Improving Top-K Decoding for Non-Autoregressive Semantic Parsing via Intent Conditioning

Geunseob (GS) Oh, Rahul Goel, Chris Hidey,
Shachi Paul, Aditya Gupta, Pararth Shah, Rushin Shah
Google
{ohgs,goelrahul}@google.com

Abstract

Semantic parsing (SP) is a core component of modern virtual assistants like Google Assistant and Amazon Alexa. While sequence-to-sequence based auto-regressive (AR) approaches are common for conversational SP, recent studies (Shrivastava et al., 2021) employ non-autoregressive (NAR) decoders and reduce inference latency while maintaining competitive parsing quality. However, a major drawback of NAR decoders is the difficulty of generating top-*k* (i.e., *k*-best) outputs with approaches such as beam search. To address this challenge, we propose a novel NAR semantic parser that introduces intent conditioning on the decoder. Inspired by the traditional intent and slot tagging parsers, we decouple the top-level intent prediction from the rest of a parse. As the top-level intent largely governs the syntax and semantics of a parse, the intent conditioning allows the model to better control beam search and improves the quality and diversity of top-*k* outputs. We introduce a hybrid teacher-forcing approach to avoid training and inference mismatch. We evaluate the proposed NAR on conversational SP datasets, TOP & TOPv2. Like the existing NAR models, we maintain the $O(1)$ decoding time complexity while generating more diverse outputs and improving top-3 exact match (EM) by 2.4 points. In comparison with AR models, our model speeds up beam search inference by 6.7 times on CPU with competitive top-*k* EM.

Query: what is going on this weekend?	
Label:	[in:get_event [sl:date_time this weekend]]
(Failure 1: repeated tokens)	
	[in:get_event [sl:date_time this weekend] [sl:date_time this weekend]]
	(Parse with the repeated slot *sl:data_time* and leaf nodes *this* and *weekend*)
(Failure 2: invalid syntax)	
	[in:get_event [sl:date_time
	(Incomplete parse; missing ']]')

Table 1: Examples of possible failures from the limited beam search of the existing NAR semantic parsers.

1 Introduction

Neural sequence models are widely used for the task of conversational semantic parsing, which converts natural language utterances to machine-understandable meaning representations. Recent approaches (Chen et al., 2020; Rongali et al., 2020; Lialin et al., 2020; Aghajanyan et al., 2020a; Shrivastava et al., 2021) combine Transformer-based sequence models (Vaswani et al., 2017; Devlin et al., 2019) and Pointer Generator Networks (Vinyals et al., 2015; See et al., 2017). A vast majority of semantic parsing approaches (Gupta et al., 2018; Chen et al., 2020; Rongali et al., 2020; Lialin et al., 2020; Aghajanyan et al., 2020a; Yin et al., 2021) employ autoregressive (AR) decoders to generate structured output frames for the quality of output parses. During the AR decoding, output tokens are generated sequentially, conditioned on all previously generated tokens. As a result, the decoding time (i.e. inference latency) increases linearly with the decoding length. This not only limits the capacity to accommodate larger language models but may also degrade the user experience of intelligent conversational assistants (e.g. Google Assistant, Amazon Alexa) due to the high inference latency.

On the contrary, non-autoregressive (NAR) decoders are capable of parallel decoding and thus allow much faster inference. Compared to AR decoders, which perform O(n) decoding steps, the NAR decoding is typically achieved in either $O(1)$ (Babu et al., 2021; Shrivastava et al., 2021) or O($\log(n)$) (Zhu et al., 2020) steps. A recent study (Babu et al., 2021) built NAR parsers and reduced the latency up to 81% compared to AR parsers. Another study by Shrivastava et al. (2021) proposed a NAR semantic parser called Span Pointer Network and cut the latency by 8.5-10x on CPU while achieving comparable performance to their AR benchmarks. All these studies suggest that the NAR decoders have significant latency benefits.

Although the recent NAR semantic parsers have decreased the performance gap from AR parsers on the exact match metric, one of the main limitations of NAR models still remain unsolved: beam search and top-k outputs. AR models leverage beam search algorithms (Wiseman and Rush, 2016), which are especially effective at producing top-k outputs (Wiseman and Rush, 2016; Gupta et al., 2018). This is possible since the beam search allows AR models to dynamically sort out less-probable candidate parses and adjust their decoding lengths. As a result, AR models are capable of generating diverse high-quality output parses.

On the other hand, the existing NAR parsers (Babu et al., 2021; Shrivastava et al., 2021) cannot employ the AR beam search as the output tokens of the NAR parsers are generated independently from each other. Instead, the existing NAR parsers perform a beam search by generating k candidate frame lengths. For each frame length, a single parse is produced, resulting in a total of k parses.

However, we find that the existing NAR beam search tends to produce duplicates of the most probable output parse. As exemplified in Table 1, top-k beam outputs often include outputs with repeated tokens and parses with invalid syntax (e.g., truncated or extended versions of the most-probable parse) rather than diverse output parses.

The ability to generate diverse top-k parses is important for modern conversational assistants. Table 2 presents an example that demonstrates how a query may correspond to different parses depending on the context. The diverse semantic parses can be leveraged in a downstream component that has more contextual information. An additional re-ranking module can also be employed to select the most relevant semantic parse.

This work focuses on improving the quality of top-k outputs of NAR conversational semantic parsing. Our idea is to leverage the fact that the top-level intent of a semantic parse mainly determines the syntax and semantics of the parse. This is the key driving point of the classic intent classification and slot tagging (Liu and Lane, 2016) based parsers, which were widely used traditional semantic parsers. We build our model upon the existing

Query: avoid bridges on my route	
Parses from the Proposed NAR	
(1a)	[in:update_directions [sl:path_avoid [in:get _location [sl:category_location bridges]]]]
(2a)	[in:get_directions [sl:path_avoid [in:get _location [sl:category_location bridges]]]]
Parses from the Baseline NAR	
(1b)	[in:update_directions [sl:path_avoid [in:get _location [sl:category_location bridges]]]]
(2b)	[in:update_directions [sl:path_avoid [in:get _location [sl:category_location bridges

Table 2: Examples of beam outputs. The parses produced by the proposed NAR are both valid depending on the context. If the query was made after the user already had set a path, parse 1a is more relevant, else 2a. In contrast, the baseline NAR only produced one valid parse. Parse 2b is an invalid duplicate of parse 1b.

NAR (i.e., frame length conditioned NAR), but decouple the prediction of the top-level intent from the output. We leverage the conditional dependencies of the frame length and parse on the top-level intent for diverse top-k outputs.

We evaluate the proposed NAR model on two datasets: TOP (Gupta et al., 2018) and TOPv2 (Chen et al., 2020) and demonstrate that we further close the gap between AR and NAR parsers. Most importantly, we show that the intent conditioning improves the quality and diversity of the top-k parses and the total inference time of our NAR beam search is 6.7x times shorter than the AR baseline on CPU.

2 Related Work

2.1 Non-autoregressive Semantic Parsing

Non-autoregressive sequence models have been an active research area across different fields of natural language processing for their fast inference speeds. While various designs of NAR decoders exist, recent works in machine translation utilize models with iterative refinements (Lee et al., 2018; Ghazvininejad et al., 2019, 2020), insertion-based models (Stern et al., 2019), and latent alignment methods (Libovický and Helcl, 2018; Chan et al., 2020; Saharia et al., 2020).

The task of semantic parsing (SP) is similar to the machine translation task as they both translate input sentences from one representation to another. For this reason, recent studies in SP have adopted modeling techniques from machine translation. Zhu et al. (2020) leveraged the insertion-based seq2seq models for NAR semantic parsing and reduced the decoding time from $O(n)$ to $O(log(n))$ while matching the performance of AR models. Babu et al. (2021); Shrivastava et al. (2021) applied an iterative refinement method *Mask-Predict* to semantic parsing and brought the time complexity further down to $O(1)$ while performing comparably to the baseline AR models. As opposed to the original Mask-Predict model that performs multiple iterations of re-masking and prediction (Ghazvininejad et al., 2019), they only perform a single iteration of the masking and output prediction as they claim that task-oriented SP does not benefit much from iterative refinements (Babu et al., 2021; Shrivastava et al., 2021). In addition to the recent sequence-to-sequence NAR semantic parsers, traditional intent and slot filling model (Mesnil et al., 2014) is another approach with non-autoregressive decoding.

2.2 Baseline NAR

Span Pointer Network (Shrivastava et al., 2021), which is one of the two recent works that leveraged the mask-predict algorithm, showed the competitive performance to the AR parsers while significantly reducing their latency. For this reason, we utilize Span Pointer Network as a basis of our work and accordingly refer to it as *baseline NAR* to distinguish the NAR semantic parser we propose. It should be noted that we replicated Span Pointer Network as it is not publicly available.

The baseline NAR is built with a pre-trained encoder, a frame length module, and a Transformer decoder. The encoder is a language model such as BERT (Devlin et al., 2019) or RoBERTa (Liu et al., 2019) that takes input queries x and outputs their encoded representations e.

$$e_{1:l} = \text{Encoder}(x_{1:l}), \tag{1}$$

where l indicates the length of the source (i.e., input query). The frame length module takes the encoded representations and predicts the length of the frame (i.e., target parse) n followed by the generation of n mask tokens.

$$\begin{aligned} n &= \text{FrameLengthModule}(e_{1:l}). \\ [\text{MASK}]_{1:n} &= \text{MaskCreation}(n). \end{aligned} \tag{2}$$

The transformer decoder takes the N mask tokens as inputs and produce h for each token in the masked sequence.

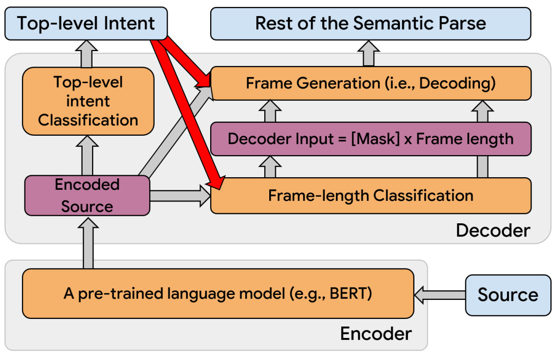
Top-level Intent
Rest of the Semantic Parse
Top-level intent Classification
Frame Generation (i.e., Decoding)
Decoder Input = [Mask] x Frame length
Encoded Source
Frame-length Classification
Decoder
A pre-trained language model (e.g., BERT)
Source
Encoder

Figure 1: The proposed model architecture. By decoupling the top-level intent prediction from the output prediction, we perform conditional generation of the frame length and parse upon the top-level intent.

$$h_{1:n} = \text{Decoder}([\text{MASK}]_{1:n}; e_{1:l}). \tag{3}$$

Lastly, the pointer-generator mapping layer is used to convert h to the target frame y as follows.

$$y_{1:n} = \text{PTR}(h_{1:n}; e_{1:l}), \tag{4}$$

where y_k is either a token from the target vocab that consists of parse symbols (e.g., intents & slots) or copy of a source token. The output tokens are generated in parallel as they are conditionally independent given the source and frame length.

2.3 Beam search for AR vs baseline NAR

AR decoders produce multiple candidate output parses per source by employing the beam search algorithm (Wiseman and Rush, 2016). During the beam search with width k, the k most probable candidates are kept at each decoding step and less probable candidates are sorted out. The decoding process finishes when a special "[END]" token is encountered. For a decoding length n, the score of an AR parse is computed as follows.

$$S_{AR}(y) = p(y|x) = \prod_{j=1}^{n} p(y_j|y_{1:j-1}, x). \tag{5}$$

Since the AR decoders consider multiple candidate tokens y_j at each decoding step, they can generate a large number of unique parses.

In comparison, the baseline NAR only produces a single output parse per frame length. The baseline NAR mimics the beam search by generating top-k frame lengths, which results in k outputs. The output parse is scored using the joint probability of the frame length and output parse as follows.

$$S_{NAR,baseline}(y) = p(y, n|x) = p(y|n, x) \cdot p(n|x) = \prod_{j=1}^{n} p(y_j|n, x) \cdot p(n|x), \tag{6}$$

where $S_{NAR,baseline}(y)$ denotes the score of the baseline NAR output. $p(y|n, x)$ is obtained using the conditional independence of the output tokens given the frame length n and source x.

3 Proposed Approach

To mitigate the aforementioned limitation of the baseline NAR, we propose Intent-conditioned Non-autoregressive Neural Semantic Parser. Our approach is motivated by traditional semantic parsers such as Liu and Lane (2016), which performs an intent classification followed by slot tagging. Another motivation comes from an observation that the top-level intent of the parse largely governs the syntax and semantics of the rest of the parse.

Our idea is to decouple the top-level intent prediction from the output prediction as shown in Figure 1. This builds explicit dependencies of the frame length and the rest of the frame on the top-level intent. As a result, the joint probability of output and frame length is expressed as follows.

$$p(y, n|x) = p(y_{2:n}|n, y_1, x) \cdot p(n|y_1, x) \cdot p(y_1|x). \tag{7}$$

This facilitates effective conditional generation of the length and output, while keeping the decoding time complexity as $O(1)$; the number of decoding steps does not scale with the output length.

3.1 Intent Conditioning

The proposed NAR utilizes a pre-trained language model to obtain encoded source representation e. Then, it performs the top-level intent prediction, which is formulated as a multiclass classification of the size of the dimensionality of intent vocab.

$$\text{logits}(y_1) = \text{IntentModule}(e_{1:l}), \tag{8}$$

where y_1 refers to the top-level intent of the semantic parse or the first token of the parse. As the top-level intent prediction is decoupled, the intent module works with a smaller vocabulary size. Compared to the output vocabulary that combines a target and copy index vocabulary, the intent vocabulary is approximately 4 times smaller with a source length 32 on TOP dataset. We use a randomly initialized Transformer as the intent module.

We use the logits of the top-level intent, as opposed to the discrete intent class variable, as the inputs to all subsequent modules. This is because the dense logits convey information about the uncertainty of the intent prediction for each intent class and help the subsequent modules to condition on richer information. We found that this resulted in better performance than the model conditioned on the 1-dimensional discrete intent.

Subsequent to the top-level intent prediction, the frame length module takes the logits of the intent y_1 and produces the decoding length followed by the creation of mask tokens. After that, the initial mask tokens together with the frame length pass through a positional encoding layer to compute the encoded mask tokens, $[\text{MASK}]_{2:n}$.

$$\begin{gathered} n - 1 = \text{FrameLengthModule}(\text{logits}(y_1); e_{1:l}), \\ [\text{MASK}]_{2:n} = \text{PosEncoding}(\text{MaskCreation}, n). \end{gathered} \tag{9}$$

The rest of the frame is generated using parallel decoding followed by the source token mapping via the pointer-generator mapping layer.

$$\begin{gathered} h_{2:n} = \text{Decoder}([\text{MASK}]_{2:n}; \text{logits}(y_1), e_{1:l}), \\ y_{2:n} = \text{PTR}(h_{2:n}; e_{1:l}), \quad y_{1:n} = \text{cat}(y_1, y_{2:n}). \end{gathered} \tag{10}$$

Finally, the output parse is obtained by concatenating the top-level intent and rest of the frame.

3.2 Training Objective of the Proposed NAR

The loss is a weighted sum of top-level intent classification loss $\mathcal{L}_{int}$, frame length classification loss $\mathcal{L}_{len}$, and output loss $\mathcal{L}_{out}$ as follows.

$$\mathcal{L} = \mathcal{L}_{out} + \lambda_{len} \cdot \mathcal{L}_{len} + \lambda_{int} \cdot \mathcal{L}_{int}. \tag{11}$$

We jointly optimize the three loss terms and employ label smoothing (LS) (Szegedy et al., 2016) as a regularizer to penalize overconfident predictions by computing negative log-likelihood (NLL) between the smoothed one-hot labels and predictions. That is, $\mathcal{L}_{int} = \text{NLL}(\text{LS}(y_{1,label}), y_{1,pred})$, $\mathcal{L}_{len} = \text{NLL}(\text{LS}(n_{label}), n_{pred})$, and $\mathcal{L}_{out} = \text{NLL}(\text{LS}(y_{2:n,label}), y_{2:n,pred})$.

3.3 Beam Search with the Proposed NAR

The proposed intent-conditioned NAR produces multiple output parses per source via the sequential conditioning on the top-level intent and length. Precisely, the model first computes top-$k1$ top-level intents. For each top-level intent, the length model outputs top-$k2$ frame lengths. As a result, $k1 \cdot k2$ pairs of top-level intents and lengths are produced. Finally, the decoder generates a single parse per pair, yielding a total of $k1 \cdot k2$ output parses.

The proposed NAR performs a single decoder pass regardless of the beam size, $k1 \cdot k2$. This is possible as we cast the $k1 \cdot k2$ pairs as the batch dimension. As a result, the time complexity of the beam search remains constant, to the extent of the memory capacity. The memory increases linearly to the beam size.

The intent conditioning provides explicit control of the top-level intents via the selection of top-$k1$ unique intents. It also builds dependencies of the frame length and output on the top-level intents. We find that these help the model to improve diversity and quality of top-k output parses. To further validate the hypothesis, we devise two experiments to (1) identify the potential (i.e., upper limit of the accuracy) of the proposed NAR beam search and (2) examine the diversity and quality of the output parses compared to the baseline NAR beam search.

3.4 Hybrid Teacher Forcing

During our initial attempts to inspect the proposed model, we discovered that the inference performance is unstable. This was due to a discrepancy in the computation of the top-level intent during the training and inference. Recall that we designed the outputs of the top-level intent module to be logits (see Equation 8), as opposed to a one-dimensional or one-hot representation. The idea is to deliver richer information to downstream modules (e.g., length module and decoder). While we observed that this idea improves the exact match for the proposed NAR architecture, it makes the inference unstable. This was because we needed to sample the top-level intents, as (smoothed) one-hot vectors, and convert them back to the logits. Conversely, in the training, we do not sample top-level intents and instead use the direct outputs (dense logits) from the top-level intent module.

We first attempted to resolve the problem with teacher-forcing of the top-level intents. While this stabilized the inference, the training became unstable due to sparsity of the teacher logits and mismatch between the model and teacher intent logits.

We address this problem by leveraging a sampling strategy depicted in Figure 2. The idea is to uniformly sample logits from a pair of teacher and model logits as follows.

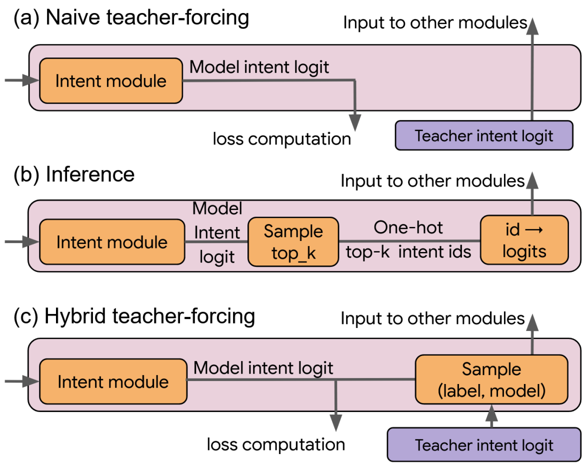

Figure 2: The computation of the top-level intent. (a) During training via naive teacher forcing, label intent logits are fed to the length module and decoder. (b) In the inference, top-k intents are sampled from the model logits and fed to the other modules. (c) Hybrid teacher-forcing uses both label and model logits for the training.

$$\text{logits}(y_1) \sim \mathcal{U}(\text{logits}(y_{1,\text{label}}), \text{logits}(y_{1,\text{model}})). \tag{12}$$

This essentially exposes the model, during the training, to the hidden space where the intents are sampled at the inference time. Without the hybrid teacher-forcing, the inference-time hidden space may not be well explored during the training, thus becomes source of instability.

We name this strategy as *hybrid teacher forcing* to reflect that the model leverages both teacher and model top-level intent logits. This sampling technique may be seen as a non-autoregressive variant of the scheduled sampling (Bengio et al., 2015). While the proposed model architecture is responsible for the performance improvement, the hybrid teacher-forcing technique helped our model to achieve the improvement by stabilizing the training and inference.

3.5 Scoring Beam Search Outputs

A scoring method is used to select the top-k parses from the $k1 \cdot k2$ pairs. We tested three methods. The first method S_1 uses the joint log probability of the top-level intent and frame length as the score.

$$S_1 = \log(p(n, y_1|x)) = \log(p(n|y_1, x) \cdot p(y_1|x)). \tag{13}$$

The second method S_2 uses the joint log probability of the output, length, and top-level intent.

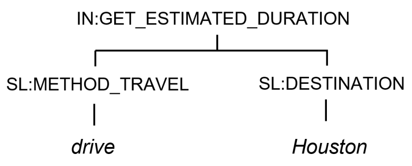

Figure 3: Decoupled representation of the semantic parse for a query "How long is my drive to Houston?".

$$S_2 = \log(p(y_{1:n}, n|x)) = \log\big(p(y_{2:n}|n, y_1, x) \cdot p(n|y_1, x) \cdot p(y_1|x)\big). \tag{14}$$

The last scoring method S_3 combines the joint probability $p(y_{1:n}, n|x)$ and a length penalty.

$$\begin{aligned} lp(y) &= ((5 + l)/6)^{\alpha}, \\ S_3 &= S_2(y)/lp(y), \end{aligned} \tag{15}$$

where lp is the length penalty (Wu et al., 2016).

Depending on $k1$, $k2$, and scoring method, the top-*k* outputs may consist of parses with multiple distinct lengths and intents or parses with a single intent with multiple lengths.

3.6 Semantic Parse Representation

We represent semantic parses using the decoupled tree representation (Aghajanyan et al., 2020a) that is an extension of the compositional tree representation (Gupta et al., 2018). An example is depicted in Figure 3. Likewise, another query "What is happening in Boston on New Year's Eve" is semantically parsed into "*[in:get_event [sl:location Boston] [sl:date_time on New Year's Eve]]*".

An example of the decoupled tree representation is *Index form* that replaces the copy tokens with the indices to the corresponding source tokens. In this sense, the above semantic parse is represented as "*[in:get_event [sl:location 4] [sl:date_time 5 6 7 8]]*". Together with the pointer network, this significantly reduces the size of output vocabulary (Rongali et al., 2020). Another representation is *Span form* introduced in (Shrivastava et al., 2021). As opposed to specifying all indices of spans of copy tokens, the span form uses the start and end indices of the spans. That is, "*[in:get_event [sl:location 4 4] [sl:date_time 5 8]]*". The benefits of the span form include shorter target lengths and fixed number of leaf nodes that helps decoupling the syntax from the semantics. In addition, the number of valid frame length classes are halved as the frame lengths of parses in the span form are always even.

4 Experiments

4.1 Datasets

To quantify the benefits of our NAR parsers over the baseline NAR, we utilize two conversational semantic parsing datasets. The first dataset is TOP (Task Oriented Parsing) (Gupta et al., 2018), a collection of human-generated queries in English and the corresponding semantic parses that are represented as hierarchical trees. Another public dataset we explore is TOPv2 (Chen et al., 2020), which is an extension of TOP with six more domains.

4.2 Evaluation Metrics

We mainly utilize *exact match* (EM) to evaluate quality of both greedy decoding and beam decoding output. EM is defined as the percentage of queries whose label parses are correctly predicted.

Inference time is the metric we use to quantify the latency benefit of our model. We measure the model inference time on a TPU (Jouppi et al., 2017) from the Google Cloud TPUv2, a Nvidia Quadro P1000 GPU, and a CPU of the Intel Cascade Lake CPU platform with 8GB RAM. All measurements were obtained using the same machine on the same process. We used batch size 1 and computed latency on 1000 examples from the val set of the TOP dataset to reduce measurement noise and minimize the variance. We report per-example average latencies of these runs.

In addition, *diversity* of the output parses is computed to quantify the capability of the proposed NAR approach at producing distinct top-*k* outputs. We mainly compute two metrics: (1) number of unique top-level intents in top-*k* output parses and (2) the distinct n-grams (Li et al., 2016), which are commonly used diversity metrics in natural language processing (Vijayakumar et al., 2018; Xu et al., 2018). Specifically, we compute percentage of distinct n-grams by counting unique n-grams in top-*k* outputs, dividing the counts by the total number of output tokens, and scaling it by 100.

4.3 Implementation details

4.3.1 Encoder

We use pre-trained $\mathrm{BERT_{BASE}}$ (L12/H768) as the encoders for all AR, baseline NAR, and proposed NAR for the experiments. We use uncased versions of them along with lowercased datasets as they generally performed better than the cased models with cased datasets. We leave it to future work to quantify the performance with other pre-trained language models such as RoBERTa (Liu et al., 2019).

4.3.2 Decoder

We primarily compare the performances of two AR baseline decoders, the baseline NAR decoder, and the proposed NAR decoder. The two AR decoders are identical except that one is trained on the label parses represented in the index form and the other in the span form. All NAR decoders were trained on the label parses in the span form. We used the same decoder architecture for all models across all experiments. As discussed earlier, the baseline NAR is a replication of the NAR model proposed by Shrivastava et al. (2021). Specifically, we use the vanilla version that does not utilize the R3F loss (Aghajanyan et al., 2020b).

Unlike Shrivastava et al. (2021) and Babu et al. (2021) that use an MLP or CNN length module, our frame length module is a randomly initialized Transformer decoder; we found that it performs better as the attention layers perform more complex weighing on the encoder output. We used the same length module for both baseline and proposed NAR models. Similarly, we use a randomly initialized Transformer decoder for the top-level intent prediction. We point out that we use two separate Transformers, one for the top-level intent, the other for the frame length module, as opposed to a shared Transformer. Further details of the model parameters are described in the appendix.

4.3.3 Hyperparameters

We observed that the ratio of the loss terms is important. In general, higher top-level intent penalty and moderate frame length penalty resulted in better accuracy. We use the hybrid teacher forcing to train the proposed NAR for the experiments presented in this paper. We trained the models using Adam optimizer (Kingma and Ba, 2014) with exponential learning rate decay. Additionally, we use 1000 learning rate warmup steps. We also apply dropout to the source embeddings and the Transformer decoder for better generalization. The details of the parameters are described in the appendix.

5 Results

We show that the proposed intent-conditioned NAR further closes the gap between AR and NAR SP. Most importantly, our model greatly improves the quality of top-k output parses while maintaining the $O(1)$ decoding time of the baseline NAR.

Model	Latency			EM	
	TPU	GPU	CPU	TOPv1	TOPv2
Base.NAR	1.00x	1.00x	1.00x	82.56	84.86
Prop.NAR	1.01x	1.11x	1.09x	83.11	85.22
AR, index	2.23x	11.01x	4.75x	83.43	85.40
AR, span	2.26x	11.01x	4.58x	83.40	85.56

Table 3: Greedy decoding results with BERT encoder. Latency numbers are measured with TOPv1 dataset and relative to the baseline NAR latency on TPU.

We first present the greedy decoding results. We then investigate the upper-bound beam search performance of the baseline and proposed NAR. Next, we share the beam search results that quantify the quality and diversity of top-k output parses. Lastly, we share the results of the comparison of various beam scoring methods. We point out that we used the identical encoder & decoder architectures and model parameters for all AR & NAR models. In addition, the identical frame length module is used for both baseline and proposed NAR models.

5.1 Greedy Decoding

Table 3 presents the greedy decoding results. At each decoding step of AR models, the most probable token is selected. This repeats until an END token is produced. In NAR greedy decoding, beam width of 1 is used for both frame length and top-level intent. No scoring method is used.

The results indicate that our model improves the performance of baseline NAR by 0.4-0.5 EM while matching the latency of the prior NAR approach. Compared to AR baselines, our model cuts the greedy decoding inference time (i.e., latency of the semantic parser including both encoder and decoder) by 2.2x on TPU, 10x on GPU, and 4.2-4.4x on CPU.

5.2 Potential Impact of Intent Conditioning for Beam Search

We designed an experiment to investigate the potential benefits of the proposed beam search. We aimed to empirically quantify the upper bound of the beam search performance of the baseline NAR and our intent-conditioned NAR. In the experiment, we first trained the baseline and proposed NAR parsers, then fed the oracle (i.e., gold) frame length to the baseline NAR or the oracle top-level intent to the proposed NAR during the inference. To ensure

Models	Oracle EM
Variants of the baseline NAR	
Shallow decoder	83.34
Shallow decoder, Label smoothing	82.57
Deep decoder	83.73
Deep decoder, Label smoothing	83.03
Best of the baseline NAR	**83.73**
Variants of the proposed NAR	
Shallow decoder	86.22
Shallow decoder, Label smoothing	86.40
Deep decoder	86.89
Deep decoder, Label smoothing	85.66
Best of the proposed NAR	**86.89**

Table 4: Empirical upper-bound of beam search with NARs. To obtain the oracle EMs, we feed the gold frame length for the baseline NARs in the inference. Likewise, we use the gold top-level intent for the proposed NARs.

validity of the results, we ran the experiments with different model configurations.

As shown in Table 4, the proposed NAR consistently scored higher oracle EM across various model configurations. It should be noted that the proposed NARs only use the gold top-level intent (i.e., the gold length is not used for the inference with the proposed NARs). Table 5 confirms these results and shows that our NAR performs more effective beam-coding and achieves higher exact match for the top-*k* output.

5.3 Beam search

In table 5, we report the beam search results on TOP dataset as top-*k* EM for $k = 1, 2, 3$. If the label parse matches with any of the top-*k* output parses, it counts towards the top-*k* EM. We also report top-level intent match (IM) percents. The top-*k* parses are selected from the $k1 \cdot k2$ beam search outputs using the scoring method 3 (Equation 15). We used the same $k1 \cdot k2$ for the fair comparison. Specifically, we used $k2 = 25$ for the baseline NAR and $k1 = 25$, $k2 = 1$ for our NAR as there exist 25 distinct length classes and 25 distinct intent classes in TOP dataset. For the proposed NAR, we observed that a higher $k2$ has minor impact on top-*k* EMs, compared to a higher $k1$. We note that the hybrid teacher forcing (Equation 12) was used for the training and helped the models achieve consistent test set accuracy.

The results demonstrate a large gap between AR and the existing NAR models in top-*k* outputs. Notably, the top-3 EM from the existing NAR model is outperformed by the baseline AR by 3.5 points. Secondly, the proposed NAR outperforms baseline NAR in top-3 EM by 2.4 and top-3 IM by 2.8 points. Third, the EM and IM gaps between the proposed and baseline NAR widen as *k* increases. Lastly, we show that the proposed NAR reduces the performance gap from the autoregressive models. The inference time of our NAR beam search is 2.5x times shorter on TPU, 8.4x times shorter on GPU, and 6.7x times shorter on CPU. Our AR baselines manage parallel compute loads better than NAR models and thus result in smaller latency gaps on TPU.

Note that a parse can have more than 1 intent (e.g., the parses shown in Table 2). In this regard, we report the total intent prediction accuracy on top of the top-level intent match (IM) as follows. The intent prediction accuracy with top-1 prediction is 93.09 (baseline NAR), 93.25 (ours), 93.15 (AR, index form), 93.65 (AR, span form). The intent prediction accuracy with top-3 predictions are 94.09 (baseline NAR), 95.68 (ours), 95.17 (AR, index form), 95.33 (AR, span form). While baseline NAR only gained 1.00% intent accuracy improvement from top-1 to top-3, the proposed NAR gained 2.43% improvements. This agrees with the EM improvements that our model offers.

5.4 Diversity

Table 6 elaborates on the diversity of the output parses with the baseline and proposed NAR models. Specifically, we compute their average numbers of unique top-level intents and distinct n-gram (Li et al., 2016) percentages in top-3 output parses. We find that top-*k* parses of the proposed NAR outperforms the baseline NAR in all diversity metrics.

We observed that the baseline NAR beam search often duplicates the most probable parse with repeated tokens and/or invalid syntax. This suggests that the frame length conditioning alone is not effective at producing top-*k* outputs (see Table 1 and 2 for examples). Table 6 quantitatively confirms the observation with lower sentence-wise and corpus-wise distinct n-gram scores of the baseline NAR, compared to the proposed NAR.

Model	Latency			EM			IM		
	TPU	GPU	CPU	top-1	top-2	top-3	top-1	top-2	top-3
Baseline NAR	1x	1x	1x	82.61	83.44	83.62	94.59	95.14	95.22
Proposed NAR	1.12x	1.65x	1.25x	83.12	85.15	86.00	94.85	97.04	98.05
AR, index form	2.78x	13.92x	8.28x	83.44	86.46	87.34	94.61	95.95	96.44
AR, span form	2.84x	13.86x	8.33x	83.47	86.22	87.12	94.67	95.82	96.22

Table 5: Beam decoding results. The proposed NAR outperforms the baseline NAR in top-3 EM by 2.4 and top-3 IM by 2.8 points. Compared to AR models, our approach cuts a beam search inference latency by 6.7 times on CPU.

Metric (higher is more diverse)	Baseline NAR	Proposed NAR
top-3 Exact Match	83.62	86.00
# of unique 1st intents in top-3	1.12	3.00
Distinct 1-gram, sentence-wise	69.84	79.92
Distinct 2-gram, sentence-wise	85.93	94.12
Distinct 1-gram, corpus-wise	25.77	40.79
Distinct 2-gram, corpus-wise	40.36	53.77

Table 6: We measure the diversities of the top-3 parses of the baseline and proposed NAR models reported in Table 5. We report sentence-wise (i.e., distinct n-grams in each parse) and corpus-wise distinct n-grams (i.e., distinct n-grams in each collection of top-3 parses).

6 Conclusion

We present a novel non-autoregressive neural semantic parser: the intent-conditioned NAR for improved top-*k* decodings. The proposed model addresses the main limitation of the recent NAR models (i.e., the limited beam search capability) by decoupling the prediction of the top-level intent from output. This builds explicit dependencies of frame lengths and output parses on top-level intents and helps NAR semantic parsers to better control beam search. The proposed NAR model further closes the performance gap from AR models by improving the quality and diversity of top-*k* outputs. We highlight that the *decoding* time complexity is $O(1)$, regardless of output length or beam width.

References

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

Armen Aghajanyan, Akshat Shrivastava, Anchit Gupta, Naman Goyal, Luke Zettlemoyer, and Sonal Gupta. 2020b. Better fine-tuning by reducing representational collapse. In *International Conference on Learning Representations*.

Arun Babu, Akshat Shrivastava, Armen Aghajanyan, Ahmed Aly, Angela Fan, and Marjan Ghazvininejad. 2021. Non-autoregressive semantic parsing for compositional task-oriented dialog. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2969–2978, Online. Association for Computational Linguistics.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. *arXiv preprint arXiv:1506.03099*.

William Chan, Chitwan Saharia, Geoffrey Hinton, Mohammad Norouzi, and Navdeep Jaitly. 2020. Imputer: Sequence modelling via imputation and dynamic programming. In *International Conference on Machine Learning*, pages 1403–1413. PMLR.

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

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

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

Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 6112–6121, Hong Kong, China. Association for Computational Linguistics.

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

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.

Norman P Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, et al. 2017. In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44th annual international symposium on computer architecture*, pages 1–12.

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

Jason Lee, Elman Mansimov, and Kyunghyun Cho. 2018. Deterministic non-autoregressive neural sequence modeling by iterative refinement. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1173–1182, Brussels, Belgium. Association for Computational Linguistics.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119, San Diego, California. Association for Computational Linguistics.

Vladislav Lialin, Rahul Goel, Andrey Simanovsky, Anna Rumshisky, and Rushin Shah. 2020. Update frequently, update fast: Retraining semantic parsing systems in a fraction of time. *arXiv preprint arXiv:2010.07865*.

Jindřich Libovický and Jindřich Helcl. 2018. End-to-end non-autoregressive neural machine translation with connectionist temporal classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3016–3021, Brussels, Belgium. Association for Computational Linguistics.

Bing Liu and Ian Lane. 2016. Attention-based recurrent neural network models for joint intent detection and slot filling. *Interspeech 2016*, pages 685–689.

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

Grégoire Mesnil, Yann Dauphin, Kaisheng Yao, Yoshua Bengio, Li Deng, Dilek Hakkani-Tur, Xiaodong He, Larry Heck, Gokhan Tur, Dong Yu, et al. 2014. Using recurrent neural networks for slot filling in spoken language understanding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(3):530–539.

Prafull Prakash, Saurabh Kumar Shashidhar, Wenlong Zhao, Subendhu Rongali, Haidar Khan, and Michael Kayser. 2020. Compressing transformer-based semantic parsing models using compositional code embeddings. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4711–4717, Online. Association for Computational Linguistics.

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

Chitwan Saharia, William Chan, Saurabh Saxena, and Mohammad Norouzi. 2020. Non-autoregressive machine translation with latent alignments. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1098–1108, Online. Association for Computational Linguistics.

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.

Bo Shao, Yeyun Gong, Weizhen Qi, Guihong Cao, Jianshu Ji, and Xiaola Lin. 2020. Graph-based transformer with cross-candidate verification for semantic parsing. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

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

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

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826.

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.

Ashwin K Vijayakumar, Michael Cogswell, Ramprasaath R Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. 2018. Diverse beam search for improved description of complex scenes. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. *Advances in Neural Information Processing Systems*, 28:2692–2700.

Sam Wiseman and Alexander M. Rush. 2016. Sequence-to-sequence learning as beam-search optimization. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1296–1306, Austin, Texas. Association for Computational Linguistics.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Jingjing Xu, Xuancheng Ren, Junyang Lin, and Xu Sun. 2018. Diversity-promoting GAN: A cross-entropy based generative adversarial network for diversified text generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3940–3949, Brussels, Belgium. Association for Computational Linguistics.

Pengcheng Yin, Hao Fang, Graham Neubig, Adam Pauls, Emmanouil Antonios Platanios, Yu Su, Sam Thomson, and Jacob Andreas. 2021. Compositional generalization for neural semantic parsing via span-level supervised attention. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2810–2823, Online. Association for Computational Linguistics.

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

A Comparisons with Other Benchmarks

We compare the AR and NAR models we used with some of the benchmarks in the literature for the TOP & TOPv2 datasets. While there are benchmarks with other pre-trained encoders, we report the prior models with pre-trained BERT encoders to be consistent with the AR and NAR models used in this work (Table 7).

Encoders	TOP	TOPv2
Non-autoregressive Models		
$\text{BERT}_{\text{BASE}}$ baseline (greedy)	82.56	84.86
$\text{BERT}_{\text{BASE}}$ **proposed** (greedy)	83.11	85.22
Autoregressive Models		
$\text{BERT}_{\text{BASE}}$ (beam size = 4) (Rongali et al., 2020)	83.13	-
$\text{BERT}_{\text{BASE}}$ (beam size = 5) (Prakash et al., 2020)	85.01	-
Transformer + $\text{BERT}_{\text{BASE}}$ (Shao et al., 2020)	82.51	-
$\text{BERT}_{\text{BASE}}$ (greedy-ours, index)	83.43	85.40
$\text{BERT}_{\text{BASE}}$ (greedy-ours, span)	83.40	85.56

Table 7: Reported performance of semantic parsers with BERT encoders on TOP and TOPv2 datasets. Greedy refers to greedy decoding (i.e., beam size 1).

We note that the two baseline AR semantic parsers used in this work are based on the AR architecture presented in Rongali et al. (2020). Our AR baselines perform comparably against the reported numbers in the literature, as shown in Table 7.

B Scoring Beam Outputs

Table 8 depicts the performance of the three scoring methods on the TOP dataset. Each scoring method sorts the beam outputs of the proposed NAR differently (Equation 13-15).

Our study indicates that the scoring method 3 works the best for our NAR model, empirically demonstrating the effectiveness of the length penalty (Wu et al., 2016) for NAR models. We observed that high length penalties (e.g., $\alpha = 2.0 - 3.0$) often corresponded to the highest EM. For fair comparison, we also applied the length penalty with $\alpha = 1.0$ for the AR models for all experiments reported in this work.

Scoring Method	Exact Match		
	top-1	top-2	top-3
S_1 (Equation 13)	83.11	84.88	85.36
S_2 (Equation 14)	83.11	84.95	85.49
S_3 (Equation 15)			
$\alpha = 1.0$	83.13	85.21	85.81
$\alpha = 3.0$	83.12	85.15	**86.00**
$\alpha = 5.0$	83.15	85.08	85.93

Table 8: Performance of the three scoring methods.

C Model Architecture Details

Here, we elaborate on the details of the model architecture and parameters used across different experiments. Table 9 specifies hyper-parameters we used to build the AR and NAR models used across different experiments.

We note that we use two separate Transformers for the frame length and intent modules. Our hypothesis is that the Transformer output vector space is different for top-level intent and frame length. We intended the frame length module to learn its best output representation as independently from intent as possible. The frame length is still conditioned on the top-level intent as in Equation 9. This way, the backprop gradient to the length module Transformer (denoted as frame_len_h in Figure 4) is separated from the gradient to the intent module. Two separate Transformer may seem to be an overkill, however, consider the number of parameters of the model (Table 9). The 134.1M params of the proposed NAR is mostly from the encoder (BERT; L12/H768), which has 110M parameters. The frame length and intent module each has about 5.7M parameters, and the remainder is for the decoder. In terms of the network size, the separate Transformer does not make much difference as opposed to an alternative architecture that has a shared Transformer for the length and intent.

Compared to the AR models, the baseline NAR has roughly 12.5% more model parameters as it includes the frame length module in addition. The proposed NAR has approximately 4.4% more model parameters than the baseline NAR due to the addition of the intent module. Figure 4 depicts the detailed model architecture for the proposed NAR. The figure includes (sub)-modules, inputs, outputs, as well as the dimensionality of them.

	AR	baseline NAR	proposed NAR
Number of parameters	114.1M	128.4M	134.1M
Number of TPUs used for training		8 Google Cloud TPUv2	
decoder (layer/hidden/head)		L4/H256/HD2	
length module (layer/hidden/head)	N/A	L8/H256/HD4	
intent module (layer/hidden/head)	N/A	N/A	L8/H256/HD4
nonlinearity		Relu	
model dropout		0.0316	
source embeddings dropout		0.0022	
λ_{length}	N/A	10	
λ_{intent}	N/A	N/A	100
optimizer		Adam	
learning rate		0.00004	
learning rate warmup steps		1000	
learning rate scheduler		Exponential Decay	
batch size		256	

Table 9: Hyper-parameters related to the models used across different experiments presented in this work.

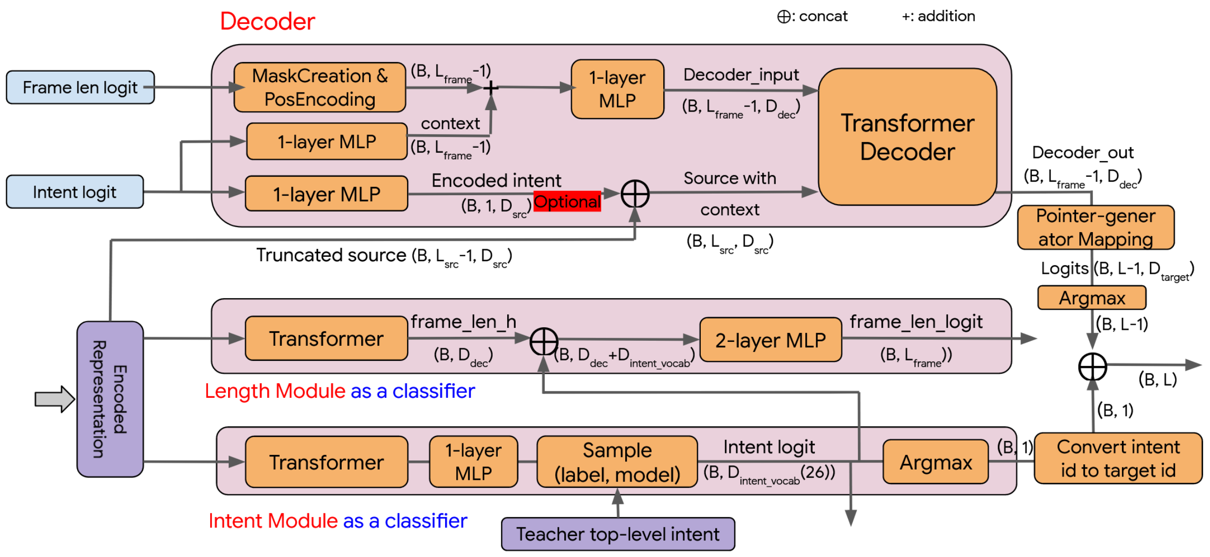

Figure 4: The detailed architecture of the proposed NAR.