Augmented Natural Language for Generative Sequence Labeling

Ben Athiwaratkun
AWS AI
benathi@amazon.com

Cicero Nogueira dos Santos
AWS AI
cicnog@amazon.com

Jason Krone
AWS AI
kronej@amazon.com

Bing Xiang
AWS AI
bxiang@amazon.com

Abstract

We propose a generative framework for joint sequence labeling and sentence-level classification. Our model performs multiple sequence labeling tasks at once using a single, shared natural language output space. Unlike prior discriminative methods, our model naturally incorporates label semantics and shares knowledge across tasks. Our framework is general purpose, performing well on few-shot, low-resource, and high-resource tasks. We demonstrate these advantages on popular named entity recognition, slot labeling, and intent classification benchmarks. We set a new state-of-the-art for few-shot slot labeling, improving substantially upon the previous 5-shot ($75.0\% \rightarrow 90.9\%$) and 1-shot ($70.4\% \rightarrow 81.0\%$) state-of-the-art results. Furthermore, our model generates large improvements ($46.27\% \rightarrow 63.83\%$) in low-resource slot labeling over a BERT baseline by incorporating label semantics. We also maintain competitive results on high-resource tasks, performing within two points of the state-of-the-art on all tasks and setting a new state-of-the-art on the SNIPS dataset.

1 Introduction

Transfer learning has been the pinnacle of recent successes in natural language processing. Large pre-trained language models are powerful backbones that can be fine-tuned for different tasks to achieve state-of-the-art performance in wide-raging applications (Peters et al., 2018; Devlin et al., 2019; Radford et al., 2019; Lewis et al., 2019; Yang et al., 2019; Liu et al., 2019).

While these models can be adapted to perform many tasks, each task is often associated to its own output space, which limits the ability to perform multiple tasks at the same time. For instance, a sentiment analysis model is normally a binary classifier that decides between class labels *"positive"* and *"negative"*, while a multi-class entailment system classifies each input as *"entail"*, *"contradict"*, or *"neither"*. This approach presents difficulty in knowledge sharing among tasks. That is, to train the model for a new task, the top-layer classifier is replaced with a new one that corresponds to novel classes. The class types are specified implicitly through different indices in the new classifier, which contain no prior information about the label meanings. This discriminative approach does not incorporate label name semantics and often requires a non-trivial amount of examples to train (Lee et al., 2020). While this transfer learning approach has been immensely successful, a more efficient approach should incorporate prior knowledge when possible.

Conditional generative modeling is a natural way to incorporate prior information and encode the output of multiple tasks in a shared predictive space. Recent work by Raffel et al. (2019) built a model called T5 to perform multiple tasks at once using natural language as its output. The model differentiates tasks by using prefixes in its input such as *"classify sentiment:"*, *"summarize:"*, or *"translate from English to German:"* and classify each input by generating natural words such as *"positive"* for sentiment classification or *"This article describes ..."* for summarization.

However, the appropriate output format for important *sequence labeling* applications in NLP, such as named entity recognition (NER) and slot labeling (SL) is not immediately clear. In this work, we propose an *augmented natural language* format for sequence labeling tasks. Our format locally tags words within the sentence (Figure 1) and is easily extensible to sentence-level classification tasks, such as intent classification (IC).

Our highlighted contributions and main findings are as follows:

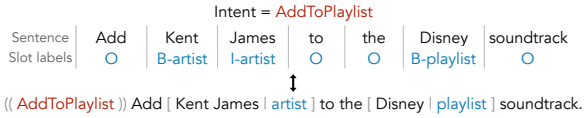


Figure 1: The conversion between the canonical BIO tagging format and our augmented natural language format.

1) We propose an effective new output format to perform joint sequence labeling and sentence classification through a generation framework.
2) We demonstrate the ability to perform multiple tasks such as named entity recognition, slot labeling and intent classification within a single model.
3) Our approach is highly effective in low-resource settings. Even without incorporating label type semantics as priors, the generative framework learns more efficiently than a token-level classification baseline. The model improves further given natural word labels, indicating the benefits of rich semantic information.
4) We show that supervised training on related sequence labeling tasks acts as an effective meta-learner that prepares the model to generate the appropriate output format. Learning each new task becomes much easier and results in significant performance gains.
5) We set a new state-of-the-art for few-shot slot labeling, outperforming the prior state-of-the-art by a large margin.
6) We plan to open source our implementation. Please visit https://arxiv.org/abs/2009.13272 for the release updates.

2 Model

Sequence Labeling as Generation

Most work on sequence labeling uses token-level classification frameworks. That is, given a list of tokens $\ell = \{\ell_i\}_{i=1}^n$, we perform a prediction on every token ℓ_i to obtain $y' = \{y'_i\}_{i=1}^n = \{f(\ell_i; \ell)\}_{i=1}^n$ where $f(\cdot)$ is a token-level prediction function. The prediction is accurate if it matches the original sequence label $y = \{y_i\}_{i=1}^n$.

In contrast to this convention, we frame sequence labeling as a conditional sequence generation problem where given the token list ℓ, we generate an output list $o = g(\ell)$ where g is a sequence-to-sequence model. A "naive" formulation for this task would be to directly generate $o = y$ given ℓ. However, this approach is prone to errors such as word misalignment and length mismatch (see supplementary materials Section A.2 for discussion).

We propose a new formulation for this generation task such that, given the input sequence ℓ, our method generates output o in **augmented natural language**. The augmented output o repeats the original input sequence ℓ with additional markers that indicate the token-spans and their associated labels. More specifically, we use the format $[\ \ell_j, \ldots, \ell_{j+t} \mid\ L\ ]$ to indicate that the token sequence $\ell_j, \ldots, \ell_{j+t}$ is labeled as L.

Fig. 1 depicts the proposed format and its equivalent canonical BIO format for the same input sentence. The conversion between the BIO format and our augmented natural language format is invertible without any information loss. This is crucial so that the generated output from model prediction can be converted back for comparison without uncertainty.

There are other formats which can encapsulate all the tagging information but are not invertible. For instance, outputting only the token spans of interest with tagging patterns $[\ \ell_j, \ldots, \ell_{j+t} \mid\ L\ ]$ without repeating the entire sentence results in the invertibility breaking down when there are duplicate token spans with different labels. We discuss this further in the appendix Section A.3.

Joint Sequence Classification and Labeling

Our sequence to sequence approach also supports joint sentence classification and sequence labeling by incorporating the sentence-level label in the augmented natural language format. In practice, we use the pattern ((sentence-level label)) in the beginning of the generated sentence, as shown in Fig. 1. The use of double parentheses is to prevent confusion with a single parenthesis that can occur in the original word sequence ℓ.

Training and Evaluation

We train our model by adapting the pre-trained T5 with the sequence to sequence framework. Additionally, we prefix the input with task descriptors in order to simultaneously perform multiple classification and labeling tasks, similar to the approach by Raffel et al. (2019). This results in a seamless multi-task framework, as illustrated in the top part of Fig. 2. To evaluate, we convert the generated output back to the canonical BIO format and calcu-

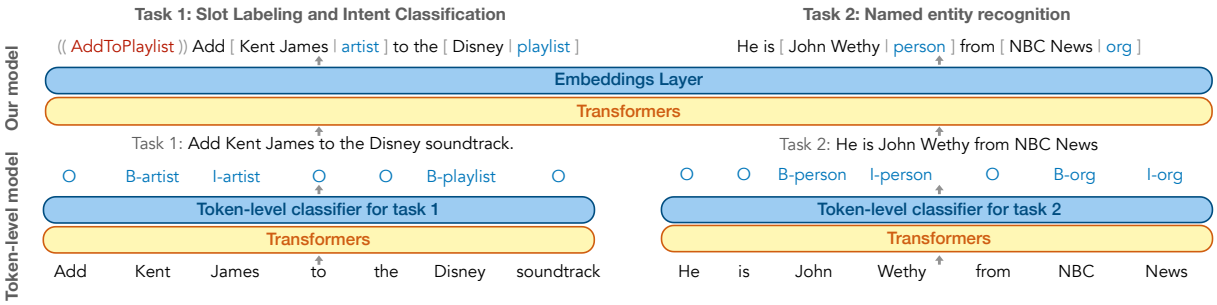


Figure 2: Comparison between our generative-style sequence labeling model (top) and the conventional token-level classification model (bottom).

late the F1 score for sequence labeling or accuracy for sentence classification.

Natural Labels

Labels are associated to real-world concepts that can be described through natural words. These words have rich information, but are often ignored in traditional discriminative approaches. In contrast, our model naturally incorporate label semantics directly through the generation-as-classification approach.

We perform label mapping in order to match the labels to its natural descriptions and use the natural labels in the augmented natural language output. Our motivation is as follows: (1) Pre-trained conditional generation models which we adapt on have richer semantics embedded in natural words, rather than dataset-specific label names. For instance, "country city state" contains more semantic information compared to "GPE", which is an original label in named entity recognition tasks. Using natural labels should allow the model to learn the association between word tokens and labels more efficiently, without requiring many examples. (2) Label knowledge can be shared among different tasks. For instance, after learning how to label names as "person", given a new task in another domain which requires labeling "artist", the model can more easily associate names with "artist" due to the proximity of "person" and "artist" in embeddings. This is not the case if the concept of "person" was learned with other uninformative words.

3 Related Work

Sequence to sequence learning has various applications including machine translation (Sutskever et al., 2014; Bahdanau et al., 2015), language modeling (Radford et al., 2018; Raffel et al., 2019), abstractive summarization (Rush et al., 2015), generative question answering (Dong et al., 2019), to name a few. However, the sequence-to-sequence framework is often not a method of choice when it comes to sequence labeling. Most models for sequence labeling use the token-level classification framework, where the model predicts a label for each element in the input sequence (Baevski et al., 2019; Li et al., 2019b; Chen et al., 2019). While select prior work adopts the sequence-to-sequence method for sequence labeling (Chen and Moschitti, 2018), this approach is not widely in use due to the difficulty of fixing the output length, output space, and alignment with the original sequence.

Multi-task and multi-domain learning often benefit sequence labeling performance (Changpinyo et al., 2018). The archetypal multi-task setup jointly trains on a target dataset and one or more auxiliary datasets. In the cross lingual setting, these auxiliary datasets typically represent high-resource languages (Schuster et al., 2018; Cotterell and Duh, 2017). While in a monolingual scenario, the auxiliary datasets commonly represent similar, high-resource tasks. Examples of similar multi-task pairs include NER and slot labeling (Louvan and Magnini, 2019) as well as dialogue state tracking and language understanding (Rastogi et al., 2018).

A recent series of works frame natural language processing tasks, such as translation, question answering, and sentence classification, as conditional sequence generation problems (Raffel et al., 2019; Radford et al., 2019; Brown et al., 2020). By unifying the model output space across tasks to consist of natural language symbols, these approaches reduce the gap between language model pre-training tasks and downstream tasks. Moreover, this framework allows acquisition of new tasks without any architectural change. The GPT-3 model (Brown

	Task & Dataset	Intent Clas.		Slot Labeling			
		SNIPS	ATIS	SNIPS	ATIS	CoNLL	Onto
SL/IC	Bi-Model (Wang et al., 2018)		**98.99**		**96.89**		
SL/IC	Joint BERT (Chen et al., 2019)	98.60	97.50	**97.00**	96.10		
SL/IC	ELMO+BiLSTM (Siddhant et al., 2019)	**99.29**	97.42	93.90	95.62		
NER	Cloze-CNN (Baevski et al., 2019)					**93.50**	
NER	BERT-MRC (Li et al., 2019a)					93.04	91.11
NER	BERT-MRC + DSC (Li et al., 2019b)					93.33	**92.07**
NER	BERT Base (Devlin et al., 2019)					92.40	88.95
	Ours: Individual	99.00	96.86	**97.43**	96.13	90.70	**90.24**
	Ours: SNIPS+ATIS	**99.29**	**97.20**	97.21	95.83		
	Ours: CoNLL+Ontonotes					**91.48**	89.52
	Ours: SNIPS+ATIS+CoNLL+Ontonotes	99.14	97.08	96.82	**96.65**	**91.48**	89.67

Table 1: Results of our models trained on combinations of datasets. Results for **Ours: individual** are from models trained on a single respective dataset. We underline scores of our models that exceed previous state-of-the-art results in each domain. Scores in boldface are the best overall scores among our models, or among the baselines. We use the boldface and underline notation for the rest of the paper.

et al., 2020) demonstrates the promise of this framework for few-shot learning. Among other successes, GPT-3 outperforms BERT-Large on the SuperGLUE benchmark using only 32 examples per task. To the best of our knowledge, we are the first to apply this multi-task conditional sequence generation framework to sequence labeling.

The conditional sequence generation framework makes it easy to incorporate label semantics, in the form of label names such as *departure city*, example values like *San Francisco*, and descriptions like "*the city from which the user would like to depart on the airline*". Label semantics provide contextual signals that can improve model performance in multi-task and low-resource scenarios. Multiple works show that conditioning input representations on slot description embeddings improves multi-domain slot labeling performance (Bapna et al., 2017; Lee and Jha, 2019). Embedding example slot values in addition to slot descriptions yields further improvements in zero-shot slot labeling (Shah et al., 2019). In contrast to our work, these approaches train slot description and slot value embedding matrices, whereas our framework can incorporate these signals as natural language input without changing the network architecture.

4 Experimental Setup and Results

4.1 Data

Datasets We use popular benchmark data SNIPS (Coucke et al., 2018) and ATIS (Hemphill et al., 1990) for slot labeling and intent classification. SNIPS is an SLU benchmark with 7 intents and 39 distinct types of slots, while ATIS is a benchmark for the air travel domain (see appendix A.4 for details). We also evaluate our approach on two named entity recognition datasets, Ontonotes (Pradhan et al., 2013) and CoNLL-2003 (Sang and Meulder, 2003).

Construction of Natural Labels We preprocess the original labels to natural words as follows. For Ontonotes and CoNLL datasets, we transform the original labels via mappings detailed in Table 9 and 5 in the appendix. For instance, we map "PER" to "person" and "GPE" to "country city state". For SNIPS and ATIS, we use the following rules to convert intent and slot labels: (1) we split words based on ".", "_", "/", and capitalized letters. For instance, we convert "object_type" to "object type" and "AddToPlaylist" to "add to playlist". These rules result in better tokenization and enrich the label semantics. We refer to these as the **natural label** setting and use is as our default.

4.2 Multi-Task Sequence Classification and Slot Labeling

We first demonstrate that our model can perform multiple tasks in our generative framework and achieve highly competitive or state-of-the-art performance. We consider 4 sequence labeling tasks and 2 classification tasks: NER on Ontonotes and CoNLL datasets; and slot labeling (SL) and in-

tent classification (IC) on SNIPS and ATIS dialog datasets. For comparison, we provide baseline results from the following models:

SL and IC: Bi-Model (Wang et al., 2018) uses two correlated bidirectional LSTMs to perform both IC and SL. **Joint BERT** (Chen et al., 2019) performs joint IC and SL with a sequential classifier on top of BERT, where the classification for the start-of-sentence token corresponds to intent class. **ELMO+Bi-LSTM** (Siddhant et al., 2019) uses a Bi-LSTM with CRF as a classifier on top of pre-trained ELMO (Peters et al., 2018).

NER: Cloze-CNN (Baevski et al., 2019) fine-tunes a Bi-LSTM with CRF model Peters et al. (2018) on a pre-trained model with a cloze-style word reconstruction task. **BERT MRC** (Li et al., 2019a) performs sequence labeling in a question answering model to predict the slot label span. **BERT MRC + Dice Loss** (Li et al., 2019b) improves upon BERT MRC with a dice loss shown to be suitable for data with imbalanced labels. **BERT** (Devlin et al., 2019) refers to a token-level classification with BERT pre-trained model. Note that the results for BERT with Ontonotes are from our own implementation.

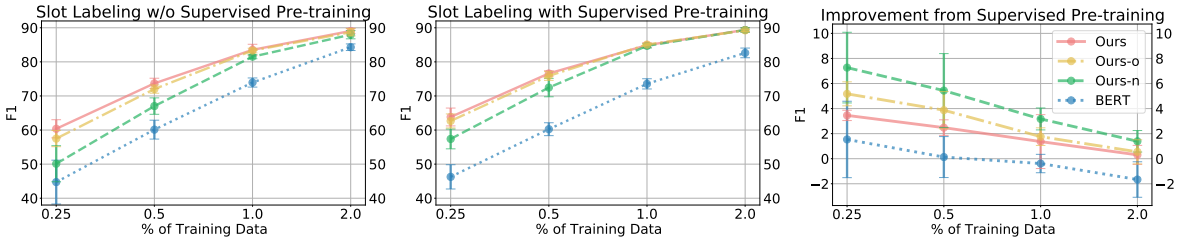
In Table 1, we report a summary of the results for our method and the baselines. Our proposed model achieves highly competitive results for ATIS, Ontonotes, and CoNLL datasets, as well as state-of-the-art slot labeling and intent classification performance on the SNIPS dataset. Unlike all the baseline models, which can perform a single task on a specific dataset, our model can perform all the tasks considered at once (last row of Table 1). For the multi-task models, our results show that different sequence labeling task can help mutually benefit each other, where ATIS slot labeling result improves from 96.13 to 96.65 and CoNLL improves from 90.70 to 91.48. While there are other approaches that perform better than our models in some tasks, we highlight the simplicity of our generation framework which performs multiple tasks seamlessly. This ability helps the models transfer knowledge among tasks with limited data, which are demonstrated through the rest of the paper.

4.3 Limited Resource Scenarios and Importance of Label Semantics

In this section, we show that our model can use the semantics of labels to learn efficiently, which is crucial for scenarios with limited labeled data. To demonstrate this effect, we use our model with the following variants of labels which differ semantic quality: (1) **natural label**, (2) **original label** and (3) **numeric label**.

The natural label version is our default setting where we use labels expressed in natural words. The original label case uses labels provided by the datasets, and the numeric label case uses numbers $0, 1, 2, ...$ as label types. In the numeric version, the model does not have pre-trained semantics of the label types and has to learn the associations between the labels and the relevant words from scratch. We also compare with the BERT **token-level classification** model. Similar to the numeric label case, the label types for BERT do not initially have associated semantics and are implicit through indices in the classifier weights. We use the SNIPS dataset to conduct our experiments due to its balanced domains (see Table 7 in Appendix). We experiment with very limited resource scenarios where we use as low as 0.25% of training data, corresponding to roughly one training sentence per label type on average.

Figure 3a shows the sequence labeling performance for varying amount of training data (see Table 10 in the appendix for numeric results). We observe that label semantics play a crucial role in the model's ability to learn effectively for limited resource scenarios. Our model with natural labels outperforms all other models, achieving an F1 score of $60.4 \pm 2.7\%$ with 0.25% training data, and giving a slight boost over using original labels ($57.5 \pm 2.4\%$). We believe that the improvement can be more dramatic in other datasets where the original labels have no meanings (such as in the numeric case), are heavily abbreviated, or contain rare words. With the numeric model, the performance suffers significantly in low-resource settings, achieving only $50.1 \pm 5.3\%$, or 10.3% lower than the natural label model, with 0.25% data. This result further supports the importance of label semantics in our generation approach. Interestingly, we also observe that the numeric model still outperforms BERT token-level classification ($44.7 \pm 6.4\%$), where neither model contains prior label semantics. This result indicates that even in the absence of label meanings, the generation approach seems more suitable than the token-level framework.



(a) (b) (c)

Figure 3: Model performance on limited amount of training data. The error bars indicate the standard deviation over 4 random trials. **Ours-o** is our model with its original labels. **Ours-n** is our model with numeric labels.

4.4 Teaching Model to Generate via Supervised Transfer Learning

While we train our model in limited data scenarios, we are asking the model to generate a new format of output given small amount of data. This is challenging since a sequence generation framework typically requires large amount of training (Sutskever et al., 2014). Despite this challenge, our model is able to outperform the classical token-level framework with ease. This section explores a clear untapped potential – by teaching our model how to generate the augmented natural language format before adapting to new tasks, we show that the performance on limited data significantly improves. This result contrasts with the BERT token-level model where supervised transfer learning hurts overall performance compared to BERT's initial pre-training due to possible overfitting.

To conduct this experiment, we train our model with the Ontonotes NER task in order to teach it the expected output format. Then, we adapt it on another task (SNIPS) with limited data, as in Section 4.3. We compare the results with the token-level BERT model, which also uses the BERT model trained on Ontonotes for supervised pre-training. We demonstrate the results in Figure 3b as well as highlight the improvement due to supervised pre-training in Figure 3c. We also provide full numeric results in the appendix Table 11 for reference.

Our model demonstrates consistent improvement, achieving an F1 score of $63.8 \pm 2.6\%$ using 0.25% of the training dataset, compared to $60.4 \pm 0.27\%$ without supervised transfer learning. The improvement trend also continues for other data settings, as shown in Figure 3c. The benefits from transfer learning is particularly strong for the numeric label model, achieving $57.4 \pm 2.9\%$ compared to $50.1 \pm 5.3\%$ for 0.25% data. This results suggests that the initial knowledge from supervised pre-training helps the model associate its labels (without prior semantics) to the associated words more easily.

The supervised transfer learning can also be seen as a *meta-learner*, which teaches the model how to perform sequence labeling in the generative style. In fact, when we investigate the model output *without* adapting to the SNIPS dataset, in addition to the output having the correct format, it already contains relevant tagging information for new tasks.

For instance, a phrase "Onto jerry's Classical Moments in Movies" from the SNIPS dataset results in the model output "Onto jerry's [Classical Moments in Movies | *work of art*]". This prediction closely matches the true label "Onto [jerry's | *playlist owner*] [Classical Moments in Movies | *playlist*]" where the true class of "Classical Moments in Movies" is *playlist*. Intuitively, the classification as *work of art* is in agreement with the true label *playlist*, but simply needs to be refined to match the allowed labels for the new task.

In contrast to our framework where the supervised transfer learning helps teach the model an output style, the transfer learning for the token-level classification simply adapts its weights and retains the same token-level structure (albeit with a new classifier). We observe no significant improvement from supervised pre-training for the BERT token-level model, which obtains an F1 score of $46.3 \pm 3.6\%$ compared to $44.7 \pm 6.4\%$ without supervised pre-training (with 0.25% SNIPS data). The improvements are also close to zero or negative for higher data settings (Figure 3c), suggesting that the pre-training of the token-level classification might overfit to the supervised data, and results in lower generalization on other downstream tasks. Overall, the final result on the BERT model lags far behind our framework, performing 17.5% lower than our model's score for 0.25% training data.

In addition, our model with numeric labels performs much better than the BERT token-level

model and further highlights the suitability of our generative output format for sequence labeling, regardless to the label semantics. Possible explanations are that the sequence to sequence label is less prone to overfitting compared to the classification framework. It could also be the case that locally tagging words with labels in the word sequence helps improve attention within the transformers model, and improve robustness to limited data.

4.5 Few-Shot Sequence Labeling

4.5.1 Few-Shot Learning

In few-shot learning, we seek to train models such that given a new task, the models are able to learn efficiently from few labels. Different tasks are sampled from various data domains which differ in terms of allowed labels and other nuances such as input styles.

We define a data domain $\mathcal{D}$ as a set of labeled examples $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N_\mathcal{D}}$ which has its set of allowed label types $\mathcal{Y}_\mathcal{D} \ni y_i$. Few-shot learning approaches are evaluated over many *episodes* of data, which represent a variety of novel tasks. Each episode $(\mathcal{S}, \mathcal{Q})$ consists of a support set $\mathcal{S}$ containing K-shot labeled samples, as well as a query set $\mathcal{Q}$ used for evaluation. Data from the evaluation episodes are drawn from the *target* domains $\{\mathcal{D}_1^T, \mathcal{D}_2^T, \ldots\}$, which the model has not previously seen.

To learn such models, we typically have access to another set of domains called the *source* domains $\{\mathcal{D}_1^S, \mathcal{D}_2^S, \ldots\}$, which can be used as the training resources. In order to train the model to learn multiple tasks well, many few-shot learning approaches use *meta-learning*, or a learning to learn approach, where the model is trained with many episodes drawn from the source domains in order to mimic the evaluation (Vinyals et al., 2016; Snell et al., 2017; Sung et al., 2018; Finn et al., 2017). We refer to this as the *episodic training*.

Another approach, called *fine-tuning*, trains the model on a regular training set from the source domains: $\cup_m \mathcal{D}_m^S$. Given an episode $(\mathcal{S}, \mathcal{Q})$ at evaluation time, the model fine-tunes it on the support $\mathcal{S}$, typically with a new classifier constructed for the new task, and evaluates on $\mathcal{Q}$.

4.5.2 Few-Shot Baselines

TransferBERT trains a token-level classification model by fine-tuning. **Matching Net (MN) + BERT** Vinyals et al. (2016) Given a word x_i, the model classifies by finding the most similar word $x_j^\mathcal{S}$ in the support set and predicts $y_j^\mathcal{S}$ as the label of x_i. The model also adapts the backbone model with episodic training. **Warm Proto Zero (WPZ) + BERT** Fritzler et al. (2019) uses token-level prototypical network (Snell et al., 2017), which classifies by comparing a word x_i to each class centroid rather than individual sample embeddings. **L-TapNet + CDT** Hou et al. (2020) uses a CRF framework and leverages label semantics in representing labels to calculate emission scores and a collapsed dependency transfer method to calculate transition scores. We note that all baselines except for TransferBERT uses episodic meta-training whereas TransferBERT uses fine-tuning. All baseline results are taken from Hou et al. (2020).

Our model performs fine-tuning with the generation framework. The major difference between our model and a token-level classification model such as TransferBERT is that we do not require a new classifier for every novel task during the fine-tuning on the support set. The sequence generation approach allows us to use the entire model and adapt it to new tasks, where the initial embeddings contain high quality semantics and help the model transfer knowledge efficiently.

4.5.3 K-shot Episode Construction

Traditionally, the support set $\mathcal{S}$ is often constructed in K-shot formats where we use only K instances of each label type. In sequence labeling problems, this definition is challenging due to the presence of multiple occurrences or multiple label types in a single sentence. We follow Hou et al. (2020) by using the following definition of a K-shot setting: All labels within the task appears at least K times in $\mathcal{S}$ and would appear less than K times if any sentence is removed. We sample 100 episodes from each domain according to this definition. Note that Hou et al. (2020)'s episodes are similar to ours, but preprocess the sentences by lowercasing and removing extra tokens such as commas (see details in Section A.6). Our model is flexible and can handle raw sentences; we therefore use the episodes from the original SNIPS dataset without any modifications.

4.5.4 Data

We perform few-shot experiments on the 7 domains $\{\mathcal{D}_1, \ldots, \mathcal{D}_7\}$ of the SNIPS dataset, namely, Weather (We), Music (Mu), Playlist (Pl), Book (Bo), ScreeningEvent (Se), Restaurant (Re), CreativeWork (Cr). To evaluate a model on domain $\mathcal{D}_i$, we meta-train the model on $\mathcal{D}_i' = \{\mathcal{D}_1, \ldots, \mathcal{D}_7\} -$

		We	Mu	Pl	Bo	Se	Re	Cr	Ave.
1-shot	TransferBERT	55.82	38.01	45.65	31.63	21.96	41.79	38.53	39.06
	MN + BERT	21.74	10.68	39.71	58.15	24.21	32.88	**69.66**	36.72
	WPZ + BERT	46.72	40.07	50.78	68.73	60.81	55.58	67.67	55.77
	L-TapNet+CDT	**71.53**	**60.56**	**66.27**	**84.54**	**76.27**	**70.79**	62.89	**70.41**
	Ours + SNIPS	**82.62**	**77.46**	**71.33**	**85.49**	**83.22**	**84.23**	**82.92**	**81.04**
	Ours + Onto	56.39	67.10	53.49	71.94	66.21	69.04	28.80	59.00
	Ours + No Meta	46.42	59.02	47.47	63.79	49.42	64.45	17.60	49.74
5-shot	TransferBERT	59.41	42.00	46.70	20.74	28.20	67.75	58.61	46.11
	MN + BERT	36.67	33.67	52.60	60.09	38.42	33.28	72.10	47.98
	WPZ + BERT	67.82	55.99	46.02	72.17	73.59	60.18	66.89	63.24
	L-TapNet+CDT	**71.64**	**67.16**	**75.88**	**84.38**	**82.58**	**70.05**	**73.41**	**75.01**
	Ours + SNIPS	**91.35**	**86.73**	**87.20**	**95.85**	**92.71**	**91.23**	**91.55**	**90.95**
	Ours + Onto	83.15	86.15	80.36	90.27	84.87	85.89	68.08	82.68
	Ours + No Meta	73.14	82.02	78.82	84.86	83.14	86.63	52.56	77.31

Table 2: Our few-shot slot labeling results on 7 domains of SNIPS dataset. Ours + SNIPS perform meta-training on the leave-one-out SNIPS data, similar to other baselines. Ours + Onto is our model trained on Ontonotes. Ours + No Meta involves no meta-training.

$\mathcal{D}_i$. We refer to this as the *leave-one-out* meta-training sets. All other baselines also use this meta-training data setup.

We note that the training set $\mathcal{D}'_i$ has data distributions that closely match $\mathcal{D}_i$ since they are both drawn from the SNIPS dataset. We investigate more challenging scenarios where we use an alternative source as a meta-training set, as well as no meta-training. In particular, we choose Ontonotes NER task as the alternative source domain. The benefits of using this setup is such that it establishes a single meta-trained model that works across *all* evaluation domains, which we offer as a challenging benchmark for future research.

4.5.5 Few-Shot Results

Table 2 demonstrates the results for few-shot experiments. Our model outperforms previous state-of-the-art on every domain evaluated. In the 5-shot case, our model achieves an average F1 score of 90.9%, exceeding the strongest baseline by 15.9%. Even without meta-training, the model is able to perform on par with state-of-the-art models, achieving an F1 score of 77.3% versus 75.0% for the baseline. Training on an alternative source (NER task) also proves to be an effective meta-learning strategy, performing better than the best baseline by 7.7%. These results indicate that our model is robust in its ability to learn sequence tagging on target domains that differ from sources. In the 1-shot case, our model achieves an average F1 score of 81.0%, outperforming the best baseline significantly (10.6% improvement).

We note that the average support sizes are around 5 to 40 sentences for the 5-shot case, and one to 8 sentences for the 1-shot case (see Table 12 and 13 for details). The results are particularly impressive given that we adapt a large transformer model based on such limited number of samples. In comparison to other fine-tuning approaches such as TransferBERT, our model performs substantially better, indicating that our generative framework is a more data-efficient approach for sequence labeling.

5 Discussion and Future Work

Our experiments consistently show that the generation framework is suitable for sequence labeling and sets a new record for few-shot learning. Our model adapts to new tasks efficiently with limited samples, while incorporating the label semantics expressed in natural words. This is akin to how humans learn. For instance, we do not learn the concept of "person" from scratch in a new task, but have prior knowledge that "person" likely corresponds to names, and refine this concept through observations. The natural language output space allows us to retain the knowledge from previous tasks through shared embeddings, unlike the token-level model which needs new classifiers for novel

tasks, resulting in a broken chain of knowledge.

Our approach naturally lends itself to life-long learning. The unified input-output format allows the model to incorporate new data from any domain. Moreover, it has the characteristics of a single, life-long learning model that works well on many levels of data, unlike other approaches that only perform well on few-shot or high-resource tasks. Our simple yet effective approach is also easily extensible to other applications such as multi-label classification, or structured prediction via nested tagging patterns.

References

Alexei Baevski, Sergey Edunov, Yinhan Liu, Luke Zettlemoyer, and Michael Auli. 2019. Cloze-driven pretraining of self-attention networks. 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 2019, Hong Kong, China, November 3-7, 2019*, pages 5359–5368. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Ankur Bapna, Gokhan Tur, Dilek Hakkani-Tur, and Larry Heck. 2017. Towards zero-shot frame semantic parsing for domain scaling. *arXiv preprint arXiv:1707.02363*.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners.

Soravit Changpinyo, Hexiang Hu, and Fei Sha. 2018. Multi-task learning for sequence tagging: An empirical study. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2965–2977, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Lingzhen Chen and Alessandro Moschitti. 2018. Learning to progressively recognize new named entities with sequence to sequence models. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2181–2191, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Qian Chen, Zhu Zhuo, and Wen Wang. 2019. BERT for joint intent classification and slot filling. *CoRR*, abs/1902.10909.

Ryan Cotterell and Kevin Duh. 2017. Low-resource named entity recognition with cross-lingual, character-level neural conditional random fields. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 91–96, Taipei, Taiwan. Asian Federation of Natural Language Processing.

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 13042–13054.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135. PMLR.

Alexander Fritzler, Varvara Logacheva, and Maksim Kretov. 2019. Few-shot classification in named entity recognition task. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing, SAC 2019, Limassol, Cyprus, April 8-12, 2019*, pages 993–1000. ACM.

Charles T. Hemphill, John J. Godfrey, and George R. Doddington. 1990. The ATIS spoken language systems pilot corpus. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, USA, June 24-27, 1990*. Morgan Kaufmann.

Yutai Hou, Wanxiang Che, Yongkui Lai, Zhihan Zhou, Yijia Liu, Han Liu, and Ting Liu. 2020. Few-shot slot tagging with collapsed dependency transfer and label-enhanced task-adaptive projection network. In *ACL*.

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

Cheolhyoung Lee, Kyunghyun Cho, and Wanmo Kang. 2020. Mixout: Effective regularization to finetune large-scale pretrained language models. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Sungjin Lee and Rahul Jha. 2019. Zero-shot adaptive transfer for conversational language understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6642–6649.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2019. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *CoRR*, abs/1910.13461.

Xiaoya Li, Jingrong Feng, Yuxian Meng, Qinghong Han, Fei Wu, and Jiwei Li. 2019a. A unified MRC framework for named entity recognition. *CoRR*, abs/1910.11476.

Xiaoya Li, Xiaofei Sun, Yuxian Meng, Junjun Liang, Fei Wu, and Jiwei Li. 2019b. Dice loss for data-imbalanced NLP tasks. *CoRR*, abs/1911.02855.

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

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Samuel Louvan and Bernardo Magnini. 2019. Leveraging non-conversational tasks for low resource slot filling: Does it help? In *Proceedings of the 20th Annual SIGdial Meeting on Discourse and Dialogue*, pages 85–91, Stockholm, Sweden. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 2227–2237. Association for Computational Linguistics.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. 2013. Towards robust linguistic analysis using ontonotes. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning, CoNLL 2013, Sofia, Bulgaria, August 8-9, 2013*, pages 143–152. ACL.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf*.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *CoRR*, abs/1910.10683.

Abhinav Rastogi, Raghav Gupta, and Dilek Hakkani-Tur. 2018. Multi-task learning for joint language understanding and dialogue state tracking. In *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*, pages 376–384, Melbourne, Australia. Association for Computational Linguistics.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389, Lisbon, Portugal. Association for Computational Linguistics.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning, CoNLL 2003, Held in cooperation with HLT-NAACL 2003, Edmonton, Canada, May 31 - June 1, 2003*, pages 142–147. ACL.

Sebastian Schuster, Sonal Gupta, Rushin Shah, and Mike Lewis. 2018. Cross-lingual transfer learning for multilingual task oriented dialog. *arXiv preprint arXiv:1810.13327*.

Darsh Shah, Raghav Gupta, Amir Fayazi, and Dilek Hakkani-Tur. 2019. Robust zero-shot cross-domain slot filling with example values. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5484–5490, Florence, Italy. Association for Computational Linguistics.

Aditya Siddhant, Anuj Kumar Goyal, and Angeliki Metallinou. 2019. Unsupervised transfer learning for spoken language understanding in intelligent agents. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 4959–4966. AAAI Press.

Jake Snell, Kevin Swersky, and Richard S. Zemel. 2017. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 4077–4087.

Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip H. S. Torr, and Timothy M. Hospedales. 2018. Learning to compare: Relation network for few-shot learning. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 1199–1208. IEEE Computer Society.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112.

Oriol Vinyals, Charles Blundell, Tim Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. 2016. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 3630–3638.

Yu Wang, Yilin Shen, and Hongxia Jin. 2018. A bi-model based RNN semantic frame parsing model for intent detection and slot filling. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 2 (Short Papers)*, pages 309–314. Association for Computational Linguistics.

Ralph Weischedel, Sameer Pradhan, Lance Ramshaw, Jeff Kaufman, Michelle Franchini, Mohammed El-Bachouti, Nianwen Xue, Martha Palmer, Jena D. Hwang, Claire Bonial, Jinho Choi, Aous Mansouri, Maha Foster, Abdel aati Hawwary, Mitchell Marcus, Ann Taylor, Craig Greenberg Eduard Hovy, Robert Belvin, and Ann Houston. 2012. Ontonotes release 5.0.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 5754–5764.

A Supplementary Materials

A.1 Experiment Setup

We describe the experiment setup for reproducibility in this section. We use Huggingface's T5-BASE conditional generation model as well as their trainer (`transformers.Trainer`) with its default hyperparameters to train all our models. The trainer uses AdamW (Kingma and Ba, 2015; Loshchilov and Hutter, 2019) and linear learning rate decay.

- We use 8 V100 GPUs for all our experiments.
- Maximum batch size per GPU = 8.
- Maximum sequence length = 128 for all tasks except Ontonotes where we use 175.
- The number of epochs in multi-task experiments is 50.
- The number of epochs for limited-resource experiments is scaled to have the same number of optimization steps as that of when we use the entire training set. For instance, if we use 10% of the entire training set, we use 500 epochs on that limited set.
- We perform no preprocessing except for replacing the labels with the natural labels described in Section 4.1.
- We use a library `seqleval`[1] for F1 evaluation which supports many tagging format such as BIO, BIOES, etc.
- We use `bert-base-multilingual-cased` for our BERT model.

Task Descriptor Prefixes A task descriptor helps encode information about the allowed set of label types. For instance, CoNLL allows only 4 types of labels whereas the 18 label types in Ontonotes are more fine-grained. A task descriptor also help encode the nuances among labels; for example, the CoNLL dataset has only one tag type "LOC" for location whereas Ontonotes differentiate locations with "GPE" (countries, cities, states) and a general "LOC" (non-GPE locations, mountain ranges, bodies of water). By specifying a task descriptor, we allow the model to learn the implicit constraints in the data and help it be able to distinguish what task it should perform given an input sentence. We use the corresponding prefixes "SNIPS: ", "ATIS: ", "Ontonotes: " or "CONLL: ". We ensure that these prefixes can be tokenized given a pretrained tokenizer properly without the model having to use the unknown token. We demonstrate in 4.2 that the prefix tags allow us to perform slot labeling (and intent classification) for different datasets using a single model. For a model trained on only a single dataset, such prefix can be omitted and does not affect the performance.

[1] https://github.com/chakki-works/seqeval.git

	sentence	label	prediction
0	It	O	O
1	abuts	O	O
2	Sanchih	B-GPE	O
3	Rural	I-GPE	B-GPE
4	Township	I-GPE	I-GPE
5	to	O	I-GPE
6	the	O	O
7	northeast	O	O
8	,	O	O
9	the	O	O
10	Kuantu	B-LOC	O
11	area	O	B-GPE
12	of	O	O
13	Taipei	B-GPE	O
14	City	I-GPE	B-GPE
15	-	-	I-GPE

Table 3: Example of a sentence and its tagging label from Ontonotes. The prediction is generated from training a sequence-to-sequence model with a raw BIO format.

A.2 A naive approach for sequence labeling as sequence to sequence

We consider training a sequence-to-sequence model where the output is the sequence of BIO tags. Table 3 demonstrates an example with model prediction. We find that the model often outputs predictions that are misaligned with the original sentence slots. This is due to the complex relationships with the tokenizer. For instance, the tokenized version of this sentence (using T5-BASE tokenizer) is of length 25 whereas the original sentence length is 14. Learning to map the slot labels to the correct tokens can be challenging.

```
_It, _, a, but, s, _San, chi, h,
_Rural, _Township, _to, _the,
_northeast, _, ,, _the, _Ku,
ant, u, _area, _of, _Tai, pe,
i, _city
```

Statistics\Dataset	SNIPS	ATIS	Ontonotes	CONLL
No. Training Samples	13084	4478	59924	14041
No. Validation Samples	700	500	8528	3250
No. Test Samples	700	893	8262	3453
Average sentence length	9.05	11.28	18.11	14.53
# of slot types (w/o BIO prefixes)	39	83	18	4
# intent types	7	21	N/A	N/A

Table 4: Dataset Statistics

CONLL-2003 slot types	Natural-word label
LOC	location
MISC	miscellaneous
ORG	organization
PER	person

Table 5: Label mapping for CoNLL-2003.

s	These	two	men	have	two	dollars
y	O	O	O	O	B-money	O

Table 6: Top row: original sentence. Bottom row: slot labels.

A.3 Shortened Generative Format

We show a failure case for a shorted generative format discussed in 2 where we repeat only the tagged pattern . Consider the following input s and label y,

If we repeat only the tagged pattern, then the output s_g will be

[**two** | money] .

Given s_g and s, it is ambiguous whether the canonical label should associate with `two` for `two dollars` or `two men`.

A.4 Dataset Details

We provide details on the statistics of datasets used in Table 4. We provide details on the intent and slot label types in Table 7 for the SNIPS dataset and Table 8 for the ATIS dataset. The tagging label types and their label mapping are listed in Table 9 for Ontonotes and 5 for CoNLL.

A.5 Low-Resource Results

We provide full numeric results for low-resource experiments from Section 4.3 and 4.4 in Table 10 and Table 11 respectively.

A.6 Few-Shot Experiments

We provide details on the data used for our few-shot experiments and the full results in this Section.

A.6.1 Episode Data

We use two data constructions: the original episodes from Hou et al. (2020) and our own constructed episodes with Hou et al. (2020)'s definition. We provide the episode statistics (Ave $|\mathcal{S}|$) in Table 12 for both constructions which demonstrate that the support size are comparable for each domain. The major difference is that Hou et al. (2020) pre-processes data by lowercasing all letters and removing extra tokens such as commas and apostrophes. In addition, Hou et al. (2020) modify the BIO prefixes in cases where the tokenization splits a token with the "B-" prefix into two or more units. For instance, the token ["lora's"] with tag [*B-playlist_owner*] becomes ["lora", "s"] with tags [*B-playlist_owner*, *I-playlist_owner*]. This treatment considerably increases the number of tokens with "I-" tags in the episodes created by Hou et al. (2020). Both data have 100 episodes with 20 query sentences. We provide the results for our episodes with lowercased words and Hou's episodes, which shows the similarity between two settings.

We note that the support size for some domains can be smaller than in other domains, according to Hou et al. (2020) K-shot definition. For instance, domain CR has around 5 sentences on average whereas domain RE has more than 30 sentences. This is because for some domains, there can be many tags of the same types in a single sentence.

A.6.2 5-Shot Results

Table 12 details the full results on both episode data and multiple variations of our models. **Ours + SNIPS** is trained on leave-one-out dataset. For instance, when we evaluate on domain **We**, we train on other domains except for **We**. This is the setting used in all other baselines. **Ours + Onto** shows the

Intent/Domain	# sen.	# sl.	Slot types (without BIO tags)
GetWeather (We)	2100	9	timeRange, condition_description, country, geographic_poi, city, state, current_location, condition_temperature, spatial_relation
PlayMusic (Mu)	2100	9	genre, year, album, music_item, playlist, service, sort, artist, track
AddToPlaylist (Pl)	2042	5	entity_name, music_item, playlist, playlist_owner, artist
RateBook (Bo)	2056	7	object_type, object_part_of_series_type, object_select, rating_value, object_name, best_rating, rating_unit
SearchScreeningEvent (Se)	2059	7	timeRange, object_location_type, object_type, location_name, movie_name, spatial_relation, movie_type
BookRestaurant (Re)	2073	14	party_size_number, served_dish, timeRange, country, poi, cuisine, spatial_relation, city, restaurant_name, sort, restaurant_type, facility, party_size_description, state
SearchCreativeWork (Cr)	2054	2	object_type, object_name
Sum	14484	53	
# Distinct slots		39	

Table 7: The intent classes and the corresponding slot labels for the SNIPS dataset. # sen is the number of sentences in the entire dataset. # sl. is the number of slot types for each intent, excluding 'O' (no tag). Note that sentences among different intent classes can share slot types and the number of slot types in total is 39 disregarding the BIO prefixes B- and I-.

results trained on Ontonotes NER task (see label types in Table 7), which is from a different domain that the SNIPS dataset. **Ours w/o meta** involves no additional meta-training and fine-tunes on our backbone model directly.

A.7 1-Shot Results

Our models tend to perform well when there are sufficient enough sentences to fine-tune on. For domain 'Cr' (SearchCreativeWork) where there are highly limited number of sentences (5 sentences on average), our model does not perform well compared to other baselines. This observation is consistent with the 1-shot results, which we include in the appendix Section A.7 Table 13, where we typically have less than 10 sentences in the support set. In this case, our model performs comparable to Warm Proto Zero with BERT on average but is outperformed by L-TapNet+CDT. Other techniques to improve on the 1-shot learning result include bootstrapping more sentences from unlabeled corpus with the labels from the support set for better optimization.

Intent	# sen.	# sl.	Slot types (without BIO tags)
atis_flight	4298	71	fromloc.city_name, toloc.city_name, round_trip, arrive_date.month_name, arrive_date.day_number, stoploc.city_name, arrive_time.time_relative, arrive_time.time, meal_description, depart_date.month_name, depart_date.day_number, airline_name, depart_time.period_of_day, depart_date.day_name, toloc.state_name, depart_time.time_relative, depart_time.time, depart_date.date_relative, or, class_type, fromloc.airport_name, flight_mod, meal, economy, city_name, airline_code, depart_date.today_relative, flight_stop, toloc.state_code, fromloc.state_name, toloc.airport_name, connect, arrive_date.day_name, fromloc.state_code, arrive_date.today_relative, depart_date.year, depart_time.start_time, depart_time.end_time, arrive_time.start_time, arrive_time.end_time, cost_relative, flight_days, mod, airport_name, aircraft_code, toloc.country_name, toloc.airport_code, return_date.date_relative, flight_number, fromloc.airport_code, arrive_time.period_of_day, depart_time.period_mod, flight_time, return_date.day_name, fare_amount, arrive_date.date_relative, arrive_time.period_mod, period_of_day, stoploc.state_code, fare_basis_code, stoploc.airport_name, return_time.period_mod, return_time.period_of_day, return_date.today_relative, return_date.month_name, return_date.day_number, compartment, day_name, airport_code, stoploc.airport_code, flight
atis_airfare	471	45	round_trip, fromloc.city_name, toloc.city_name, cost_relative, fare_amount, class_type, economy, airline_name, flight_mod, depart_time.time_relative, depart_time.time, arrive_date.month_name, arrive_date.day_number, airline_code, flight_number, stoploc.city_name, toloc.airport_name, depart_date.date_relative, depart_date.day_name, depart_date.month_name, depart_date.day_number, toloc.state_code, depart_time.period_of_day, flight_stop, fromloc.state_name, toloc.state_name, toloc.airport_code, aircraft_code, depart_date.year, arrive_time.time_relative, arrive_time.time, fromloc.airport_code, fromloc.airport_name, depart_date.today_relative, return_date.month_name, return_date.day_number, connect, meal, arrive_date.date_relative, arrive_date.day_name, or, depart_time.period_mod, flight_time, flight_days, fromloc.state_code
atis_ground_service	291	23	toloc.airport_name, city_name, fromloc.airport_name, toloc.city_name, state_code, transport_type, airport_name, fromloc.city_name, or, depart_date.date_relative, depart_date.day_name, time, depart_date.month_name, depart_date.day_number, today_relative, flight_time, state_name, period_of_day, time_relative, day_name, month_name, day_number, airport_code
atis_airline	195	36	fromloc.city_name, toloc.city_name, toloc.state_code, mod, depart_date.day_name, class_type, depart_date.today_relative, stoploc.city_name, aircraft_code, arrive_date.month_name, arrive_date.day_number, toloc.airport_name, fromloc.state_code, depart_time.period_of_day, airline_code, flight_number, depart_time.time_relative, depart_time.time, depart_date.month_name, depart_date.day_number, arrive_time.time, city_name, airport_name, flight_stop, arrive_time.period_of_day, fromloc.airport_code, airline_name, depart_date.date_relative, connect, flight_days, round_trip, cost_relative, fromloc.airport_name, depart_time.start_time, depart_time.end_time, toloc.state_name
atis_abbreviation	180	14	fare_basis_code, airport_code, airline_code, meal, meal_code, restriction_code, airline_name, aircraft_code, class_type, days_code, mod, fromloc.city_name, toloc.city_name, booking_class
atis_aircraft	90	23	fromloc.city_name, toloc.city_name, depart_time.time_relative, depart_time.time, toloc.state_code, airline_name, mod, class_type, depart_date.day_name, airline_code, flight_number, stoploc.city_name, depart_time.period_of_day, flight_mod, aircraft_code, arrive_time.time_relative, arrive_time.time, arrive_date.day_name, depart_date.month_name, depart_date.day_number, arrive_date.month_name, arrive_date.day_number, city_name
atis_flight_time	55	20	flight_time, airline_name, toloc.airport_code, depart_date.month_name, depart_date.day_number, fromloc.city_name, toloc.city_name, depart_date.day_name, depart_time.period_of_day, airline_code, flight_number, flight_mod, depart_date.date_relative, depart_time.time, fromloc.airport_name, aircraft_code, depart_time.time_relative, airport_name, class_type, meal_description
atis_quantity	54	25	airline_code, class_type, flight_stop, fromloc.city_name, toloc.city_name, arrive_date.month_name, arrive_date.day_number, depart_date.month_name, depart_date.day_number, economy, airline_name, round_trip, toloc.airport_name, depart_date.today_relative, arrive_time.time_relative, arrive_time.time, fare_basis_code, city_name, stoploc.city_name, flight_number, flight_days, depart_time.time_relative, depart_time.time, depart_time.period_of_day, aircraft_code
atis_airport	38	9	city_name, state_code, fromloc.city_name, mod, airport_name, toloc.city_name, flight_stop, state_name, airline_name
atis_capacity	37	5	fromloc.city_name, toloc.city_name, airline_name, aircraft_code, mod
atis_flight, atis_airfare	33	21	fromloc.city_name, toloc.city_name, airline_name, flight_number, depart_date.day_name, depart_date.month_name, depart_date.day_number, flight_mod, round_trip, depart_time.time_relative, depart_time.time, cost_relative, fare_amount, depart_date.date_relative, arrive_time.time_relative, arrive_time.time, depart_time.period_of_day, toloc.state_code, flight_stop, return_date.date_relative, return_date.day_name
atis_distance	30	8	fromloc.airport_name, toloc.city_name, fromloc.city_name, depart_time.time, depart_date.month_name, depart_date.day_number, city_name, airport_name
atis_city	25	11	city_name, airline_name, airport_code, fromloc.airport_code, fromloc.city_name, toloc.city_name, depart_time.time_relative, depart_time.time, depart_time.period_of_day, airport_name, class_type
atis_ground_fare	25	6	transport_type, city_name, fromloc.city_name, fromloc.airport_name, airport_name, toloc.city_name
atis_flight_no	20	22	toloc.city_name, fromloc.city_name, arrive_time.time_relative, arrive_time.time, fromloc.state_name, toloc.state_name, depart_date.day_name, depart_date.month_name, depart_date.day_number, airline_name, depart_time.time, flight_mod, toloc.state_code, flight_time, cost_relative, class_type, depart_time.time_relative, depart_time.period_of_day, stoploc.city_name, flight_number, or, depart_date.today_relative
atis_meal	12	12	meal, fromloc.city_name, toloc.airport_code, airline_name, flight_number, toloc.city_name, airline_code, arrive_time.time, toloc.state_code, depart_date.day_name, depart_time.period_of_day, meal_description
atis_restriction	6	6	restriction_code, cost_relative, round_trip, fromloc.city_name, toloc.city_name, fare_amount
atis_airline, atis_flight_no	2	7	fromloc.city_name, toloc.city_name, depart_date.date_relative, depart_date.month_name, depart_date.day_number, arrive_time.time_relative, arrive_time.time
atis_day_name	2	2	fromloc.city_name, toloc.city_name
atis_aircraft, atis_flight,atis_flight_no	1	5	fromloc.city_name, toloc.city_name, airline_name, depart_time.time_relative, depart_time.time
atis_cheapest	1	1	cost_relative
atis_ground_service, atis_ground_fare	1	1	fromloc.airport_name
atis_airfare, atis_flight_time	1	3	flight_time, fromloc.city_name, toloc.city_name
atis_airfare, atis_flight	1	4	airline_name, flight_number, fromloc.airport_code, toloc.airport_code
atis_flight, atis_airline	1	5	fromloc.city_name, toloc.city_name, depart_date.day_name, depart_time.time_relative, depart_time.time
atis_flight_no, atis_airline	1	5	fromloc.city_name, toloc.city_name, depart_date.day_name, depart_time.time_relative, depart_time.time
Sum	5871	390	
# Distinct slots		83	

Table 8: The intent classes and the corresponding slot labels for the ATIS dataset. # sen is the number of sentences in the entire dataset. # sl. is the number of slot types for each intent, excluding 'O' (no tag). Note that sentences among different intent classes can share slot types and the number of slot types in total is 83 disregarding the BIO prefixes B- and I-.

Ontonotes slot types	Natural-word label	Descriptions
CARDINAL	cardinal	Numerals that do not fall under another type
DATE	date	Absolute or relative dates or periods
EVENT	event	Named hurricanes, battles, wars, sports events, etc.
FAC	facility	Buildings, airports, highways, bridges, etc.
GPE	country city state	countries, cities, states
LANGUAGE	language	Any named language
LAW	law	Named documents made into laws
LOC	location	Non-GPE locations, mountain ranges, bodies of water
MONEY	money	Monetary values, including unit
NORP	nationality religious political group	Nationalities or religious or political groups
ORDINAL	ordinal	”first”, ”second”
ORG	organization	Companies, agencies, institutions, etc.
PERCENT	percent	Percentage (including “%”)
PERSON	person	People, including fictional
PRODUCT	product	Vehicles, weapons, foods, etc. (Nor services)
QUANTITY	quantity	Measurements, as of weight or distance
TIME	time	Times smaller than a day
WORK_OF_ART	work of art	Titiles of books, songs, etc.

Table 9: Label mapping for Ontonotes. The descriptions are obtained from Weischedel et al. (2012)

			Slot Labeling F1 Score							
%	**no. sen**	**s/t**	**Ours**	±	**Ours-o**	±	**Ours-n**	±	**BERT**	±
0.25	33	0.84	60.37	2.66	57.49	2.37	50.12	5.29	44.73	6.43
0.5	65	1.68	73.69	1.51	72.02	1.22	67.03	2.41	60.12	2.77
1	131	3.35	83.53	1.63	83.32	0.68	81.52	0.60	73.94	1.34
2	262	6.71	89.06	0.79	88.69	1.07	87.97	1.21	84.30	0.97
5	654	16.77	93.35	0.21	92.98	0.45	92.47	0.30	90.99	0.69
10	1308	33.55	94.65	0.18	94.51	0.29	94.41	0.22	93.56	0.68
20	2617	67.10	95.55	0.20	95.14	0.24	95.55	0.36	94.77	0.15
40	5234	134.19	96.15	0.15	96.22	0.10	96.16	0.11	96.05	0.29
100	13084	335.49	96.71	0.05	96.96	0.10	97.01	0.10	96.73	0.16

Table 10: Test results of our models for under varying levels of training resources on the SNIPS training data. **%** and **no. sen** columns indicate the % of the original training data and number of training sentences. **s/t** indicates the number of sentences per slot label type.

			Slot Labeling F1 Score							
%	**no. sen**	**s/t**	**Ours**	±	**Ours-o**	±	**Ours-n**	±	**BERT**	±
0.25	33	0.8	63.83	2.62	62.67	2.06	57.39	2.90	46.27	3.56
0.5	65	1.7	76.60	0.79	75.89	1.32	72.46	2.70	60.25	1.89
1	131	3.4	84.90	0.71	85.08	0.20	84.69	0.62	73.55	1.50
2	262	6.7	89.37	0.65	89.24	0.32	89.36	0.65	82.63	1.42
5	654	16.8	93.74	0.23	93.21	0.17	93.63	0.32	89.83	0.43
10	1308	33.5	94.78	0.24	94.60	0.41	94.78	0.23	92.23	0.35
20	2617	67.1	95.70	0.32	95.66	0.18	95.87	0.18	94.39	0.34
40	5234	134.2	96.52	0.25	96.60	0.23	96.40	0.21	95.67	0.32
100	13084	335.5	97.27	0.05	97.16	0.18	97.29	0.20	96.44	0.23

Table 11: Test results of our models for under varying levels of training resources on the SNIPS training data, with supervised pre-training on Ontonotes NER dataset. **%** and **no. sen** columns indicate the % of the original training data and number of training sentences. **s/t** indicates the number of sentences per slot label type.

		We	**Mu**	**Pl**	**Bo**	**Se**	**Re**	**Cr**	**Ave.**
Hou et al. (2020)'s Episodes	Ave. $\|\mathcal{S}\|$	28.91	34.43	13.84	19.83	19.27	41.58	5.28	
	SimBERT	53.46	54.13	42.81	75.54	57.10	55.30	32.38	52.96
	TransferBERT	59.41	42.00	46.70	20.74	28.20	67.75	58.61	46.11
	Matching Net	36.67	33.67	52.60	60.09	38.42	33.28	72.10	47.98
	WPZ + BERT	67.82	55.99	46.02	72.17	73.59	60.18	66.89	63.24
	L-TapNet+CDT	**71.64**	**67.16**	**75.88**	**84.38**	**82.58**	**70.05**	**73.41**	**75.01**
	Ours + SNIPS	87.66	81.62	83.37	89.72	86.80	86.14	73.02	84.05
	±	7.35	9.75	8.75	5.23	8.33	6.30	11.79	
	Ours + Onto	77.87	79.71	79.87	84.18	75.84	78.71	40.23	73.77
	±	7.98	8.54	9.08	5.64	8.35	7.54	14.59	
	Ours + No Meta	70.82	74.24	73.88	83.32	74.94	75.18	41.22	70.51
	±	8.39	9.26	8.11	6.11	9.93	7.24	16.42	
Our episodes	Ave. $\|\mathcal{S}\|$	24.92	32.49	12.56	18.44	17.18	37.06	5.70	
	Ours + SNIPS	**91.35**	**86.73**	**87.20**	**95.85**	**92.71**	**91.23**	**91.55**	**90.95**
	±	6.13	6.52	6.59	3.13	5.45	5.86	7.25	
	Ours + Onto	83.15	86.15	80.36	90.27	84.87	85.89	68.08	82.68
	±	6.52	6.15	10.87	5.12	7.98	6.52	24.50	
	Ours + No Meta	73.14	82.02	78.82	84.86	83.14	86.63	52.56	77.31
	±	9.41	5.09	9.94	3.55	4.87	8.52	22.06	
	Ours-l + SNIPS	86.18	83.51	84.15	89.33	85.74	86.34	76.69	84.60
	±	6.76	7.63	6.67	5.31	7.46	6.53	11.34	
	Ours-l + Onto	76.07	79.25	77.54	82.49	74.92	80.44	42.78	73.36
	±	9.21	10.10	9.19	6.45	10.17	6.37	16.56	
	Ours-l + No Meta	68.56	76.17	74.63	81.02	74.47	78.88	38.60	70.33
	±	11.06	9.20	8.21	7.41	9.16	7.67	17.39	

Table 12: Our 5-shot slot tagging results on 7 domains of the SNIPS dataset. We provide the average and the standard deviation of F1 scores over 100 episodes. Ours-l indicate that we use lowercased words for input sentences.

		We	Mu	Pl	Bo	Se	Re	Cr	Ave.
Hou et al. (2020)'s Episodes	Ave. $\|\mathcal{S}\|$	6.15	7.66	2.96	4.34	4.29	9.41	1.30	
	SimBERT	36.10	37.08	35.11	68.09	41.61	42.82	23.91	40.67
	TransferBERT	55.82	38.01	45.65	31.63	21.96	41.79	38.53	39.06
	Matching Net	21.74	10.68	39.71	58.15	24.21	32.88	**69.66**	36.72
	WPZ + BERT	46.72	40.07	50.78	68.73	60.81	55.58	67.67	55.77
	L-TapNet+CDT	**71.53**	**60.56**	**66.27**	**84.54**	**76.27**	**70.79**	62.89	**70.41**
	Ours + SNIPS	77.81	74.66	67.81	79.60	71.05	78.61	64.07	73.37
	±	7.21	7.46	7.22	6.13	7.88	5.58	10.53	
	Ours + Onto	50.56	68.54	56.11	70.17	55.66	64.78	12.86	54.10
	±	9.53	6.75	10.52	5.96	5.76	6.65	11.45	
	Ours + No Meta	40.28	53.42	45.71	63.90	42.57	59.29	14.11	45.61
	±	9.23	10.71	11.93	8.26	7.94	7.45	11.50	
Our episodes	Ave. $\|\mathcal{S}\|$	6.17.	6.99	2.66	3.73	4.15	9.07	1.27	
	Ours + SNIPS	**82.62**	**77.46**	**71.33**	**85.49**	**83.22**	**84.23**	**82.92**	**81.04**
	±	5.96	7.56	7.15	5.74	6.47	5.67	8.71	
	Ours + Onto	56.39	67.10	53.49	71.94	66.21	69.04	28.80	59.00
	±	10.67	7.50	11.18	7.43	10.02	7.68	17.67	
	Ours + No Meta	46.42	59.02	47.47	63.79	49.42	64.45	17.60	49.74
	±	9.52	9.45	11.01	7.33	10.32	7.63	7.63	
	Ours-l + SNIPS	77.42	73.45	67.05	76.85	72.54	79.54	63.44	72.90
	±	6.09	8.18	8.26	5.93	8.97	5.62	10.19	
	Ours-l + Onto	50.65	62.58	50.55	65.42	50.23	61.82	18.59	51.41
	±	8.47	7.94	8.32	6.71	9.83	6.92	12.15	
	Ours-l + No Meta	41.47	57.69	44.13	61.22	40.96	59.60	11.00	45.15
	±	8.25	12.46	7.45	6.55	11.50	8.60	10.87	

Table 13: Our 1-shot slot tagging results on Hou et al. (2020)'episodes and our own constructed episodes. We provide the average and the standard deviation of F1 scores over 100 episodes. Ours-l indicate that we use lowercased words for input sentences.