trials show the significance of techniques proposed in this paper especially for longer target sequences.

Errors by length It is known that non-autoregressive models have difficulty at larger sequence lengths (Ghazvininejad et al., 2019). In table 2, we show our model's accuracy in each respective length bucket on the TOP dataset. We see that the AR and NAR model follow a similar distribution of errors, however the NAR model seems to error at a higher rate for the longer lengths.

Knowledge Distillation Following prior work (Ghazvininejad et al., 2019; Zhou et al., 2020), we train our model with sequence-level knowledge distillation (Kim and Rush, 2016). We train our system on data generated by the current SOTA autoregressive models BART (Lewis et al., 2019; Aghajanyan et al., 2020). In table 3 we show the impact of knowledge distillation in our task on both the non-autoregressive and autoregressive variants of LightConv Pointer. These results support prior work in machine translation for distillation of au-

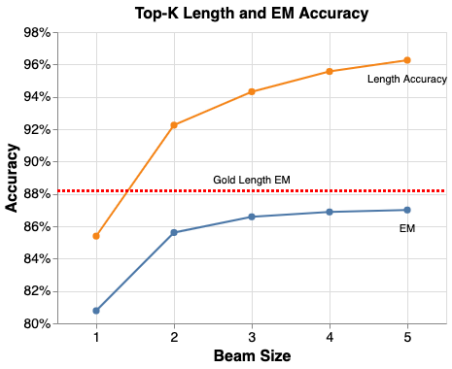


Figure 6: Distilled NAR LightConv Pointer Top-K accuracy for exact match (EM) accuracy (blue) and Top-K length accuracy (orange), as well as the EM accuracy with gold length (dotted red line) for the TOP dataset.

Model	TOP	DSTC2	SNIPS
BART (Teacher Model)	87.10	89.06	91.00
Distilled NAR LightConv Pointer	80.89	88.16	81.71
Distilled AR LightConv Pointer	81.53	88.21	80.29

Table 3: EM accuracy of various models leveraging KD from the teacher BART model on the TOP dataset.

toregressive teachers to non-autoregressive models showing distillation improving our models on TOP and SNIPS, however we notice minimal changes on DSTC2.

The importance of length prediction An important part of our non-autoregressive model is length prediction. In figure 6, we report exact match accuracy @ top k beams and length accuracy @ top k beams (where top K refers to whether the correct answer was in the top K predictions) for the TOP dataset. We can see a tight correlation between our length accuracy and exact match accuracy, showing how our model is bottle necked by the length prediction.

Providing gold length as a feature, led to an exact match accuracy of **88.20%** (shown in red on figure 6), an absolute 7.31 point improvement over our best result with our non-autoregressive LightConv Pointer.

5 Related Work

Non-autoregressive Decoding Recent work in machine translation has made a lot of progress in fully non-autoregressive models (Gu et al., 2017; Ma et al., 2019; Ghazvininejad et al., 2020a; Saharia et al., 2020) and parallel decoding (Lee et al., 2018; Ghazvininejad et al., 2019; Gu et al., 2019; Ghazvininejad et al., 2020b; Kasai et al., 2020). While many advancements have been made in machine translation, we believe we are the first to explore the non-autoregressive semantic parsing setting. In our work, we extend the CMLM to work for semantic parsing. We make two important adjustments: first, we use a different masking approach where we mask everything and do one-step generation. Second, we note the importance of the length prediction task for parsing and improve the length prediction module in the CMLM.

Seq2Seq For Semantic Parsing Recent advances in language understanding have lead to increased reliance on seq2seq architectures. Recent work by Rongali et al. 2020; Aghajanyan et al. 2020, showed the advantages from using a pointer generator architecture for resolving complex queries (e.g. composition and cross domain queries) that could not be handled by word tagging models. Since we target the same task, we adapt their pointer decoder into our proposed architecture. However, to optimize for latency and compression we train CNN based architectures (Desai et al. 2020 and Wu et al. 2019b) to leverage the inherent model parallelism compared to the BiLSTM model proposed in Aghajanyan et al. 2020 and more compression compared to the transformer seq2seq baseline proposed in Rongali et al. 2020. To further improve latency we look at parallel decoding through non-autoregressive decoding compared to prior work leveraging autoregressive models.

6 Conclusion

This work introduces a novel alternative to autoregressive decoding and efficient encoder-decoder architecture for semantic parsing. We show that in 3 semantic parsing datasets, we are able to speed up decoding significantly while minimizing accuracy regression. Our model is able to generate parse trees competitive with state of the art autoregressive models with significant latency savings, allowing complex NLU systems to be delivered on edge devices.

There are a couple of limitations of our proposed model that naturally extend themselves to future work. Primarily, we cannot support true beam decoding, we decode a single prediction for each length prediction however there may exist multiple beams for each length prediction. Also for longer parse trees and more complex semantic parsing systems such as session based understanding, our NAR decoding scheme could benefit from multiple

iterations. Lastly, though we explored models without pre-training in this work, recent developments show the power of leveraging pre-trained models such as RoBERTa and BART. We leave it to future work to extend our non-autoregressive decoding for pre-trained models.

Acknowledgements

We would like to thank - Sandeep Subramanian (MILA), Karthik Prasad (Facebook AI), Arash Einolghozati (Facebook) and Yinhan Liu for the helpful discussions.

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 *EMNLP/IJCNLP*.

Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, et al. 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *arXiv preprint arXiv:1805.10190*.

Shrey Desai, Geoffrey Goh, Arun Babu, and Ahmed Aly. 2020. Lightweight convolutional representations for on-device natural language processing. *arXiv preprint arXiv:2002.01535*.

Arash Einolghozati, Panupong Pasupat, Sonal Gupta, Rushin Shah, Mrinal Mohit, Mike Lewis, and Luke Zettlemoyer. 2018. Improving semantic parsing for task oriented dialog. In *Conversational AI Workshop at NeurIPS 2018*.

Marjan Ghazvininejad, Vladimir Karpukhin, Luke Zettlemoyer, and Omer Levy. 2020a. Aligned cross entropy for non-autoregressive machine translation. *arXiv preprint arXiv:2004.01655*.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. 2019. Mask-predict: Parallel decoding of conditional masked language models.

Marjan Ghazvininejad, Omer Levy, and Luke Zettlemoyer. 2020b. Semi-autoregressive training improves mask-predict decoding. *arXiv preprint arXiv:2001.08785*.

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 OK Li, and Richard Socher. 2017. Non-autoregressive neural machine translation. *arXiv preprint arXiv:1711.02281*.

Jiatao Gu, Changhan Wang, and Junbo Zhao. 2019. Levenshtein transformer. In *Advances in Neural Information Processing Systems*, pages 11179–11189.

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.

Dilek Hakkani-Tür, Gökhan Tür, Asli Celikyilmaz, Yun-Nung Chen, Jianfeng Gao, Li Deng, and Ye-Yi Wang. 2016. Multi-domain joint semantic frame parsing using bi-directional rnn-lstm. In *Interspeech*, pages 715–719.

Matthew Henderson, Blaise Thomson, and Jason D. Williams. 2014. The second dialog state tracking challenge. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 263–272, Philadelphia, PA, U.S.A. Association for Computational Linguistics.

Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. 2019. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1314–1324.

Jungo Kasai, James Cross, Marjan Ghazvininejad, and Jiatao Gu. 2020. Parallel machine translation with disentangled context transformer. *arXiv preprint arXiv:2001.05136*.

Yoon Kim and Alexander M Rush. 2016. Sequence-level knowledge distillation. *arXiv preprint arXiv:1606.07947*.

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

Jason D. Lee, Elman Mansimov, and Kyunghyun Cho. 2018. Deterministic non-autoregressive neural sequence modeling by iterative refinement. In *Proc. of EMNLP*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension.

Xuezhe Ma, Chunting Zhou, Xian Li, Graham Neubig, and Eduard Hovy. 2019. Flowseq: Non-autoregressive conditional sequence generation with generative flow. *arXiv preprint arXiv:1909.02480*.

Kenton Murray, Jeffery Kinnison, Toan Q. Nguyen, Walter Scheirer, and David Chiang. 2019. Auto-sizing the transformer network: Improving speed, efficiency, and performance for low-resource machine translation. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 231–240, Hong Kong. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. 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 8024–8035. Curran Associates, Inc.

Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. 2017. Regularizing neural networks by penalizing confident output distributions. *arXiv preprint arXiv:1701.06548*.

Subendhu Rongali, Luca Soldaini, Emilio Monti, and Wael Hamza. 2020. Don't parse, generate! a sequence to sequence architecture for task-oriented semantic parsing. *arXiv preprint arXiv:2001.11458*.

Chitwan Saharia, William Chan, Saurabh Saxena, and Mohammad Norouzi. 2020. Non-autoregressive machine translation with latent alignments. *arXiv preprint arXiv:2004.07437*.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. 2015. Rethinking the inception architecture for computer vision.

Elan van Biljon, Arnu Pretorius, and Julia Kreutzer. 2020. On optimal transformer depth for low-resource language translation.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. 2019a. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10734–10742.

Felix Wu, Angela Fan, Alexei Baevski, Yann N. Dauphin, and Michael Auli. 2019b. Pay less attention with lightweight and dynamic convolutions.

Chenwei Zhang, Yaliang Li, Nan Du, Wei Fan, and Philip S Yu. 2018. Joint slot filling and intent detection via capsule neural networks. *arXiv preprint arXiv:1812.09471*.

Victor Zhong, Caiming Xiong, and Richard Socher. 2018. Global-locally self-attentive encoder for dialogue state tracking. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1458–1467, Melbourne, Australia. Association for Computational Linguistics.

Chunting Zhou, Jiatao Gu, and Graham Neubig. 2020. Understanding knowledge distillation in non-autoregressive machine translation. In *International Conference on Learning Representations*.