Few-shot Intent Classification and Slot Filling with Retrieved Examples

Dian Yu[*]
University of California, Davis
dianyu@ucdavis.edu

Luheng He
Google Research
luheng@google.com

Yuan Zhang
Google Research
zhangyua@google.com

Xinya Du[*]
Cornell University
xd75@cornell.edu

Panupong Pasupat
Google Research
ppasupat@google.com

Qi Li
Google Assistant
qilqil@google.com

Abstract

Few-shot learning arises in important practical scenarios, such as when a natural language understanding system needs to learn new semantic labels for an emerging, resource-scarce domain. In this paper, we explore retrieval-based methods for intent classification and slot filling tasks in few-shot settings. Retrieval-based methods make predictions based on labeled examples in the retrieval index that are similar to the input, and thus can adapt to new domains simply by changing the index without having to retrain the model. However, it is non-trivial to apply such methods on tasks with a complex label space like slot filling. To this end, we propose a span-level retrieval method that learns similar contextualized representations for spans with the same label via a novel batch-softmax objective. At inference time, we use the labels of the retrieved spans to construct the final structure with the highest aggregated score. Our method outperforms previous systems in various few-shot settings on the CLINC and SNIPS benchmarks.

1 Introduction

Few-shot learning is a crucial problem for practical language understanding applications. In the few-shot setting, the model (typically trained on source domains with abundant data) needs to adapt to a set of unseen labels in the target domain with only a few examples. For instance, when developers introduce a new product feature, a query understanding model has to learn new semantic labels from a small dataset they manage to collect.

Few-shot learning is challenging due to the imbalance in the amount of data between the source and target domains. Traditional classification methods, even with the recent advancement of pre-trained language models (Peters et al., 2018; Devlin et al., 2019), could suffer from over-fitting (Snell

et al., 2017; Triantafillou et al., 2019) or catastrophic forgetting (Wu et al., 2019) when incorporating the data-scarce target domain. On the other hand, *metric learning methods* (Weinberger et al., 2006; Vinyals et al., 2016; Snell et al., 2017) have been shown to work well in few-shot scenarios. These methods are based on modeling similarity between inputs, effectively allowing the model to be decoupled from the semantics of the output space. For example, a model would learn that the utterance *"I'd like to book a table at black horse tavern at 7 pm"* (from Figure 1) is similar to *"make me a reservation at 8"* and thus are likely to have similar semantic representations, even without knowing the semantic schema in use. Unlike learning output labels, which is difficult when examples are scarce, learning a similarity model can be done on the abundant source domain data, making such models data-efficient even in few-shot settings.

While there are many instantiations of metric learning methods (see Section 3), we focus on *retrieval-based* methods, which maintain an explicit retrieval index of labeled examples. The most basic setting of retrieval-based model for few-shot learning is: after training a similarity model and encoding target domain data into the index, we can retrieve examples most similar to the given input, and then make a prediction based on their labels. Compared to methods that do not maintain an index, such as Prototypical Networks (Snell et al., 2017), retrieval-based methods are less sensitive to outliers with few data points, and are powerful when we have abundant data in the source domain (Triantafillou et al., 2019).

However, applying retrieval-based models on tasks with a structured output space is non-trivial. For example, even if we know that the utterance in Figure 1 is similar to *"make me a reservation at 8"*, we cannot directly use its slot values (e.g., the time slot has value *"8"* which is not in the input), and not all slots in the input (e.g., *"black*

---

[*]Work done during internship at Google Research

query [CLS] I'd like to book a table at black horse tavern at 7 pm ...

0.58   0.87   0.96   0.97

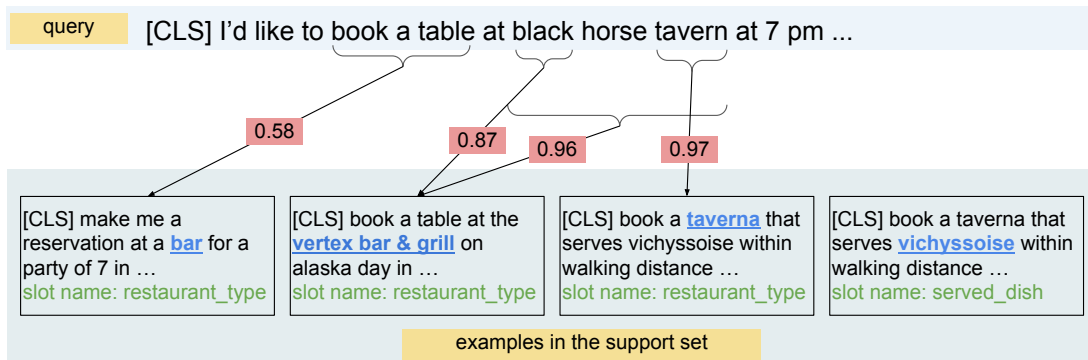| [CLS] make me a reservation at a **bar** for a party of 7 in …
 slot name: restaurant_type | [CLS] book a table at the **vertex bar & grill** on alaska day in …
 slot name: restaurant_type | [CLS] book a **taverna** that serves vichyssoise within walking distance …
 slot name: restaurant_type | [CLS] book a taverna that serves **vichyssoise** within walking distance …
 slot name: served_dish |

examples in the support set

Figure 1: Illustration of span-level retrieval for slot filling. For each span (including spans that are not valid slots such as *"book a table"*) in the input utterance, we retrieve its most similar span from the retrieval index, and then assign the slot name as the prediction with a similarity score. We use modified beam search to decode an output that maximizes the average similarity score. The gold slots are *"black horse tavern"* and *"7 pm"* in this example.

horse tavern") have counterparts in the retrieved utterance. While previous works have exploited token-level similarity methods in a BIO-tagging framework, they had to separately simulate the label transition probabilities, which might still suffer from domain shift in few-shot settings (Wiseman and Stratos, 2019; Hou et al., 2020).

In this work, we propose Retriever, a retrieval-based framework that tackles both classification and span-level prediction tasks. The core idea is to match *token spans* in an input to the most similar labeled spans in a retrieval index. For example, for the span *"7 pm"* in the input utterance, the model retrieves *"8"* as a similar span (given their surrounding contexts), thus predicting that *"7 pm"* has the same slot name time as *"8"*. During training, we fine-tune a two-tower model with BERT (Devlin et al., 2019) encoders, along with a novel batch softmax objective, to encourage high similarity between contextualized span representations sharing the same label. At inference time, we retrieve the most similar span from the few-shot examples for every potential input span, and then decode a structured output that has the highest average span similarity score.

We show that our proposed method is effective on both few-shot intent classification and slot-filling tasks, when evaluated on CLINC (Larson et al., 2019) and SNIPS (Coucke et al., 2018) datasets, respectively. Experimental results show that Retriever achieves high accuracy on few-shot target domains without retraining on the target data. For example, it outperforms the strongest baseline by 4.45% on SNIPS for the slot-filling task.

2 Benefits of Retriever

In addition to being more robust against overfitting and catastrophic forgetting problems, which are essential in few-shot learning settings, our proposed method has multiple advantages overs strong baselines. For instance, if the scheme is changed or some prediction bugs need to be fixed, there is minimum re-training required. More importantly, compared to classification or Prototypical Networks which require adding arbitrary number of instances to the training data and hope that the model will predict as expected (Yu et al., 2019; Liang et al., 2020), Retriever can guarantee the prediction when a similar query is encountered. At the same time, Retriever is more interpretable where the retrieved examples can serve as explanations. In addition, different from the simplified assumption that one utterance may only have one intent (Bunt, 2009; Yu and Yu, 2019), Retriever can be used to predict multiple labels. Lastly, because Retriever does not need to model transition probability, the decoding procedure can be paralleled and potentially be modified to be non-autoregressive for speedup. We can also tune threshold (explained in Section 5.2) to change precision and recall according to use case requirements.

3 Related Work

Few-shot metric learning   Metric learning methods target at learning representations through distance functions. Koch et al. (2015) proposed Siamese Networks which differentiated input examples with contrastive and triplet loss functions (Schroff et al., 2015) on positive and negative pairs. While they are more data efficient for new

classes than linear classifiers, Siamese Networks are hard to train due to weak pairs sampled from training batch (Gillick et al., 2019). In comparison, Prototypical Networks (Snell et al., 2017) proposed to compute class representations by averaging embeddings of support examples for each class. These methods have been mostly explored in computer vision and text classification (Geng et al., 2019; Yu et al., 2018), and consistently outperform Siamese Networks and retrieval-based methods such as *k*-nearest-neighbors, especially when there are more classes and fewer annotated examples (Triantafillou et al., 2019; Sun et al., 2019). However, newly added examples which are outliers may change the prototypical representations dramatically that can harm all predictions on the class. In addition, these methods do not perform well when there are more annotated data available per class (Triantafillou et al., 2019).

Recently, Wang et al. (2019) showed that a simple nearest neighbor model with feature transformations can achieve competitive results with the state-of-the-art methods on image classification. Inspired by their work, we train our retrieval-based model with a novel batch softmax objective.

Metric learning in language understanding Utilizing relevant examples to boost model performance has been applied to language modeling (Khandelwal et al., 2020), question answering (Guu et al., 2020; Lewis et al., 2020), machine translation (Zhang et al., 2018), and text generation (Peng et al., 2019). Recently, metric learning has been applied to intent classification (Sun et al., 2019; Krone et al., 2020). Ren and Xue (2020) trained Siamese Networks before learning a linear layer for intent classification and showed competitive results with traditional methods in the full-data setting. Similar ideas are also extended to sequence labeling tasks such as named entity recognition (NER, Wiseman and Stratos, 2019; Fritzler et al., 2019) by maximizing the similarity scores between contextual tokens representations sharing the same label. Krone et al. (2020) utilized Prototypical Networks to learn intent and slot name prototype representations and classified each token to its closest prototype. They showed better results than meta-learning, another prevalent few-shot learning method (Finn et al., 2017; Mishra et al., 2018). In order to consider label dependencies that are essential in slot tagging tasks (Huang et al., 2015), Hou et al. (2020) proposed a collapsed dependency

transfer (CDT) mechanism by simulating transition scores for the target domain from transition probabilities among BIO labels in the source domain, outperforming previous methods on slot filling by a large margin. Yang and Katiyar (2020) further explored the transition probability by evenly distributing the collapsed transition scores to the target domain to maintain a valid distribution. However, this simulation is noisy and the difference between the source and target domains can result in biased transition probabilities.

The most similar approach to ours is a concurrent work from Ziyadi et al. (2020), which learns span boundaries and sentence similarities before retrieving the most similar span, inspired by question-answering models. Even though this approach predicts spans before retrieving on the span level and thus bypasses the problem of transition probability in previous research, it only achieves unsatisfactory results. Different from these researches, we propose to learn span representations using a batch softmax objective without having to explicitly learn span boundaries. Our method achieves more accurate slot and intent prediction than previous methods in the few-shot setting.

4 Setup

We consider two tasks where the input is an utterance $\mathbf{x}$ with tokens $x_1, \ldots, x_n$ and the output is some structure $\mathbf{y}$. For the *slot filling* task, the output $\mathbf{y}$ is a set of non-overlapping labeled spans $\{(r_i, \ell_i)\}_{i=1}^m$ where r_i is a span of $\mathbf{x}$ (e.g., *"7 pm"*) and ℓ_i is a slot name (e.g., `time`). For the *intent classification* task, the output $\mathbf{y}$ is simply an intent label ℓ for the whole utterance $\mathbf{x}$. For notational consistency, we view intent classification as predicting a labeled span (r, ℓ) where $r = \mathbf{x}_{1:n}$.

In the few-shot setup, examples $(\mathbf{x}, \mathbf{y})$ are divided into *source* and *target* domains. Examples in the target domain may contain some labels ℓ that are unseen in the source domain. The model will be given ample training data from the source domain, but only a few training examples from the target domain. For instance, the model receives only $K = 5$ examples for each unseen label. The model can be evaluated on test data from both domains.

5 Model

We propose a retrieval-based model, `Retriever`, for intent classification and slot filling in the few-shot setting. Figure 1 illustrates our approach. At

a high level, from examples $(\mathbf{x}, \mathbf{y})$ in the target training data (and optionally the source training data), we construct a retrieval index consisting of labeled spans (r, ℓ) from $\mathbf{y}$. Given a test utterance $\mathbf{x}$, for each span of interest in $\mathbf{x}$ (all spans $\mathbf{x}_{i:j}$ for slot filling; only $\mathbf{x}_{1:n}$ for intent classification), we retrieve the most similar labeled spans (r, ℓ) from the index, and then use them to decode an output $\mathbf{y}$ that maximizes the average span similarity score.

The use of retrieval provides several benefits. For instance, we empirically show in Section 7.1 that the model does not suffer from catastrophic forgetting because both source and target data are present in the retrieval index. Class imbalance can also be directly mitigated in the retrieval index. Additionally, since the trained model is non-parametric, we could replace the retrieval index to handle different target domains without having to retrain the model. This also means that the model does not need access to target data during training, unlike traditional classification methods.

5.1 Retriever

The retriever is the only trainable component in our model. Given a query span $r' = \mathbf{x}_{i:j}$ from the input $\mathbf{x}$, the retriever returns a set of labeled spans (r, ℓ) with the highest similarity scores $s(\mathbf{z}, \mathbf{z}')$, where $\mathbf{z} = E(r)$ and $\mathbf{z}' = E(r')$ are the contextualized embedding vectors of r and r', respectively.

Similarity score To compute the contextualized embeddings $\mathbf{z}$ and $\mathbf{z}'$ of spans r and r', we first apply a Transformer model initialized with pre-trained BERT on the utterances where r and r' come from. For slot filling, we follow Toshniwal et al. (2020) and define the span embedding as the concatenated embeddings of the its first and last wordpieces. For intent classification, we use the embedding of the [CLS] token. We then define $s(\mathbf{z}, \mathbf{z}')$ as the dot product[1] between $\mathbf{z}$ and $\mathbf{z}'$.

Training with batch softmax We use examples from the source domain to train Retriever. Let $\ell_1, \ldots, \ell_N$ be the N class labels (slot or intent labels) in the source domain. To construct a training batch, for each class label ℓ_i, we sample B spans $r_i^1, \ldots, r_i^B$ from the training data with that label, and compute their embeddings $\mathbf{z}_i^1, \ldots, \mathbf{z}_i^B$.

---

[1]We experimented with affine transformation as well as cosine similarity but did not see any performance gain. For intent classification, using the [CLS] token achieves better results than averaging word embeddings.

Then, for each query span r_i^j, we compute similarity scores against all other spans in the batch to form a $B \times N$ similarity matrix:

$$
S_i^j = \begin{bmatrix} s(\mathbf{z}_i^j, \mathbf{z}_1^1) & s(\mathbf{z}_i^j, \mathbf{z}_2^1) & \ldots & s(\mathbf{z}_i^j, \mathbf{z}_N^1) \\ s(\mathbf{z}_i^j, \mathbf{z}_1^2) & s(\mathbf{z}_i^j, \mathbf{z}_2^2) & \ldots & s(\mathbf{z}_i^j, \mathbf{z}_N^2) \\ \vdots & \vdots & \ddots & \vdots \\ s(\mathbf{z}_i^j, \mathbf{z}_1^B) & s(\mathbf{z}_i^j, \mathbf{z}_2^B) & \ldots & s(\mathbf{z}_i^j, \mathbf{z}_N^B) \end{bmatrix}.
\tag{1}
$$

We now summarize the score between r_i^j and each label $\ell_{i'}$ by applying a reduction function (defined shortly) along each column to get a $1 \times N$ vector:

$$
\hat{S}_i^j = \begin{bmatrix} s(\mathbf{z}_i^j, \mathbf{z}_1^*) & s(\mathbf{z}_i^j, \mathbf{z}_2^*) & \ldots & s(\mathbf{z}_i^j, \mathbf{z}_N^*) \end{bmatrix}
\tag{2}
$$

We use the softmax of $\hat{S}_i^j$ as the model's probability distribution on the label of r_i^j. The model is then trained to optimize the cross-entropy loss on this distribution against the gold label ℓ_i.

We experiment with three reduction functions, *mean* (Eq. 3), *max* (Eq. 4), and *min-max* (Eq. 5):

$$
s(\mathbf{z}_i^j, \mathbf{z}_{i'}^*) = \frac{1}{B} \sum_{j'=1}^{B} s(\mathbf{z}_i^j, \mathbf{z}_{i'}^{j'}) = s\left(\mathbf{z}_i^j, \frac{1}{B} \sum_{j'=1}^{B} \mathbf{z}_{i'}^{j'} \right)
\tag{3}
$$

$$
s(\mathbf{z}_i^j, \mathbf{z}_{i'}^*) = \max_{\substack{1 \le j' \le B; \\ j' \ne j \text{ if } i = i'}} s(\mathbf{z}_i^j, \mathbf{z}_{i'}^{j'})
\tag{4}
$$

$$
s(\mathbf{z}_i^j, \mathbf{z}_{i'}^*) = \begin{cases} \min_{1 \le j' \le B} s(\mathbf{z}_i^j, \mathbf{z}_{i'}^{j'}), & \text{if } i = i' \\ \max_{1 \le j' \le B} s(\mathbf{z}_i^j, \mathbf{z}_{i'}^{j'}), & \text{otherwise} \end{cases}
\tag{5}
$$

The *mean* reduction averages embeddings of the spans with the same label and is equivalent to Prototypical Networks. Similar to hard negative sampling to increase margins among classes (Schroff et al., 2015; Roth et al., 2020; Yang et al., 2019), *max* takes the most similar span to the query (excluding the query itself) as the label representation, while *min-max* takes the least similar span when considering spans with the same label as the query.

5.2 Inference

After training, we build a dense retrieval index where each entry (r, ℓ) is indexed by $\mathbf{z} = E(r)$. The entries (r, ℓ) come from examples $(\mathbf{x}, \mathbf{y})$ in the *support set* which, depending on the setting, could be just the target training data or a mixture of source and target data. For each query span r' of the input utterance $\mathbf{x}$, we embed the span and compute the similarity scores against all index entries.

Intent classification For intent classification, both index entries and query spans are restricted to the whole utterances. The entire process thus boils down to retrieving the most similar utterance based on the [CLS] token embedding. We simply output the intent label of the retrieved utterance.

Slot filling In contrast to BIO decoding for token-level similarity models (Hou et al., 2020), decoding with span retrieval results poses unique challenges as gold span boundaries are not known a priori. Hence, we use a modified beam search procedure with simple heuristics to compose the spans.

Specifically, for each of the $n \times m$ spans in an utterance of length n (where the hyperparameter m is the maximum span length), we retrieve the most similar span from the retrieval index. Then we normalize the similarity scores by L2-norm so that they are within the range $[0, 1]$. Since we do not explicitly predict span boundaries, all $n \times m$ spans, including non-meaningful ones (e.g., *"book a"*), will have a retrieved span. Such non-meaningful spans should be dissimilar to any labeled span in the retrieval index. We thus choose to filter the spans with a score threshold to get a smaller set of candidate spans. In addition, we adjust the threshold dynamically (by reducing the threshold for a few times) if no span is above the current threshold.

Once we get candidate spans with similarity scores, we use beam search to decode a set of spans with maximum average scores.[2] We go through the list of candidate spans in the descending order of their similarity scores. For each candidate span, we expand beam states if the span does not overlap with the existing spans in the beam. The search beams are pruned based on the average similarity score of the spans included so far. Lastly, we add spans in the filtered set which do not overlap with the final beam.

Beam search can avoid suboptimal decisions that a greedy algorithm would make. For instance, if we greedily process the example in Figure 1, *"black"* and *"tavern"* would become two independent spans, even though their average similarity score is lower than the correct span *"black horse tavern"*. Nevertheless, beam search is prone to mixing up span boundaries and occasionally predicts consecutive partial spans such as *"black horse"* and *"tavern"* as individual slots. Since consecutive spans of the *same* slot label are rare in slot filling

datasets, we merge the two spans if their retrieval scores are within a certain range:

$$r_{i:j}, r_{j:k} = \begin{cases} r_{i:k} & \text{if } |s(\mathbf{z}_{i:j}, \mathbf{z}') - s(\mathbf{z}_{j:k}, \mathbf{z}'')| < \lambda \\ r_{i:j}, r_{j:k} & \text{otherwise} \end{cases}$$

where $r_{i:j}$ and $r_{j:k}$ are two consecutive potential spans sharing the same label, and $\mathbf{z}'$ and $\mathbf{z}''$ are the embeddings of their retrieved spans, respectively ($r_{i:k}$ indicates merging the two spans into one span; λ is the merge threshold where $\lambda = 1$ means always merge and $\lambda = 0$ means never merge).

6   Experiments and Results

We evaluate our proposed approach on two datasets: CLINC (Larson et al., 2019) for intent classification and SNIPS (Coucke et al., 2018) for slot filling. Note that we use *max* (Eq. 4) as the reduction function for both tasks since it empirically yields the best results. The effect of reduction functions will be analyzed later in Section 7.1.

6.1   Intent Classification

The CLINC intent classification dataset (Larson et al., 2019) contains utterances from 10 intent categories (e.g., "travel"), each containing 15 intents (e.g., "flight_status", "book_flight"). To simulate the few-shot scenario where new domains and intents are introduced, we designate n_c categories and n_i intents per category as the source domain (with all 100 training examples per intent), and use the remaining $150 - n_c \times n_i$ intents as the target domain. We experiment with $(n_c, n_i) = (10, 10)$, $(8, 10)$, and $(5, 15)$. [3] The target training data contains either 2 or 5 examples per target intent.

We compare our proposed method `Retriever` with a classification model `BERT fine-tune` and a Prototypical Network model `Proto`. The former learns a linear classifier on top of BERT embeddings (Devlin et al., 2019), and the latter learns class representations based on Prototypical Networks.[4] We also show results with the initial BERT checkpoint without training (`Proto`[frz], `Retriever`[frz]). We use the same batch size for all models, and tune other hyperparameters on the development set before testing.

---

[2] We use beam search for simplicity. Other search methods such as Viterbi algorithm (Forney, 1973) can also be used.

[3] $(n_c, n_i) = (10, 10)$ simulates the situation where all the categories are known, but we adapt to new intents in all 10 categories; $(n_c, n_i) = (5, 15)$ simulates the situation where we adapt to 5 entirely new intent categories.

[4] Previous work show that Prototypical Networks outperforms other optimization-based and metric-learning models such as MAML in (intent) classification tasks (Triantafillou et al., 2019; Krone et al., 2020).

	support_set=all			support_set=balance			support_set=tgt
	tgt	src	avg	tgt	src	avg	tgt
Initial BERT							
`Proto`[frz]	14.07	25.02	21.37	-	-	-	-
`Retriever`[frz]	8.24	54.76	39.25	22.09	25.29	24.22	37.93
Pre-train on src domain							
`BERT fine-tune`	-	96.51	-	-	-	-	-
`Proto`	75.02	95.73	88.83	-	-	-	-
`Retriever`	62.69	**97.08**	85.62	75.93	95.44	88.94	88.53
`Retriever min-max`	66.00	96.64	86.43	71.82	95.14	87.37	86.38
Fine-tune on tgt domain							
`BERT fine-tune`	78.89	43.91	55.57	-	-	-	-
`Proto`	80.44	95.57	90.53	-	-	-	90.35
`Retriever`	66.76	96.95	86.89	79.20	95.50	90.07	**91.16**
`Retriever min-max`	67.64	96.84	87.11	77.60	95.35	89.43	89.56
Fine-tune on tgt domain with src data							
`BERT fine-tune`	72.00	95.18	87.45	-	-	-	-
`Proto`	83.33	94.82	90.99	-	-	-	90.22
`Retriever`	69.51	97.04	87.86	**84.95**	95.41	**91.92**	90.78
`Retriever min-max`	71.35	96.96	88.42	81.00	94.55	90.03	89.82

Table 1: Intent accuracy on CLINC for $n_c = 10, n_i = 10$ with 5-shots. Our retrieval-based method outperform BERT fine-tune and Prototypical Networks in both target and source domains. We report results for our method when the support set consists of all examples in the source and target domains (all), when the support set consists of balanced few-shot number of examples for intents in both source and target domains (balance), and when the support set consists of examples of the target domain only (tgt) which serves as an upper-bound.

Evaluation We sample domains and intents three times for each (n_c, n_i) setting, and report average prediction accuracy. We report accuracy on intents from the target domain (tgt), source domain (src), and the macro average across all intents (avg).

In addition to applying the model to the target domain after pre-training on the source domain without re-training (*Pre-train on src domain*), we also evaluate the model performance with fine-tuning. We re-train the model with either target domain data only (*Fine-tune on tgt domain*) or a combination of source and target domain data (*Fine-tune on tgt domain with src data*).

Moreover, we evaluate the models with the following support set variations: with target domain data and all data in the source domain (support_set=all), with equal number of examples (same as the few-shot number) per intent (support_set=balance), and with only examples from the target domain (support_set=tgt). The last one serves as an upper-bound for the target domain accuracy.

Results Table 1 shows the results for $(n_c, n_i) = (10, 10)$ and 5 examples per target intent; results on other settings exhibit the same patterns (See Appendix A.3). We observe that `Retriever` per-

forms the best on the source domain (97.08%) before fine-tuning. `Retriever` also achieves the highest accuracy on the target domain (84.95%) after fine-tuning, while maintaining competitive performance on the source domain (95.41%) among all the methods.

6.2 Slot Filling

SNIPS (Coucke et al., 2018) is a slot filling dataset containing 39 slot names from 7 different domains: GetWeather (GW), PlayMusic (PM), AddToPlaylist (ATP), RateBook (RB), FindScreeningEvent (FSE), BookRestaurant (BR), and SearchCreativeWork (SCW). Following Hou et al. (2020), we train models on five source domains, use a sixth one for development, and test on the remaining domain. We directly use the K-shot split provided by Hou et al. (2020), where the support set consists of the minimum number of utterances such that at least K instances exist for each slot name. We also set $K = 5$ in our experiment. Appendix A.2 contains further details about the setup.

We compare against two baselines and three models from the previous work. `BERT Tagging` is a BERT-based BIO tagging model (Devlin et al., 2019) fine-tuned on the testing domain

	GW	PM	ATP	RB	FSE	BR	SCW	Average F1
Classification-based								
`BERT Tagging`	59.41	42.00	46.07	20.74	28.20	67.75	58.61	46.11
Token-level								
`SimilarToken`[frz]	53.46	54.13	42.81	75.54	57.10	55.30	32.38	52.96
`MatchingToken`	36.67	33.67	52.60	69.09	38.42	33.28	72.10	47.98
`ProtoToken`	67.82	55.99	46.02	72.17	**73.59**	60.18	**66.89**	63.24
`L-TapNet+CDT+Proto`	-	-	-	-	-	-	-	67.27
`L-Proto+CDT`[pw]*	74.68	56.73	52.20	78.79	80.61	69.59	67.46	68.58
`L-TapNet+CDT+Proto`[pw]*	71.64	67.16	75.88	84.38	82.58	70.05	73.41	75.01
Span-level (ours)								
`Proto`[frz]	39.47	38.35	47.68	69.36	38.60	42.39	19.90	42.25
`Proto`	64.47	53.97	54.64	73.37	42.89	62.48	27.76	54.23
`Retriever`[frz]	63.39	46.01	51.11	79.65	62.42	62.13	33.85	56.94
`Retriever`	**82.95**	**61.74**	**71.75**	**81.65**	73.10	**79.54**	51.35	**71.72**

Table 2: Results on SNIPS test data with 5-shot support sets. Our span-based retrieval model outperforms previous classification-based and token-level retrieval models even without label semantics. Classification-based and token-level results are reported in Hou et al. (2020). *Pair-wise embeddings (marked with [pw]) are expensive at inference time, so we do not compare our method with these directly.

after training on the source domains, while `SimilarToken`[frz] uses BERT embeddings to retrieve the most similar token based on cosine similarity without any training. `MatchingToken` and `ProtoToken` are two token-level methods that leveraged Matching Networks (Vinyals et al., 2016) and Prototypical Networks (Snell et al., 2017) respectively. `L-TapNet+CDT+proto` (Hou et al., 2020) is an adaptation of TapNet (Yoon et al., 2019) with label semantics, CDT transition probabilities, and Prototypical Networks.

We experiment with several variants of our proposed method. `Proto` trains Prototypical Networks to compute span class representations. `Retriever` retrieves the most similar slot example for each span. Both methods use the same decoding method. Similar to `SimilarToken`[frz], `Proto`[frz] and `Retriever`[frz] use the original BERT embeddings without any training. All models are trained on source domains and early stopped based on performance on the development domains.

Evaluation We report F1 scores for each testing domain in a cross-validation episodic fashion. Following Hou et al. (2020), we evaluate each testing domain by sampling 100 different support sets and ten exclusive query utterances for each support set. We calculate F1 scores for each episode and report average F1 scores across 100 episodes.

Results Table 2 summarizes the experiment results on the SNIPS dataset. Our span-level method

(`Retriever`) achieves higher averaged F1 than all five baselines, outperforming the strongest token-level method (`L-TapNet+CDT+proto`) by 4.45%. This shows that our model is effective at span-level predictions. More importantly, the better performance suggests that our span-level `Retriever` model is more efficient at capturing span structures compared to simulated dependencies as our method does not suffer from the potential discrepancy in the transition probabilities between the target and source domains.

Although Hou et al. (2020) showed that adding pairwise embeddings with cross-attention yielded much better performance, this method is expensive both in memory and computation at inference time, especially when the support set is large (Humeau et al., 2019). For fair comparison, we do not directly compare with methods using pairwise embeddings (methods with [pw] in Table 2). Note that our method with pre-computed support example embeddings even outperforms `L-Proto+CDT`[pw] with less memory and computation cost.

7 Analysis

7.1 Intent Classification

Models without re-training The *pre-train on src domain* section in Table 1 shows the results of models that are only pre-trained on the source domains but **not** fine-tuned on the target domains. Classification models such as `BERT fine-tune` cannot make predictions on target domains in this

setting. In contrast, even without seeing any target domain examples during training, retrieval-based models can still make predictions on new domains by simply including new examples in the support sets. With support_set=all, `Retriever` achieves 97.08% on the source domain while `Proto` performs worse than `BERT fine-tune`, consistent with previous findings (Triantafillou et al., 2019). `Retriever` achieves the best accuracy (75.93%) on target domains with a balanced support set on all intents (support_set=balance). More importantly, `Retriever` also achieves competitive accuracy on source domains (95.44%), demonstrating that our proposed model achieves the best of both worlds even without re-training on new domains.

Varying the support set at inference time The construction of the support set is critical to retrieval-based methods. In Table 1, we present the model performances under different support settings (all, balance, tgt). The support_set=tgt setting serves as an upper bound for the target domain accuracy for both `Retriever` and `Proto` methods. In general, `Retriever` achieve the best performance on the source domain intents when we use full support sets (support_set=all). In comparison, if we use a balanced support set (support_set=balance), we can achieve much higher accuracy on the target domain while having a slight degradation on the source domain intents prediction. This is because full support sets have more source domain examples to increase confusion over target domains.

Data for fine-tuning The *Fine-tune on tgt domain* section in Table 1 shows different model behaviors when fine-tuned on the target domain data directly. While `BERT fine-tune` achieves high accuracy (78.89%) on the target domain, it suffers from catastrophic forgetting on the source domain (43.91%). On the other hand, `Proto` and `Retriever` can get high accuracy on the target domain (80.44% and 79.20%) while maintaining high performance on the source domain.

When we combine data from the source domain, we observe performance gains in all the models under the *Fine-tune on tgt domain with src data* section. Specifically, we add few-shot source domain examples as contrastive examples for the models to learn better utterance/class representations for `Retriever` and `Proto`. Results show that accuracy on the target domain increases by over 3% compared to only using target domain data. This

	tgt	src	avg
`BERT fine-tune`	-	-	-
`Proto`	+12.89	-0.51	+5.18
`Retriever`	**+14.60**	**-0.14**	**+6.11**
`Retriever min-max`	+10.79	-0.20	+4.47

Table 3: Improvement (%) over `BERT fine-tune` on target (tgt), source (src), and average (avg), after fine-tuning on the 5-shot support sets. Numbers are averaged over different (n_c, n_i) data samples.

suggests that unlike other retrieval-based methods such as kNN, `Retriever` does not require a large support set to guarantee prediction accuracy.

Impact of reduction functions We compare the reduction functions proposed in Section 5.1 and found that `max` performs the best. Since `mean` is equivalent to Prototypical Networks, we compared to `Proto` directly in the experiments. `min-max` is more intuitive in contrasting with least similar examples within the same class compared to `max`. However, its performance is worse than `max`. We speculate the reason to be that we retrieve the example with the maximum score at inference time so that the boundary margin may not be utilized.

Performance over different settings Table 3 shows the average improvement of our methods over the `BERT fine-tune` baseline, where all models are fine-tuned on the target domain with a balanced few-shot dataset after training on the source domain (same as *Fine-tune on tgt domain with src data* section in Table 1). Both `Proto` and `Retriever` outperforms the baseline on the target domains with a large margin, and `Retriever` has the best improvement on all intents on average.

7.2 Slot Filling

We note that `Retriever` outperforms the strongest baselines but reaches a low score on the SCW domain. This may be due to the bigger difference between the test (SCW) and the development domain (GW) including the size of the support set and their respective slot names. We also found that from all the correctly predicted slot spans, 96.73% predicted the correct slot names. This shows that the majority of the errors come from querying with invalid spans. We believe that span-based pre-training such as Span-BERT (Joshi et al., 2020) could make our proposed method achieve better results.

beam size	merge threshold	avg. F1
1	0.99	70.10
5	0.99	71.47
10	0.99	71.72
10	0	70.43
10	1	72.10

Table 4: Ablation study on beam size and merge condition. Merge threshold of 0 means never merge and 1 means always merge. Using larger beam and merging consecutive spans improve F1 scores.

Analyzing `Proto` From Table 2, `Retriever` outperforms `Proto` by 17% when training the span representations. We conjecture that this is caused by `Proto` learning noisy prototype. Compared to `Retriever`, the similarity scores between the spans and their corresponding class representations are low, indicating that the span-level prototypes may not be clearly separated.

Ablation on decoding method Table 4 compares beam search to greedy search. Results suggest that beam search with larger beam sizes achieve better F1 scores. As discussed in Section 5.2, we merge same-label spans during inference based on a score threshold. As shown in Table 4, merging spans results in a 1.67% F1 gain (70.43% vs 72.10%) under the same beam size.

Error Analysis We find that the main problem of our proposed model is that tokens surrounding the gold span may contain excessive contextual information so that these surrounding invalid spans retrieve corresponding spans with high similarities. For instance, in the query "add my track to old school metal playlist", the token "playlist" retrieves an actual playlist span with a high similarity score. Another major issue is that the similarity score retrieved by a partial of the gold span sometimes is higher than that retrieved by the whole span. Our ablation results on merge threshold shown in Table 4 also suggest that partial spans may retrieve complete spans individually so that if we merge consecutive spans with the same slot name, we can achieve higher F1 scores.

8   Conclusion

In this paper, we propose a retrieval-based method, `Retriever`, for few-shot intent classification and slot filling. We conduct extensive experiments to compare different model variants and baselines, and show that our proposed approach is effective in the few-shot learning scenario. We believe that our method can also work on open domain dialog tasks where annotations may be more scarce and other text classification tasks. In the future, we plan to extend our method to predict more complex structures with span-based retrieval.

9   Ethical Considerations

Our intended use case is few-shot domain adaption to new classes. Our experiments are done on English data, but the method is not English-specific. We use 8 Cloud TPUs V2 cores[5] for training and one V100 GPU for inference. Since our model does not have to be retrained for the new domains, it can reduce the resources needed when applying such systems. We claim that our proposed method outperforms baselines on few-shot slot filling and intent classification examples. Our experiments mainly focus on the 5-shot setting and the 2-shot setting, which are typical testing scenarios applied by previous work with the same claim.

Acknowledgments

References

Harry Bunt. 2009. The dit++ taxonomy for functional dialogue markup. In *AAMAS 2009 Workshop, Towards a Standard Markup Language for Embodied Dialogue Acts*, pages 13–24.

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.

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.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of

---

[5]https://cloud.google.com/tpu

deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *ICML'17*, pages 1126–1135. JMLR.org.

G. D. Forney. 1973. The viterbi algorithm. *Proc. of the IEEE*, 61:268 – 278.

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 '19, page 993–1000, New York, NY, USA. Association for Computing Machinery.

Ruiying Geng, Binhua Li, Yongbin Li, Xiaodan Zhu, Ping Jian, and Jian Sun. 2019. Induction networks for few-shot text classification. 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 3904–3913, Hong Kong, China. Association for Computational Linguistics.

Daniel Gillick, Sayali Kulkarni, Larry Lansing, Alessandro Presta, Jason Baldridge, Eugene Ie, and Diego Garcia-Olano. 2019. Learning dense representations for entity retrieval. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 528–537, Hong Kong, China. Association for Computational Linguistics.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Realm: Retrieval-augmented language model pre-training.

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 *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1381–1393, Online. Association for Computational Linguistics.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *CoRR*, abs/1508.01991.

Samuel Humeau, Kurt Shuster, Marie-Anne Lachaux, and Jason Weston. 2019. Real-time inference in multi-sentence tasks with deep pretrained transformers. *CoRR*, abs/1905.01969.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2020. SpanBERT: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2020. Generalization through memorization: Nearest neighbor language models. In *International Conference on Learning Representations*.

Gregory R. Koch, Richard Zemel, and Ruslan Salakhutdinov. 2015. Siamese neural networks for one-shot image recognition. In *ICML Deep Learning workshop*.

Jason Krone, Yi Zhang, and Mona Diab. 2020. Learning to classify intents and slot labels given a handful of examples. In *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI*, pages 96–108, Online. Association for Computational Linguistics.

Stefan Larson, Anish Mahendran, Joseph J. Peper, Christopher Clarke, Andrew Lee, Parker Hill, Jonathan K. Kummerfeld, Kevin Leach, Michael A. Laurenzano, Lingjia Tang, and Jason Mars. 2019. An evaluation dataset for intent classification and out-of-scope prediction. 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 1311–1316, Hong Kong, China. Association for Computational Linguistics.

Patrick Lewis, Ethan Perez, Aleksandara Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks.

Kai-Hui Liang, Austin Chau, Yu Li, Xueyuan Lu, Dian Yu, Mingyang Zhou, Ishan Jain, Sam Davidson, Josh Arnold, Minh Nguyen, and Zhou Yu. 2020. Gunrock 2.0: A User Adaptive Social Conversational System. *Alexa Prize Proceedings*.

Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. 2018. A simple neural attentive meta-learner. In *International Conference on Learning Representations*.

Hao Peng, Ankur Parikh, Manaal Faruqui, Bhuwan Dhingra, and Dipanjan Das. 2019. Text generation with exemplar-based adaptive decoding. 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 2555–2565, Minneapolis, Minnesota. Association for Computational Linguistics.

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

F. Ren and S. Xue. 2020. Intention detection based on siamese neural network with triplet loss. *IEEE Access*, 8:82242–82254.

Karsten Roth, Timo Milbich, Samarth Sinha, Prateek Gupta, Björn Ommer, and Joseph Paul Cohen. 2020. Revisiting training strategies and generalization performance in deep metric learning.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4077–4087. Curran Associates, Inc.

Shengli Sun, Qingfeng Sun, Kevin Zhou, and Tengchao Lv. 2019. Hierarchical attention prototypical networks for few-shot text classification. 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 476–485, Hong Kong, China. Association for Computational Linguistics.

Shubham Toshniwal, Haoyue Shi, Bowen Shi, Lingyu Gao, Karen Livescu, and Kevin Gimpel. 2020. A cross-task analysis of text span representations. In *Proceedings of the 5th Workshop on Representation Learning for NLP*, pages 166–176, Online. Association for Computational Linguistics.

Eleni Triantafillou, Tyler Zhu, Vincent Dumoulin, Pascal Lamblin, Kelvin Xu, Ross Goroshin, Carles Gelada, Kevin Swersky, Pierre-Antoine Manzagol, and Hugo Larochelle. 2019. Meta-dataset: A dataset of datasets for learning to learn from few examples. *CoRR*, abs/1903.03096.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, koray kavukcuoglu, and Daan Wierstra. 2016. Matching networks for one shot learning. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 3630–3638. Curran Associates, Inc.

Yan Wang, Wei-Lun Chao, Kilian Q. Weinberger, and Laurens van der Maaten. 2019. Simpleshot: Revisiting nearest-neighbor classification for few-shot learning.

Kilian Q Weinberger, John Blitzer, and Lawrence Saul. 2006. Distance metric learning for large margin nearest neighbor classification. In *Advances in Neural Information Processing Systems*, volume 18, pages 1473–1480. MIT Press.

Sam Wiseman and Karl Stratos. 2019. Label-agnostic sequence labeling by copying nearest neighbors. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5363–5369, Florence, Italy. Association for Computational Linguistics.

Chien-Sheng Wu, Andrea Madotto, Ehsan Hosseini-Asl, Caiming Xiong, Richard Socher, and Pascale Fung. 2019. Transferable multi-domain state generator for task-oriented dialogue systems. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 808–819, Florence, Italy. Association for Computational Linguistics.

Yi Yang and Arzoo Katiyar. 2020. Simple and effective few-shot named entity recognition with structured nearest neighbor learning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6365–6375, Online. Association for Computational Linguistics.

Yinfei Yang, Gustavo Hernandez Abrego, Steve Yuan, Mandy Guo, Qinlan Shen, Daniel Cer, Yun-hsuan Sung, Brian Strope, and Ray Kurzweil. 2019. Improving multilingual sentence embedding using bidirectional dual encoder with additive margin softmax. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 5370–5378. International Joint Conferences on Artificial Intelligence Organization.

Sung Whan Yoon, Jun Seo, and Jaekyun Moon. 2019. TapNet: Neural network augmented with task-adaptive projection for few-shot learning. In *Proceedings of Machine Learning Research*, volume 97 of *Proceedings of Machine Learning Research*, pages 7115–7123, Long Beach, California, USA. PMLR.

Dian Yu, Michelle Cohn, Yi Mang Yang, Chun Yen Chen, Weiming Wen, Jiaping Zhang, Mingyang Zhou, Kevin Jesse, Austin Chau, Antara Bhowmick, Shreenath Iyer, Giritheja Sreenivasulu, Sam Davidson, Ashwin Bhandare, and Zhou Yu. 2019. Gunrock: A social bot for complex and engaging long conversations. 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): System Demonstrations*, pages 79–84, Hong Kong, China. Association for Computational Linguistics.

Dian Yu and Zhou Yu. 2019. Midas: A dialog act annotation scheme for open domain human machine spoken conversations.

Mo Yu, Xiaoxiao Guo, Jinfeng Yi, Shiyu Chang, Saloni Potdar, Yu Cheng, Gerald Tesauro, Haoyu Wang, and Bowen Zhou. 2018. Diverse few-shot text classification with multiple metrics. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1206–1215, New Orleans, Louisiana. Association for Computational Linguistics.

Jingyi Zhang, Masao Utiyama, Eiichro Sumita, Graham Neubig, and Satoshi Nakamura. 2018. Guiding neural machine translation with retrieved translation pieces. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1325–1335, New Orleans, Louisiana. Association for Computational Linguistics.

Morteza Ziyadi, Yuting Sun, Abhishek Goswami, Jade Huang, and Weizhu Chen. 2020. Example-based named entity recognition.

A Appendices

A.1 Implementation details

We use the public uncased BERT-base model from https://github.com/google-research/bert for embedding spans. Our implementation is adapted from https://github.com/google-research/bert/blob/master/run_classifier.py. Since the span embedder in the retriever is the only trainable component in our model, the number of parameters is the same as the initial BERT model.

On SNIPS, we set the initial learning rate to be 2×10^{-5} with 10% data for warmup. We set per-class batch size to be 5 for 5-shot experiments. We use F1 score on the development domain as the metric for early stopping. For decoding, we set $m = 7$ to be the maximum span length and $\lambda = 0.99$ as the merging threshold. For dynamic threshold, we decrease the threshold by 0.05 each time for 10 times until at least one span is above the current threshold. We also use the development domain results to choose individual threshold for each target domain to filter invalid spans. We use grid search between [0.85, 0.97] with a step of 0.05 to search for the best threshold on the development domain. Our span-level evaluation is modified from conlleval script: https://www.clips.uantwerpen.be/conll2000/chunking/conlleval.txt.

On CLINC, we set the initial learning rate to be 5×10^{-5} and 1×10^{-5} for fine-tuning on the target domain. We set per-class batch size to be 8 for training on the source domain, and 5 and 2 for 5-shot and 2-shot fine-tuning.

A.2 SNIPS Data Details

Test Domain	Dev. Domain	Avg. $|S|$
GetWeather	PlayMusic	28.91
PlayMusic	AddToPlaylist	34.43
AddToPlaylist	RateBook	13.84
RateBook	FindScreeningEvent	19.83
FindScreeningEvent	BookRestaurant	19.27
BookRestaurant	SearchCreativeWork	41.58
SearchCreativeWork	GetWeather	5.28

Table 5: Corresponding development domain and average support set size for each testing domain.

A.3 CLINC Results on Different Data Constructions

	support_set=all			support_set=balance			support_set=tgt
	tgt	src	avg	tgt	src	avg	tgt
Initial BERT							
Proto$^{\tt frz}$	14.07	25.02	21.37	-	-	-	-
Retriever$^{\tt frz}$	8.24	54.76	39.25	22.09	25.29	24.22	37.93
Pre-train on src domain							
BERT fine-tune	-	96.51	-	-	-	-	-
Proto	75.02	95.73	88.83	-	-	-	-
Retriever	62.69	**97.08**	85.62	75.93	95.44	88.94	88.53
Retriever min-max	66.00	96.64	86.43	71.82	95.14	87.37	86.38
Fine-tune on tgt domain							
BERT fine-tune	78.89	43.91	55.57	-	-	-	-
Proto	80.44	95.57	90.53	-	-	-	90.35
Retriever	66.76	96.95	86.89	79.20	95.50	90.07	**91.16**
Retriever min-max	67.64	96.84	87.11	77.60	95.35	89.43	89.56
Fine-tune on tgt domain with src data							
BERT fine-tune	72.00	95.18	87.45	-	-	-	-
Proto	83.33	94.82	90.99	-	-	-	90.22
Retriever	69.51	97.04	87.86	**84.95**	95.41	**91.92**	90.78
Retriever min-max	71.35	96.96	88.42	81.00	94.55	90.03	89.82

Table 6: Intent accuracy on CLINC for $n_c = 10, n_i = 10$ with 5-shots.

	support_set=all			support_set=balance			support_set=tgt
	tgt	src	avg	tgt	src	avg	tgt
Initial BERT							
Proto$^{\tt frz}$	8.70	24.06	18.94	-	-	-	-
Retriever$^{\tt frz}$	3.91	54.96	37.94	17.38	16.83	17.01	27.96
Pre-train on src domain							
BERT fine-tune	-	96.51	-	-	-	-	-
Proto	76.40	95.70	89.27	-	-	-	-
Retriever	53.73	97.02	82.59	73.13	94.32	87.26	86.47
Retriever min-max	53.47	96.87	82.40	68.47	95.29	86.35	81.76
Fine-tune on tgt domain							
BERT fine-tune	75.57	50.91	59.13	-	-	-	-
Proto	76.36	95.06	**88.82**	-	-	-	86.67
Retriever	55.20	97.05	83.10	74.89	94.57	88.01	87.74
Retriever min-max	55.96	96.92	83.27	71.09	95.25	87.19	83.91
Fine-tune on tgt domain with src data							
BERT fine-tune	64.97	95.15	85.09	-	-	-	-
Proto	**77.02**	95.29	89.20	-	-	-	86.31
Retriever	56.36	**97.17**	83.56	76.87	94.11	88.36	**88.18**
Retriever min-max	58.82	96.85	84.17	74.31	94.32	87.56	83.98

Table 7: Intent accuracy on CLINC for $n_c = 10, n_i = 10$ with 2-shots.

	support_set=all			support_set=balance			support_set=tgt
	tgt	src	avg	tgt	src	avg	tgt
Initial BERT							
`Proto`[frz]	12.57	21.67	17.42	-	-	-	-
`Retriever`[frz]	9.46	56.96	34.79	23.37	25.87	24.70	32.72
Pre-train on src domain							
`BERT fine-tune`	-	96.92	-	-	-	-	-
`Proto`	73.17	94.96	84.79	-	-	-	-
`Retriever`	66.56	96.76	82.67	76.56	94.99	86.39	83.72
`Retriever min-max`	66.08	96.64	82.38	73.14	95.13	84.87	79.52
Fine-tune on tgt domain							
`BERT fine-tune`	76.92	44.17	59.45	-	-	-	-
`Proto`	77.53	94.88	86.78	-	-	-	85.21
`Retriever`	68.35	**97.11**	83.69	77.54	95.67	87.21	85.72
`Retriever min-max`	70.65	96.76	84.58	76.74	94.90	86.43	85.41
Fine-tune on tgt domain with src data							
`BERT fine-tune`	70.48	95.11	83.61	-	-	-	-
`Proto`	79.89	94.32	87.59	-	-	-	85.41
`Retriever`	71.46	96.88	85.02	**79.90**	94.61	**87.75**	**89.67**
`Retriever min-max`	71.73	96.86	85.13	78.43	95.31	87.43	85.32

Table 8: Intent accuracy on CLINC for $n_c = 8, n_i = 10$ with 5-shots.

	support_set=all			support_set=balance			support_set=tgt
	tgt	src	avg	tgt	src	avg	tgt
Initial BERT							
`Proto`[frz]	7.24	20.99	14.57	-	-	-	-
`Retriever`[frz]	4.76	57.24	32.75	17.64	18.29	17.99	24.17
Pre-train on src domain							
`BERT fine-tune`	-	96.92	-	-	-	-	-
`Proto`	70.59	94.42	83.30	-	-	-	-
`Retriever`	54.46	**97.22**	77.27	68.06	93.90	81.84	78.89
`Retriever min-max`	56.36	96.79	77.92	65.43	94.46	80.91	74.19
Fine-tune on tgt domain							
`BERT fine-tune`	68.94	56.41	62.25	-	-	-	-
`Proto`	73.19	94.70	84.66	-	-	-	79.83
`Retriever`	57.58	97.01	78.61	70.98	93.56	83.02	**80.35**
`Retriever min-max`	58.71	96.89	79.07	68.71	94.64	82.54	76.51
Fine-tune on tgt domain with src data							
`BERT fine-tune`	61.53	95.10	79.43	-	-	-	-
`Proto`	**73.57**	94.59	**84.78**	-	-	-	80.08
`Retriever`	59.06	97.07	79.33	72.33	93.11	83.42	80.14
`Retriever min-max`	60.49	96.89	79.91	70.92	94.68	83.59	79.21

Table 9: Intent accuracy on CLINC for $n_c = 8, n_i = 10$ with 2-shots.

	support_set=all			support_set=balance			support_set=tgt
	tgt	src	avg	tgt	src	avg	tgt
Initial BERT							
Proto$^{\text{frz}}$	15.02	17.58	16.30	-	-	-	-
Retriever$^{\text{frz}}$	10.65	55.97	33.31	21.16	25.85	23.51	26.71
Pre-train on src domain							
BERT fine-tune	-	96.87	-	-	-	-	-
Proto	67.05	96.37	81.71	-	-	-	-
Retriever	64.19	**97.18**	80.69	68.40	95.85	82.13	72.02
Retriever min-max	61.79	96.71	79.25	65.11	95.85	80.48	68.21
Fine-tune on tgt domain							
BERT fine-tune	69.32	64.76	67.04	-	-	-	-
Proto	73.63	96.09	84.86	-	-	-	77.45
Retriever	68.65	97.05	82.85	74.59	96.24	85.42	77.96
Retriever min-max	67.94	97.04	82.49	71.88	96.27	84.07	75.93
Fine-tune on tgt domain with src data							
BERT fine-tune	67.93	95.95	81.94	-	-	-	-
Proto	74.43	95.63	85.03	-	-	-	76.61
Retriever	71.51	96.91	84.21	**76.37**	95.81	**86.09**	**78.62**
Retriever min-max	70.50	96.92	83.71	73.76	95.79	84.78	76.65

Table 10: Intent accuracy on CLINC for $n_c = 5, n_i = 15$ with 5-shots.

	support_set=all			support_set=balance			support_set=tgt
	tgt	src	avg	tgt	src	avg	tgt
Initial BERT							
Proto$^{\text{frz}}$	9.39	16.19	12.79	-	-	-	-
Retriever$^{\text{frz}}$	5.18	56.25	30.72	14.99	18.36	16.68	19.63
Pre-train on src domain							
BERT fine-tune	-	96.87	-	-	-	-	-
Proto	63.50	95.72	79.61	-	-	-	-
Retriever	56.19	**97.14**	76.67	63.75	95.36	79.55	67.82
Retriever min-max	55.88	96.94	76.41	60.32	95.50	77.91	63.60
Fine-tune on tgt domain							
BERT fine-tune	61.91	77.11	69.51	-	-	-	-
Proto	66.38	95.72	81.05	-	-	-	70.40
Retriever	58.83	97.01	77.92	66.90	95.30	81.10	**71.26**
Retriever min-max	57.97	97.07	77.52	63.23	95.78	79.50	66.44
Fine-tune on tgt domain with src data							
BERT fine-tune	60.32	96.16	78.24	-	-	-	-
Proto	66.98	95.77	**81.38**	-	-	-	70.28
Retriever	59.88	96.96	78.42	**67.23**	94.85	81.04	70.52
Retriever min-max	60.08	97.08	78.58	65.45	95.73	80.59	68.06

Table 11: Intent accuracy on CLINC for $n_c = 5, n_i = 15$ with 2-shots.