Discriminative Nearest Neighbor Few-Shot Intent Detection by Transferring Natural Language Inference

Jian-Guo Zhang[1*] **Kazuma Hashimoto**[2†] **Wenhao Liu**[2] **Chien-Sheng Wu**[2]
Yao Wan[3] **Philip S. Yu**[1] **Richard Socher**[2] **Caiming Xiong**[2]
[1] University of Illinois at Chicago, Chicago, USA
[2]Salesforce Research, Palo Alto, USA
[3]Huazhong University of Science and Technology, Wuhan, China
{jzhan51,psyu}@uic.edu, wanyao@hust.edu.cn
{k.hashimoto,wu.jason,rsocher,cxiong}@salesforce.com

Abstract

Intent detection is one of the core components of goal-oriented dialog systems, and detecting out-of-scope (OOS) intents is also a practically important skill. Few-shot learning is attracting much attention to mitigate data scarcity, but OOS detection becomes even more challenging. In this paper, we present a simple yet effective approach, discriminative nearest neighbor classification with deep self-attention. Unlike softmax classifiers, we leverage BERT-style pairwise encoding to train a binary classifier that estimates the best matched training example for a user input. We propose to boost the discriminative ability by transferring a natural language inference (NLI) model. Our extensive experiments on a large-scale multi-domain intent detection task show that our method achieves more stable and accurate in-domain and OOS detection accuracy than RoBERTa-based classifiers and embedding-based nearest neighbor approaches. More notably, the NLI transfer enables our 10-shot model to perform competitively with 50-shot or even full-shot classifiers, while we can keep the inference time constant by leveraging a faster embedding retrieval model.

1 Introduction

Intent detection is one of the core components when building goal-oriented dialog systems. The goal is to achieve high intent classification accuracy, and another important skill is to accurately detect unconstrained user intents that are out-of-scope (OOS) in a system (Larson et al., 2019). A practical challenge is data scarcity because different systems define different sets of intents, and thus few-shot learning is attracting much attention. However, previous work has mainly focused on the few-shot intent classification without OOS (Luo et al., 2018; Casanueva et al., 2020).

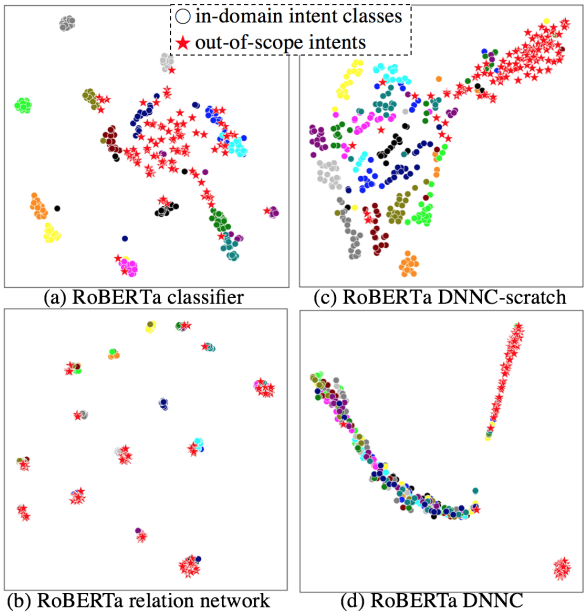

Figure 1: tSNE visual comparison for OOS detection between existing methods ((a) and (b)) and our proposed method ((c) and (d)). Their embeddings before their classifier layers are used (best viewed in color).

OOS detection can be considered as out-of-distribution detection (Hendrycks and Gimpel, 2017; DeVries and Taylor, 2018). Recent work has shown that large-scale pre-trained models like BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019) still struggle with out-of-distribution detection, despite their strong in-domain performance (Hendrycks et al., 2020). Figure 1 (a) shows how unseen input text is mapped into a feature space, by a RoBERTa-based model for 15-way 5-shot intent classification. The separation between OOS and some in-domain intents is not clear, which presumably hinders the model's OOS detection ability. This observation calls for investigation into more sample-efficient approaches to handling

[*]Work done while the first author was an intern at Salesforce Research.
[†]Corresponding author.

Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, pages 5064–5082,
November 16–20, 2020.

the in-domain and OOS examples accurately.

In this paper, we tackle the task from a different angle, and propose a discriminative nearest neighbor classification (DNNC) model. Instead of expecting the text encoders to be generalized enough to discriminate both the in-domain and OOS examples, we make full use of the limited training examples both in training and inference time as a nearest neighbor classification schema. We leverage the BERT-style paired text encoding with deep self-attention to directly model relations between pairs of user utterances. We then train a matching model as a pairwise binary classifier to estimate whether an input utterance belongs to the same class of a paired example. We expect this to free the model from having the OOS separation issue in Figure 1 (a) by avoiding explicit modeling of the intent classes. Unlike an embedding-based matching function as in relation networks (Sung et al., 2018) (Figure 1 (b)), the deep pairwise matching function produces clear separation between the in-domain and OOS examples (Figure 1 (c)). We further propose to seamlessly transfer a natural language inference (NLI) model to enhance this clear separation (Figure 1 (d)).

We verify our hypothesis by conducting extensive experiments on a large-scale multi-domain intent detection task with OOS (Larson et al., 2019) in various few-shot learning settings. Our experimental results show that, compared with RoBERTa classifiers and embedding nearest neighbor approaches, our DNNC attains more stable and accurate performance both in in-domain and OOS accuracy. Moreover, our 10-shot model can perform competitively with a 50-shot or even full-shot classifier, with the performance boost by the NLI transfer. We also show how to speedup our DNNC's inference time without sacrificing accuracy.

2 Background

2.1 Task: Few-Shot Intent Detection

Given a user utterance u at every turn in a goal-oriented dialog system, an intent detection model $I(u)$ aims at predicting the speaker's intent:

$$I(u) = c, \tag{1}$$

where c is one of pre-defined N intent classes $\mathbf{C} = \{C_1, C_2, \ldots, C_N\}$, or is categorized as OOS. The OOS category corresponds to user utterances whose requests are not covered by the system. In other words, any utterance can be OOS as long as it does *not* fall into any of the N intent classes, so the definition of OOS is different depending on $\mathbf{C}$.

Balanced K-shot learning In a few-shot learning scenario, we have a limited number of training examples for each class, and we assume that we have K examples for each of the N classes in our training data. In other words, we have $N \cdot K$ training examples in total. We denote the i-th training example from the j-th class C_j as $e_{j,i} \in E$, where E is the set of the examples. K is typically 5 or 10.

2.2 Multi-Class Classification

The goal is to achieve high accuracy both for the intent classification and OOS detection. One common approach to this task is using a multi-class classification model. Specifically, to get a strong baseline for the few-shot learning use case, one can leverage a pre-trained model as transfer learning, which has been shown to achieve state-of-the-art results on numerous natural language processing tasks. We use BERT (Devlin et al., 2019; Liu et al., 2019) as a text encoder:

$$h = \mathrm{BERT}(u) \in \mathbb{R}^d, \tag{2}$$

where h is a d-dimensional output vector corresponding to the special token `[CLS]` as in the following input format: `[[CLS]`, u, `[SEP]]`.[1]

To handle the intent classification and the OOS detection, we apply the threshold-based strategy in Larson et al. (2019), to the softmax output of the N-class classification model (Hendrycks and Gimpel, 2017):

$$p(c|u) = \mathrm{softmax}(Wh + b) \in \mathbb{R}^N, \tag{3}$$

where $W \in \mathbb{R}^{N \times d}$ and $b \in \mathbb{R}^N$ are the classifier's model parameters. The classification model is trained by cross-entropy loss with the ground-truth intent labels of the training examples. At inference time, we first take the class C_j with the largest value of $p(c = C_j|u)$, then output $I(u) = C_j$ if $p(c = C_j|u) > T$, where $T \in [0.0, 1.0]$ is a threshold to be tuned, and otherwise we output $I(u) = \mathrm{OOS}$.

2.3 Nearest Neighbor Classification

As the fundamental building block of our proposed method, we also review nearest neighbor classification (i.e., k-nearest neighbors (kNN) classification

[1]The format of these special tokens is different in RoBERTa, but we use the original BERT's notations.

	Input utterance	Utterance to be compared	Label
(a)	i need you to send 500 dollars from my high tier account to my regular checking	help me move my money please	pos.
(b)	i would like to know when the bill is due	i need to know the amounts due for my utilities and cable bills	neg.
(c)	And can you tell me any of the names and addresses? Annie considered.	Are you able to inform me of any name or address?	pos.
(d)	It's Sunday, what channel is this?	It's Sunday, can you change the channel?	neg.
(e)	I want to go back to Marguerite.	I never want to return to Marguerite.	neg.

Table 1: Training examples for our model. The first two examples ((a)–(b)) come from the CLINC150 dataset (Larson et al., 2019), and the other three examples ((c)–(e)) come from the MNLI dataset (Williams et al., 2018).

with $k = 1$), a simple and well-established concept for classification (Simard et al., 1993; Cunningham and Delany, 2007). The basic idea is to classify an input into the same class of the most relevant training example based on a certain metric.

In our task, we formulate a nearest neighbor classification model as the following:

$$I(u) = \text{class}\left(\arg\max_{e_{j,i} \in E} S(u, e_{j,i})\right), \quad (4)$$

where $\text{class}(e_{j,i})$ is a function returning the intent label (class) of the training example $e_{j,i}$, and S is a function that estimates some relevance score between u and $e_{j,i}$. To detect OOS, we can also use the uncertainty-based strategy in Section 2.2; that is, we take the output label from Equation (4) if the corresponding relevance score is greater than a threshold, and otherwise we output $I(u) = \text{OOS}$.

3 Proposed Method

This section first describes how to directly model inter-utterance relations in our nearest neighbor classification scenario. We then introduce a binary classification strategy by synthesizing pairwise examples, and propose a seamless transfer of NLI. Finally, we describe how to speedup our method's inference process.

3.1 Deep Pairwise Matching Function

The objective of $S(u, e_{j,i})$ in Equation (4) is to find the best matched utterance from the training set E, given the input utterance u. The typical methodology is to embed each data example into a vector space and (1) use an off-the-shelf distance metric to perform a similarity search (Cunningham and Delany, 2007) or (2) learn a distant metric between the embeddings (Sung et al., 2018). However, as shown in Figure 1, the text embedding methods do not discriminate the OOS examples well enough.

To model fine-grained relations of utterance pairs to distinguish in-domain and OOS intents, we propose to formulate $S(u, e_{j,i})$ as follows:

$$h = \text{BERT}(u, e_{j,i}) \in \mathbb{R}^d, \quad (5)$$
$$S(u, e_{j,i}) = \sigma(W \cdot h + b) \in \mathbb{R}, \quad (6)$$

where BERT is the same model in Equation (2), except that we follow a different input format to accommodate pairs of utterances: `[[CLS]]`, u, `[SEP]`, $e_{j,i}$, `[SEP]`]. σ is the sigmoid function, and $W \in \mathbb{R}^{1\times d}$ and $b \in \mathbb{R}$ are the model parameters. We can interpret our method as wrapping both the embedding and matching functions into the paired encoding with the deep self-attention in BERT (Equation (5)) along with the discriminative model (Equation (6)). It has been shown that the paired text encoding is crucial in capturing complex relations between queries and documents in document retrieval (Watanabe et al., 2017; Nie et al., 2019; Asai et al., 2020).

3.2 Discriminative Training

We train the matching model $S(u, e_{j,i})$ as a binary classifier, such that $S(u, e_{j,i})$ is closed to 1.0 if u belongs to the same class of $e_{j,i}$, and otherwise closed to 0.0. The model is trained by a binary cross-entropy loss function.

Positive examples To create positive examples, we consider all the possible ordered pairs within the same intent class: $(e_{j,i}, e_{j,\ell})$ such that $i \neq \ell$. We thus have $N \times K \times (K-1)$ positive examples in total. Table 1 (a) shows a positive example created from an intent, "transfer," in a banking domain.

Negative examples For negative examples, we consider all the possible ordered pairs across any two different intent classes: $(e_{j,i}, e_{o,\ell})$ such that $j \neq o$. We thus have $K^2 \times N \times (N-1)$ negative examples in total, and this number is in general greater than that of the positive examples. Table 1 (b) shows a negative example, where the input utterance comes from the intent, "bill due", and the paired sentence from another intent, "bill balance".

3.3 Seamless Transfer from NLI

A key characteristic of our method is that we seek to model the relations between the utterance pairs, instead of explicitly modeling the intent classes. To mitigate the data scarcity setting in few-shot learning, we consider transferring another inter-sentence-relation task.

This work focuses on NLI; the task is to identify whether a hypothesis sentence can be entailed by a premise sentence (Bowman and Zhu, 2019). We treat the NLI task as a binary classification task: `entailment` (positive) or `non-entailment` (negative).[2] We first pre-train our model with the NLI task, where the premise sentence corresponds to the u-position, and the hypothesis sentence corresponds to the $e_{j,i}$-position in Equation (5). Note that it is not necessary to modify the model architecture since the task format is consistent, and we can train the NLI model solely based on existing NLI datasets. Once the NLI model pre-training is completed, we fine-tune the NLI model with the intent classification training examples described in Section 3.2. This allows us to transfer the NLI model to any intent detection datasets seamlessly.

Why NLI? The NLI task has been actively studied, especially since the emergence of large scale datasets (Bowman et al., 2015; Williams et al., 2018), and we can directly leverage the progress. Moreover, recent work is investigating cross-lingual NLI (Eriguchi et al., 2018; Conneau et al., 2018), and this is encouraging to consider multilinguality in future work. On the other hand, while we can find examples relevant to the intent detection task, as shown in Table 1 ((c), (d), and (e)), we still need the few-shot fine-tuning. This is because a domain mismatch still exists in general, and perhaps more importantly, our intent detection approach is not exactly modeling NLI.

Why not other tasks? There are other tasks modeling relationships between sentences. Paraphrase (Wieting and Gimpel, 2018) and semantic relatedness (Marelli et al., 2014) tasks are such examples. It is possible to automatically create large-scale paraphrase datasets by machine translation (Ganitkevitch et al., 2013). However, our task is not a paraphrasing task, and creating negative examples is crucial and non-trivial (Chambers and Jurafsky, 2010). In contrast, as described above, the NLI setting comes with negative examples by nature. The semantic relatedness (or textual similarity) task is considered as a coarse-grained task compared to NLI, as discussed in the previous work (Hashimoto et al., 2017), in that the task measures semantic or topical relatedness. This is not ideal for the intent detection task, because we need to discriminate between topically similar utterances of different intents. In summary, the NLI task well matches our objective, with access to the large datasets.

[2] A widely-used format is a three-way classification task with `entailment`, `neutral`, and `contradiction`, but we merge the latter two classes into a single `non-entailment` class.

3.4 A Joint Approach with Fast Retrieval

The number of model parameters of the multi-class classification model in Section 2.2 and our model in Section 3 is almost the same when we use the same pre-trained models. However, our example-based method has an inference-time bottleneck in Equation (5), where we need to compute the BERT encoding for all $N \times K$ $(u, e_{j,i})$ pairs.

We follow common practice in document retrieval to reduce the inference-time bottleneck (Nie et al., 2019; Asai et al., 2020), by introducing a fast text retrieval model to select a set of top-k examples E_k from the training set E, based on its retrieval scores. We then replace E in Equation (4) with the shrunk set E_k. The cost of the paired BERT encoding is now constant, regardless the size of E. Either TF-IDF (Chen et al., 2017) or embedding-based retrieval (Johnson et al., 2017; Seo et al., 2019; Lee et al., 2019) can be used for the first step. We use the following fast kNN.

Faster kNN As a baseline and a way to instantiate our joint approach, we use Sentence-BERT (SBERT) (Reimers and Gurevych, 2019) to separately encode u and $e_{j,i}$ ($x \in \{u, e_{j,i}\}$) as follows:

$$v(x) \quad = \quad \text{SBERT}(x) \in \mathbb{R}^d, \qquad (7)$$

where the input text format is identical to that of BERT in Equation (2). SBERT is a BERT-based text embedding model, fine-tuned by siamese networks with NLI datasets. Thus both our method and SBERT transfer the NLI task in different ways.

Cosine similarity between $v(u)$ and $v(e_{j,i})$ then replaces $S(u, e_{j,i})$ in Equation (6). To get a fair comparison, instead of using the encoding vectors produced by the original SBERT, we fine-tune SBERT with our intent training examples described in Section 3.2. The cosine similarity is symmetric, so we have half the training examples. We use the

	N	Train	Dev.	Test
All domains	150	15,000	3,000	4,500
Single domain	15	1,500	300	450
OOS	-	n/a	100	1,000

Table 2: Dataset statistics. The number of the examples is equally distributed across the intent classes.

pairwise cosine-based loss function in Reimers and Gurevych (2019). After the model training, we pre-compute $v(e_{j,i})$ for fast retrieval.

4 Experimental Settings

4.1 Dataset: Multi-Domain Intent Detection

We use a recently-released dataset, CLINC150,[3] for multi-domain intent detection (Larson et al., 2019). The CLINC150 dataset defines 150 types of intents in total (i.e., $N = 150$), where there are 10 different domains and 15 intents for each of them. Table 2 shows the dataset statistics.

OOS evaluation examples The dataset also provides OOS examples whose intents do not belong to any of the 150 intents. From the viewpoint of out-of-distribution detection (Hendrycks and Gimpel, 2017; Hendrycks et al., 2020), we do not use the OOS examples during the training stage; we only use the evaluation splits as in Table 2.

Single-domain experiments The task in the CLINC150 dataset is like handling many different services in a single system; that is, topically different intents are mixed (e.g., "alarm" in the "Utility" domain, and "pay bill" in the "Banking" domain). In contrast, it is also a reasonable setting to handle each domain (or service) separately as in Rastogi et al. (2019). In addition to the all-domain experiment, we conduct single-domain experiments, where we only focus on a specific domain with its 15 intents (i.e., $N = 15$). More specifically, we use four domains, "Banking," "Credit cards," "Work," and "Travel," among the ten domains. Note that the same OOS evaluation sets are used.

4.2 Evaluation Metrics

We follow Larson et al. (2019) to report in-domain accuracy, $\mathrm{Acc}_{\mathrm{in}}$, and OOS recall, R_{oos}. These two metrics are defined as follows:

$$\mathrm{Acc}_{\mathrm{in}} = C_{\mathrm{in}}/N_{\mathrm{in}}, \quad R_{\mathrm{oos}} = C_{\mathrm{oos}}/N_{\mathrm{oos}}, \tag{8}$$

where C_{in} is the number of correctly predicted in-domain intent examples, and N_{in} is the total number of the examples evaluated. This is analogous to the calculation of the OOS recall.

Threshold selection We use the uncertainty-based OOS detection, and therefore we need a way to set the threshold T. For each T in $[0.0, 0.1, \ldots, 0.9, 1.0]$, we calculate a joint score $J_{\mathrm{in\_oos}}$ defined as follows:

$$J_{\mathrm{in\_oos}} = \mathrm{Acc}_{\mathrm{in}} + R_{\mathrm{oos}}, \tag{9}$$

and select a threshold value to maximize the score on the development set. There is a trade-off to be noted; the larger the value of T is, the higher R_{oos} (and the lower $\mathrm{Acc}_{\mathrm{in}}$) we expect, because the models predict OOS more frequently.

Notes on $J_{\mathrm{in\_oos}}$ Our joint score $J_{\mathrm{in\_oos}}$ in Equation (9) gives the same weight to the two metrics, $\mathrm{Acc}_{\mathrm{in}}$ and R_{oos}, compared to other combined metrics. For example, Larson et al. (2019) and Wu et al. (2020) used a joint accuracy score:

$$\frac{C_{\mathrm{in}} + C_{\mathrm{oos}}}{N_{\mathrm{in}} + N_{\mathrm{oos}}} = \frac{\mathrm{Acc}_{\mathrm{in}} + r R_{\mathrm{oos}}}{1 + r}, \tag{10}$$

where $r = N_{\mathrm{oos}}/N_{\mathrm{in}}$ depends on the balance between N_{in} and N_{oos}, and thus this combined metric can put much more weight on the in-domain accuracy when $N_{\mathrm{in}} \gg N_{\mathrm{oos}}$. Table 2 shows $r = 100/3000$ $(= 0.0333)$ in the development set of the "all domains" setting, which underestimates the importance of R_{oos}. Actually, the OOS recall scores in Larson et al. (2019) and Wu et al. (2020) are much lower than those with our RoBERTa classifier, and the trade-off with respect to the tuning process was not discussed.[4]

We also report OOS precision and OOS F1 for more comprehensive evaluation:

$$P_{\mathrm{oos}} = C_{\mathrm{oos}}/N'_{\mathrm{oos}}, \quad F_1 = H(P_{\mathrm{oos}}, R_{\mathrm{oos}}), \tag{11}$$

where N'_{oos} is the number of examples predicted as OOS, and $H(\cdot, \cdot)$ is the harmonic mean.

4.3 Model Training and Configurations

We use RoBERTa (the `base` configuration with $d = 768$) as a BERT encoder for all the BERT/SBERT-based models in our experiments,[5]

[3] https://github.com/clinc/oos-eval.

[4] When we refer to our RoBERTa classifier's scores $(A_{\mathrm{in}}, R_{\mathrm{oos}}) = (92.9 \pm 0.2,\ 90.2 \pm 0.5)$ in Table 4, their corresponding scores are $(96.2,\ 52.3)$ and $(96.1,\ 46.3)$ in Larson et al. (2019) and Wu et al. (2020), respectively.

[5] We use https://github.com/huggingface/transformers and https://github.com/UKPLab/sentence-transformers.

because RoBERTa performed significantly better and more stably than the original BERT in our few-shot experiments. We combine three NLI datasets, SNLI (Bowman et al., 2015), MNLI (Williams et al., 2018), and WNLI (Levesque et al., 2011) from the GLUE benchmark (Wang et al., 2018) to pre-train our proposed model.

We apply label smoothing (Szegedy et al., 2016) to all the cross-entropy loss functions, which has been shown to improve the reliability of the model confidence (Müller et al., 2019). Experiments were conducted on single NVIDIA Tesla V100 GPU with 16GB memory.

Sampling training examples We conduct our experiments with $K = 5, 10$ following the task definition in Section 2.1. We randomly sample K examples from the entire training sets in Table 2, for each in-domain intent class 10 times unless otherwise stated. We train a model with a consistent hyper-parameter setting across the 10 different runs and follow the threshold selection process based on a mean score for each threshold. We also report a standard deviation for each result.

Using the development set We would not always have access to a large enough development set in the few-shot learning scenario. However, we still use the development set provided by the dataset to investigate the models' behaviors when changing hyper-parameters like the threshold.

Models compared We list the models used in our experiments:

- **Classifier baselines:** "Classifier" is the RoBERTa-based classification model described in Section 2.2. We further seek solid baselines by data augmentation. "Classifier-EDA" is the classifier trained with data augmentation techniques in Wei and Zou (2019). "Classifier-BT" is the classifier trained with back-translation data augmentation (Yu et al., 2018; Shleifer, 2019) by using a transformer-based English↔German translation system (Vaswani et al., 2017).

- **Non-BERT classifier:** We also test a state-of-the-art fast embedding-based classifier, "USE+ConveRT" (Henderson et al., 2019; Casanueva et al., 2020), in the "all domains" setting. Casanueva et al. (2020) showed that the "USE+ConveRT" outperformed a BERT classifier on the CLINC150 dataset, while it was not evaluated along with the OOS detection task. We modified their original code[6] to apply the uncertainty-based OOS detection.

- **kNN baselines:**[7] "Emb-kNN" is the kNN method with S(Ro)BERT(a) described in Section 3.4, and "Emb-kNN-vanilla" is *without* using our intent training examples for fine-tuning. "TF-IDF-kNN" is another kNN baseline using TF-IDF vectors, which tells us how well string matching performs on our task. We also implement a relation network (Sung et al., 2018), "RN-kNN," to learn a similarity metric between the SRoBERTa embeddings, instead of using the cosine similarity.

- **Proposed method:**[8] "DNNC" is our proposed method, and "DNNC-scratch" is *without* the NLI pre-training in Section 3.3. "DNNC-joint" is our joint approach on top of top-k retrieval by Emb-kNN (Section 3.4).

More details about the model training and the data augmentation configurations are described in Appendix A and Appendix B, respectively.

5 Experimental Results

This section shows our experimental results. Appendix C shows some additional figures.

5.1 Model Performance CLINC150 Dataset

Single domains We first show test set results of 5-shot and 10-shot in-domain classification and OOS detection accuracy in Table 3 for the four selected domains. In the 5-shot setting, the proposed DNNC method consistently attains the best results across all the four domains. The comparison between DNNC-scratch and DNNC shows that our NLI task transfer is effective. In the 10-shot setting, all the approaches generally experience an accuracy improvement due to the additional training data, and the dominant performance of DNNC weakens, although it remains highly competitive. We can see that our DNNC is comparable with or even surpasses some of the 50-shot classifier's scores, and the data augmentation techniques are not always helpful when we use the strong pre-trained model.

[6] https://github.com/connorbrinton/polyai-models/releases/tag/v1.0.

[7] We tried weighted voting in Cunningham and Delany (2007), but $k = 1$ performed better in general.

[8] Our code is available at https://github.com/salesforce/DNNC-few-shot-intent.

5-shot	In-domain accuracy		OOS recall		OOS precision		OOS F1	
	Banking	Credit cards	Banking	Credit cards	Banking	Credit cards	Banking	Credit cards
classifier	79.7 ± 1.8	79.9 ± 3.7	93.3 ± 2.9	93.3 ± 3.0	92.5 ± 0.9	93.6 ± 1.3	92.9 ± 1.8	93.4 ± 1.6
classifier-EDA	79.9 ± 3.0	73.1 ± 5.0	91.0 ± 2.6	94.1 ± 2.6	92.6 ± 1.5	90.0 ± 1.7	91.8 ± 1.9	92.0 ± 1.3
classifier-BT	77.2 ± 2.8	70.3 ± 4.4	91.5 ± 2.4	91.1 ± 1.8	91.4 ± 1.2	89.5 ± 1.8	91.4 ± 1.4	90.3 ± 1.7
DNNC-scratch	83.4 ± 1.9	84.6 ± 3.2	93.2 ± 2.7	96.4 ± 1.0	96.2 ± 0.9	95.8 ± 1.1	94.7 ± 1.2	96.1 ± 0.9
DNNC	**88.6 ± 1.3**	**90.5 ± 2.9**	**94.7 ± 1.0**	**97.7 ± 1.1**	**97.0 ± 0.3**	**97.8 ± 0.5**	**95.9 ± 0.5**	**97.8 ± 0.6**
classifier (50-shot)	90.0 ± 1.4	90.3 ± 1.4	93.3 ± 1.1	93.9 ± 1.2	96.3 ± 0.6	96.4 ± 0.6	94.8 ± 0.7	95.1 ± 0.6
5-shot	**Work**	**Travel**	**Work**	**Travel**	**Work**	**Travel**	**Work**	**Travel**
classifier	84.4 ± 1.9	87.8 ± 2.5	94.8 ± 1.6	96.1 ± 0.9	95.4 ± 0.7	94.7 ± 1.0	95.1 ± 0.9	95.4 ± 0.7
classifier-EDA	80.5 ± 3.3	88.2 ± 2.4	95.2 ± 1.6	94.6 ± 2.1	92.8 ± 1.3	94.9 ± 1.0	94.0 ± 0.9	94.7 ± 0.9
classifier-BT	80.3 ± 3.4	90.8 ± 2.4	94.4 ± 1.1	92.3 ± 1.9	93.1 ± 1.0	95.9 ± 1.1	93.8 ± 0.8	94.1 ± 1.3
DNNC-scratch	83.2 ± 2.1	87.8 ± 3.5	96.3 ± 1.7	94.9 ± 2.8	96.7 ± 0.9	94.9 ± 1.4	96.5 ± 0.6	94.9 ± 0.9
DNNC	**89.9 ± 3.2**	**91.8 ± 1.6**	**96.7 ± 1.1**	**96.4 ± 1.2**	**97.9 ± 1.1**	**96.5 ± 0.7**	**97.3 ± 0.5**	**96.5 ± 0.7**
classifier (50-shot)	94.3 ± 0.8	97.0 ± 0.3	95.2 ± 0.8	92.6 ± 1.4	97.7 ± 0.3	98.6 ± 0.2	96.5 ± 0.5	95.5 ± 0.8

10-shot	In-domain accuracy		OOS recall		OOS precision		OOS F1	
	Banking	Credit cards	Banking	Credit cards	Banking	Credit cards	Banking	Credit cards
classifier	85.2 ± 1.3	83.7 ± 2.1	93.3 ± 1.0	93.8 ± 1.4	94.4 ± 0.5	93.9 ± 0.9	93.8 ± 0.6	93.8 ± 0.7
classifier-EDA	82.5 ± 1.3	79.3 ± 1.7	**95.9 ± 0.8**	96.8 ± 0.8	93.3 ± 0.4	92.3 ± 0.8	94.6 ± 0.5	94.5 ± 0.3
classifier-BT	82.2 ± 1.9	82.9 ± 1.8	94.9 ± 1.9	89.3 ± 2.0	93.1 ± 0.8	94.3 ± 0.6	94.0 ± 0.9	91.7 ± 1.2
DNNC-scratch	89.6 ± 1.6	92.1 ± 1.1	92.1 ± 3.1	94.8 ± 1.2	97.5 ± 0.9	**98.1 ± 0.4**	94.7 ± 1.5	96.4 ± 0.6
DNNC	**91.2 ± 1.1**	**92.1 ± 1.0**	94.8 ± 1.1	**97.8 ± 0.8**	**97.5 ± 0.4**	97.8 ± 0.3	**96.1 ± 0.6**	**97.8 ± 0.4**
classifier (50-shot)	90.0 ± 1.4	90.3 ± 1.4	93.3 ± 1.1	93.9 ± 1.2	96.3 ± 0.6	96.4 ± 0.6	94.8 ± 0.7	95.1 ± 0.6
10-shot	**Work**	**Travel**	**Work**	**Travel**	**Work**	**Travel**	**Work**	**Travel**
classifier	86.0 ± 2.2	93.0 ± 1.2	97.2 ± 0.7	94.8 ± 0.9	94.5 ± 0.8	96.8 ± 0.6	95.8 ± 0.3	95.8 ± 0.4
classifier-EDA	86.4 ± 1.7	93.0 ± 1.0	97.0 ± 0.9	95.2 ± 1.3	94.7 ± 0.7	96.9 ± 0.5	95.8 ± 0.5	96.0 ± 0.6
classifier-BT	83.7 ± 1.7	91.9 ± 1.1	**97.4 ± 0.9**	**96.1 ± 0.8**	93.6 ± 0.7	96.4 ± 0.5	95.5 ± 0.6	**96.2 ± 0.3**
DNNC-scratch	92.6 ± 1.7	**96.4 ± 0.6**	94.1 ± 1.4	84.8 ± 3.0	98.4 ± 0.6	**98.5 ± 0.3**	96.2 ± 0.8	91.1 ± 1.8
DNNC	**95.0 ± 1.0**	96.0 ± 0.8	95.5 ± 0.9	93.3 ± 1.9	**99.0 ± 0.4**	98.3 ± 0.4	**97.2 ± 0.5**	95.7 ± 1.0
classifier (50-shot)	94.3 ± 0.8	97.0 ± 0.3	95.2 ± 0.8	92.6 ± 1.4	97.7 ± 0.3	98.6 ± 0.2	96.5 ± 0.5	95.5 ± 0.8

Table 3: Testing results on the four different domains.

	In-domain accuracy		OOS recall		OOS precision		OOS F1	
	5-shot	10-shot	5-shot	10-shot	5-shot	10-shot	5-shot	10-shot
classifier	77.7 ± 0.6	83.2 ± 0.6	**91.9 ± 1.1**	**92.9 ± 0.8**	53.9 ± 0.8	58.2 ± 1.0	68.0 ± 1.0	71.6 ± 0.8
USE+ConveRT	76.9 ± 0.7	82.6 ± 0.5	90.7 ± 0.8	89.1 ± 0.4	49.0 ± 0.8	55.1 ± 0.7	63.6 ± 0.7	68.1 ± 0.6
DNNC	**84.9 ± 0.8**	**91.6 ± 0.3**	90.1 ± 1.6	83.0 ± 2.0	**64.7 ± 2.5**	**81.4 ± 2.0**	**75.3 ± 2.0**	**82.1 ± 0.3**
classifier (full-shot)	92.9 ± 0.2		90.2 ± 0.5		76.7 ± 0.7		82.9 ± 0.6	
USE+ConveRT (full-shot)	95.0 ± 0.1		64.6 ± 0.6		79.3 ± 0.7		71.2 ± 0.5	

Table 4: Testing results on the whole dataset (5 runs).

Entire CLINC150 dataset Next, Table 4 shows results to compare our method with the classifier and USE+ConveRT baselines, on the entire CLINC150 dataset with the 150 intents. USE+ConveRT performs worse than the RoBERTa-based classifier on the OOD detection task. The advantage of DNNC for in-domain intent detection is clear, with its 10-shot in-domain accuracy close to the upper-bound accuracy for the classifier baseline. One observation is that our DNNC method tends to be more confident about its prediction, with the increasing number of the training examples; as a result, the OOS recall becomes lower in the 10-shot setting, while the OOS precision is much higher than the other baselines. Better controlling the confidence output of the model is an interesting direction for future work.

When the USE+ConveRT baseline is evaluated along with the OOS detection task, its overall accuracy is not as good as the other RoBERTa-based models, despite its potential in the purely in-domain classification. This indicates that the fine-tuned (Ro)BERT(a) models are more robust to out-of-distribution examples than shallower models like USE+ConveRT, also suggested in Hendrycks et al. (2020).

5.2 Robustness of DNNC

As described in Section 4.2, we select the threshold to determine OOS by making a trade-off between in-domain classification and OOS detection accuracy. It is therefore desirable to have a model with candidate thresholds that provide high in-domain accuracy as well as OOS precision and recall.

We observe in Figure 3 that in the 5-shot setting, DNNC is the most robust to the threshold selection. The contrast between the classification model and DNNC-scratch suggests that nearest neighbor approaches (in this case DNNC) make for stronger discriminators; the advantage of DNNC over DNNC-scratch further demonstrates the power of the NLI transfer and, perhaps more importantly, the effectiveness of the pairwise discriminative pre-training. This result is consistent with the intuition we gained from Figure 1, and the overall observation is also consistent across different settings.

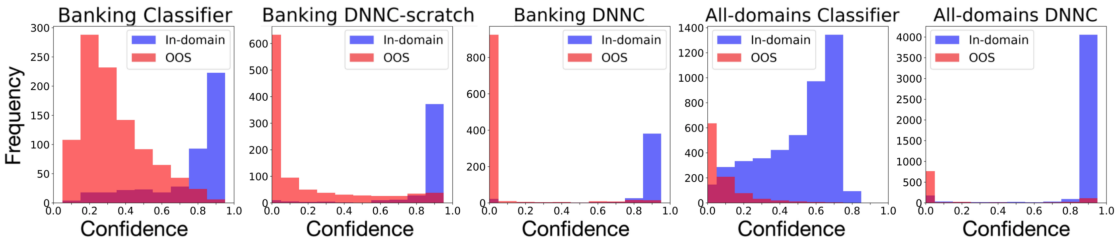

Figure 2: Model confidence level on 5-shot test sets for the banking domain and all domains.

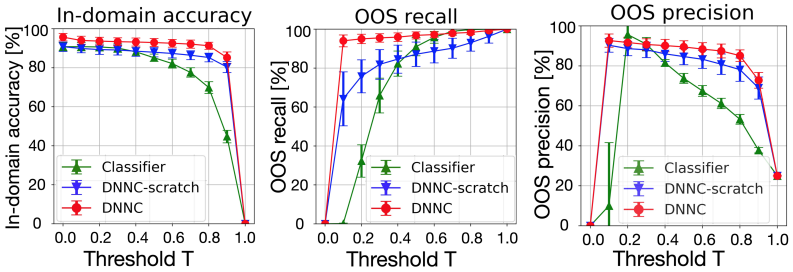

Figure 3: Development set results on the banking domain in the 5-shot setting. In this series of plots, a model with a higher area-under-the-curve is more robust.

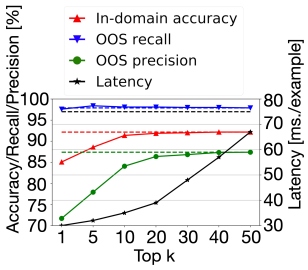

Figure 4: Accuracy vs. latency of DNNC-joint.

Banking	In-domain accuracy 5-shot	In-domain accuracy 10-shot	OOS recall 5-shot	OOS recall 10-shot	OOS precision 5-shot	OOS precision 10-shot	OOS F1 5-shot	OOS F1 10-shot	Latency [ms./example] 5-shot	Latency [ms./example] 10-shot
TF-IDF-kNN	59.2 ± 3.6	65.9 ± 3.8	61.0 ± 2.8	51.3 ± 1.6	97.6 ± 0.6	98.2 ± 0.3	75.1 ± 2.2	67.4 ± 1.3	1	1
Emb-kNN-vanilla	64.4 ± 2.8	65.4 ± 2.3	89.5 ± 1.1	94.6 ± 0.5	95.1 ± 0.5	92.6 ± 0.7	92.2 ± 0.5	93.6 ± 0.5	15	15
Emb-kNN	78.4 ± 2.4	84.3 ± 1.2	92.0 ± 2.0	91.6 ± 1.2	91.4 ± 0.8	93.7 ± 0.5	91.7 ± 0.7	92.6 ± 0.7	15	15
RN-kNN	79.5 ± 3.1	89.0 ± 1.4	88.3 ± 1.9	75.9 ± 4.0	91.6 ± 1.3	95.9 ± 0.6	89.9 ± 1.0	84.7 ± 2.5	17	17
DNNC-joint	88.5 ± 1.2	91.0 ± 1.0	95.0 ± 0.9	95.2 ± 1.1	96.9 ± 0.4	97.3 ± 0.4	96.0 ± 0.5	96.3 ± 0.6	36	36
DNNC	88.6 ± 1.3	91.2 ± 1.1	94.7 ± 1.0	94.8 ± 1.1	97.0 ± 0.3	97.5 ± 0.4	95.9 ± 0.5	96.1 ± 0.6	73	143
All domains	**5-shot**	**10-shot**	**5-shot**	**10-shot**	**5-shot**	**10-shot**	**5-shot**	**10-shot**	**5-shot**	**10-shot**
TF-IDF-kNN	35.4 ± 0.7	31.3 ± 1.1	72.2 ± 2.1	83.1 ± 1.5	24.7 ± 0.4	23.5 ± 0.4	36.8 ± 0.6	36.6 ± 0.6	1	2
Emb-kNN-vanilla	54.2 ± 0.5	50.7 ± 0.9	87.5 ± 0.7	95.1 ± 0.3	39.6 ± 0.3	34.2 ± 0.3	54.5 ± 0.4	50.3 ± 0.3	16	19
Emb-kNN	78.7 ± 1.0	86.9 ± 0.3	86.8 ± 2.4	82.3 ± 0.8	55.0 ± 1.5	66.5 ± 1.0	67.3 ± 1.6	73.5 ± 0.7	16	19
RN-kNN	76.6 ± 0.6	88.7 ± 1.1	88.9 ± 1.4	76.6 ± 2.7	50.0 ± 0.3	71.0 ± 0.5	64.0 ± 0.2	73.7 ± 1.0	16	18
DNNC-joint	84.5 ± 0.8	91.2 ± 0.2	90.6 ± 1.6	85.1 ± 1.8	63.6 ± 2.4	78.4 ± 2.1	74.7 ± 1.9	81.6 ± 0.4	37	41
DNNC	84.9 ± 0.8	91.6 ± 0.3	90.1 ± 1.6	83.0 ± 2.0	64.7 ± 2.5	81.4 ± 2.0	75.3 ± 2.0	82.1 ± 0.3	697	1498

Table 5: Comparison among the nearest neighbor methods on the test sets for the banking domain and all the domains. The latency is measured on a single NVIDIA Tesla V100 GPU, where the batch size is 1 to simulate an online use case. DNNC-joint is based on top-20 Emb-kNN retrieval.

To further understand the differences in behaviors between the classification model and DNNC method, we examine the output from the final softmax/sigmoid function (model confidence score) in Figure 2. At 5-shot, the classifier method still struggles to fully distinguish the in-domain examples from the OOS examples in its confidence scoring, while DNNC already attains a clear distinction between the two. Again, we can clearly see the effectiveness of the NLI transfer.

With the model architectures for BERT-based classifier and DNNC being the same (RoBERTa is used for both methods) except for the final layer (multi-class-softmax vs. binary sigmoid), this result suggests that the pairwise NLI-like training is more sample-efficient, making it an excellent candidate for the few-shot use case.

5.3 DNNC-joint for Faster Inference

Despite its effectiveness in few-shot intent and OOS settings, the proposed DNNC method might not scale in high-traffic use cases, especially when the number of classes, N, is large, due to the inference-time bottleneck (Section 3.4). With this in mind, we proposed the DNNC-joint approach, wherein a faster model is used to filter candidates for the fine-tuned DNNC model.

We compare the accuracy and inference latency metrics for various methods in Table 5. Note that Emb-kNN and RN-kNN exhibit excellent latency performance, but they fall considerably short in both the in-domain intent and OOS detection accuracy, compared to DNNC and the DNNC-joint methods. On the other hand, the DNNC-joint model shows competitiveness in both inference latency and accuracy. These results indicate that the

current text embedding approaches like SBERT are not enough to fully capture fine-grained semantics.

Intuitively, there is a trade-off between latency and inference accuracy: with aggressive filtering, the DNNC inference step needs to handle a smaller number of training examples, but might miss informative examples; with less aggressive filtering, the NLI model sees more training examples during inference, but will take longer to process single user input. This is illustrated in Figure 4, where the in-domain intent and OOS accuracy metrics (on the development set of the banking domain in the 5-shot setting) improve with the increase of k, while the latency increases at the same time. Empirically, $k = 20$ appears to strike the balance between latency and accuracy, with the accuracy metrics similar to those of the DNNC method, while being much faster than DNNC (dashed lines are the corresponding DNNC references).

6 Discussions and Related Work

Interpretability Interpretability is an important line of research recently (Jiang et al., 2019; Sydorova et al., 2019; Asai et al., 2020). The nearest neighbor approach (Simard et al., 1993) is appealing in that we can explicitly know which training example triggers each prediction. Table 11 in Appendix C shows some examples.

Call for better embeddings Emb-kNN and RN-kNN are not as competitive as DNNC. This encourages future work on the task-oriented evaluation of text embeddings in kNN.

Training time Our DNNC method needs longer training time than that of the classifier (e.g., 90 vs. 40 seconds to train a single-domain model), because we synthesize the pairwise examples. As a first step, we used all the training examples to investigate the effectiveness, but it is an interesting direction to seek more efficient pairwise training.

Distilled model Another way to speedup our model is to use distilled pre-trained models (Sanh et al., 2019). We replaced the RoBERTa model with a distilled RoBERTa model, and observed large variances with significantly lower OOS accuracy. Hendrycks et al. (2020) also suggested that the distilled models would not be robust to out-of-distribution examples.

Few-shot text classification Few-shot classification (Fei-Fei et al., 2006; Vinyals et al., 2016b) has been applied to text classification tasks (Deng et al., 2019; Geng et al., 2019; Xu et al., 2019), and few-shot intent detection is also studied but without OOS (Luo et al., 2018; Casanueva et al., 2020). There are two common scenarios: 1) learning with plenty of examples and then generalizing to unseen classes with a few examples, and 2) learning with a few examples for all seen classes. Meta-learning (Finn et al., 2017; Geng et al., 2019) is widely studied in the first scenario. In our paper, we have focused on the second scenario, assuming that there are only a limited number of training examples for each class. Our work is related to metric-based approaches such as matching networks (Vinyals et al., 2016a), prototypical networks (Snell et al., 2017) and relation networks (Sung et al., 2018), as they model nearest neighbours in an example-embedding or a class-embedding space. We showed that a relation network with the RoBERTa embeddings does not perform comparably to our method. We also considered several ideas from prototypical networks (Sun et al., 2019), but those did not outperform our Emb-kNN baseline. These results indicate that deep self-attention is the key to the nearest neighbor approach with OOS detection.

7 Conclusion

In this paper, we have presented a simple yet efficient nearest-neighbor classification model to detect user intents and OOS intents. It includes paired encoding and discriminative training to model relations between the input and example utterances. Moreover, a seamless transfer from NLI and a joint approach with fast retrieval are designed to improve the performance in terms of the accuracy and inference speed. Experimental results show superior performance of our method on a large-scale multi-domain intent detection dataset with OOS. Future work includes its cross-lingual transfer and cross-dataset (or cross-task) generalization.

Acknowledgments

This work is supported in part by NSF under grants III-1763325, III-1909323, and SaTC-1930941. We thank Huan Wang, Wenpeng Yin for their insightful discussions, and the anonymous reviewers for their helpful and thoughtful comments. We also thank Jin Qu, Tian Xie, Xinyi Yang, and Yingbo Zhou for their support in the deployment of DNNC into the internal system.

References

Akari Asai, Kazuma Hashimoto, Hannaneh Hajishirzi, Richard Socher, and Caiming Xiong. 2020. Learning to Retrieve Reasoning Paths over Wikipedia Graph for Question Answering. In *8th International Conference on Learning Representations (ICLR)*.

Samuel Bowman and Xiaodan Zhu. 2019. Deep Learning for Natural Language Inference. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorials*, pages 6–8.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642.

Iñigo Casanueva, Tadas Temčinas, Daniela Gerz, Matthew Henderson, and Ivan Vulić. 2020. Efficient Intent Detection with Dual Sentence Encoders. *arXiv preprint arXiv:2003.04807*.

Nathanael Chambers and Daniel Jurafsky. 2010. Improving the Use of Pseudo-Words for Evaluating Selectional Preferences. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 445–453.

Rajen Chatterjee, Christian Federmann, Matteo Negri, and Marco Turchi. 2019. Findings of the WMT 2019 Shared Task on Automatic Post-Editing. In *Proceedings of the Fourth Conference on Machine Translation (Volume 3: Shared Task Papers, Day 2)*, pages 11–28.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to Answer Open-Domain Questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879.

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel Bowman, Holger Schwenk, and Veselin Stoyanov. 2018. XNLI: Evaluating Cross-lingual Sentence Representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2475–2485.

Pádraig Cunningham and Sarah Jane Delany. 2007. k-Nearest Neighbour Classifiers. Technical Report UCD-CSI-2007-04, School of Computer Science & Informatics, University College Dublin.

Shumin Deng, Ningyu Zhang, Zhanlin Sun, Jiaoyan Chen, and Huajun Chen. 2019. When low resource nlp meets unsupervised language model: Meta-pretraining then meta-learning for few-shot text classification. *arXiv*, pages arXiv–1908.

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.

Terrance DeVries and Graham W. Taylor. 2018. Learning Confidence for Out-of-Distribution Detection in Neural Networks. *arXiv preprint arXiv:1802.04865*.

Akiko Eriguchi, Melvin Johnson, Orhan Firat, Hideto Kazawa, and Wolfgang Macherey. 2018. Zero-Shot Cross-lingual Classification Using Multilingual Neural Machine Translation. *arXiv preprint arXiv:1809.04686*.

Li Fei-Fei, Rob Fergus, and Pietro Perona. 2006. One-shot learning of object categories. *IEEE transactions on pattern analysis and machine intelligence*, 28(4):594–611.

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-Volume 70*, pages 1126–1135. JMLR. org.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The Paraphrase Database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 758–764.

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 3895–3904.

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2017. A Joint Many-Task Model: Growing a Neural Network for Multiple NLP Tasks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1923–1933.

Matthew Henderson, Iñigo Casanueva, Nikola Mrkšić, Pei-Hao Su, Tsung-Hsien Wen, and Ivan Vulić. 2019. ConveRT: Efficient and Accurate Conversational Representations from Transformers. *arXiv preprint arXiv:1911.03688*.

Dan Hendrycks and Kevin Gimpel. 2017. A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks. In *5th International Conference on Learning Representations (ICLR)*.

Dan Hendrycks, Xiaoyuan Liu, Eric Wallace, Adam Dziedzic, Rishabh Krishnan, and Dawn Song.

2020. Pretrained Transformers Improve Out-of-Distribution Robustness. *arXiv preprint arXiv:2004.06100*.

Yichen Jiang, Nitish Joshi, Yen-Chun Chen, and Mohit Bansal. 2019. Explore, Propose, and Assemble: An Interpretable Model for Multi-Hop Reading Comprehension. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2714–2725.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017. Billion-scale similarity search with GPUs. *arXiv preprint arXiv:1702.08734*.

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.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent Retrieval for Weakly Supervised Open Domain Question Answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6086–6096.

Hector J Levesque, Ernest Davis, and Leora Morgenstern. 2011. The Winograd schema challenge. *AAAI Spring Symposium: Logical Formalizations of Commonsense Reasoning*, 46:47.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv preprint arXiv:1907.11692*.

Ilya Loshchilov and Frank Hutter. 2017. Fixing Weight Decay Regularization in Adam. *arXiv preprint arXiv:1711.05101*.

Bingfeng Luo, Yansong Feng, Zheng Wang, Songfang Huang, Rui Yan, and Dongyan Zhao. 2018. Marrying Up Regular Expressions with Neural Networks: A Case Study for Spoken Language Understanding. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2083–2093.

Marco Marelli, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini, and Roberto Zamparelli. 2014. SemEval-2014 Task 1: Evaluation of Compositional Distributional Semantic Models on Full Sentences through Semantic Relatedness and Textual Entailment. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 1–8.

Rafael Müller, Simon Kornblith, and Geoffrey E Hinton. 2019. When does label smoothing help? In *Advances in Neural Information Processing Systems 32*, pages 4694–4703.

Matteo Negri, Marco Turchi, Rajen Chatterjee, and Nicola Bertoldi. 2018. ESCAPE: a Large-scale Synthetic Corpus for Automatic Post-Editing. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.

Yixin Nie, Songhe Wang, and Mohit Bansal. 2019. Revealing the Importance of Semantic Retrieval for Machine Reading at Scale. 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 2553–2566.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning*, pages 1310–1318.

Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. 2019. Towards Scalable Multi-domain Conversational Agents: The Schema-Guided Dialogue Dataset. *arXiv preprint arXiv:1909.05855*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-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)*, pages 3982–3992.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Minjoon Seo, Jinhyuk Lee, Tom Kwiatkowski, Ankur Parikh, Ali Farhadi, and Hannaneh Hajishirzi. 2019. Real-Time Open-Domain Question Answering with Dense-Sparse Phrase Index. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4430–4441.

Sam Shleifer. 2019. Low Resource Text Classification with ULMFit and Backtranslation. *arXiv preprint arXiv:1903.09244*.

Patrice Simard, Yann LeCun, and John S. Denker. 1993. Efficient Pattern Recognition Using a New Transformation Distance. In *Advances in Neural Information Processing Systems 5*, pages 50–58.

Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical Networks for Few-shot Learning. In *Advances in Neural Information Processing Systems 30*, pages 4077–4087.

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.

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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1199–1208.

Alona Sydorova, Nina Poerner, and Benjamin Roth. 2019. Interpretable Question Answering on Knowledge Bases and Text. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4943–4951.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, and Jon Shlens. 2016. Rethinking the Inception Architecture for Computer Vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems 30*, pages 5998–6008.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, koray kavukcuoglu, and Daan Wierstra. 2016a. Matching Networks for One Shot Learning. In *Advances in Neural Information Processing Systems 29*, pages 3630–3638.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. 2016b. Matching networks for one shot learning. In *Advances in neural information processing systems*, pages 3630–3638.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355.

Yusuke Watanabe, Bhuwan Dhingra, and Ruslan Salakhutdinov. 2017. Question Answering from Unstructured Text by Retrieval and Comprehension. *arXiv preprint arXiv:1703.08885*.

Jason Wei and Kai Zou. 2019. EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks. 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 6382–6388.

John Wieting and Kevin Gimpel. 2018. ParaNMT-50M: Pushing the Limits of Paraphrastic Sentence Embeddings with Millions of Machine Translations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 451–462.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. 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 1112–1122.

Chien-Sheng Wu, Steven Hoi, Richard Socher, and Caiming Xiong. 2020. ToD-BERT: Pre-trained Natural Language Understanding for Task-Oriented Dialogues. *arXiv preprint arXiv:2004.06871*.

Hu Xu, Bing Liu, Lei Shu, and P Yu. 2019. Open-world learning and application to product classification. In *The World Wide Web Conference*, pages 3413–3419.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V. Le. 2018. QANet: Combining Local Convolution with Global Self-Attention for Reading Comprehension. In *6th International Conference on Learning Representations (ICLR)*.

Appendix

A Training Details

Dataset preparation To use the CLINC150 dataset (Larson et al., 2019)[9] in our ways, especially for the single-domain experiments, we provide preprocessing scrips accompanied with our code.

General training This section describes the details about the model training in Section 4.3. For each component related to RoBERTa and SRoBERTa, we solely follow the two libraries, transformers and sentence-transformers, for the sake of easy reproduction of our experiments.[10] The example code to train the NLI-style models is also available.[11] We use the `roberta-base` configuration[12] for all the RoBERTa/SRoBERTa-based models in our experiments. All the model

[9] https://github.com/clinc/oos-eval.

[10] https://github.com/huggingface/transformers and https://github.com/UKPLab/sentence-transformers.

[11] https://github.com/huggingface/transformers/tree/master/examples/text-classification.

[12] https://s3.amazonaws.com/models.huggingface.co/bert/roberta-base-config.json.

Dataset	SNLI	WNLI	MNLI
Size of the development set	9999	70	9814
Accuracy	94.5%	41.4%	92.1%

Table 6: Development results on three NLI datasets.

parameters including the RoBERTa parameters are updated during all the fine-tuning processes, where we use the AdamW (Loshchilov and Hutter, 2017) optimizer with a weight decay coefficient of 0.01 for all the non-bias parameters. We use a gradient clipping technique (Pascanu et al., 2013) with a clipping value of 1.0, and also use a linear warmup learning-rate scheduling with a proportion of 0.1 with respect to the maximum number of training epochs.

Pre-training on NLI tasks For the pre-training on NLI tasks, we fine-tune a `roberta-base` model on three publicly available datasets, i.e., SNLI (Bowman et al., 2015), MNLI (Williams et al., 2018), and WNLI (Levesque et al., 2011) from the GLUE benchmark (Wang et al., 2018). The optimizer and gradient clipping follow the above configurations. The number of training epochs is set to 4; the batch size is set to 32; the learning rate is set to $2e-5$. We use a linear warmup learning-rate scheduling with a proportion of 0.06 by following Liu et al. (2019). The evaluation results on the development sets are shown in Table 6, where the low accuracy of WNLI is mainly caused by the data size imbalance. We note that these NLI scores are not comparable with existing NLI scores, because we converted the task to the binary classification task for our model transfer purpose.

Text pre-processing For all the RoBERTa-based models, we used the RoBERTa `roberta-base`'s tokenizer provided in the transformers library.[13] We did not perform any additional pre-processing in our experiments.

Hyper-parameter settings Table 7 shows the hyper-parameters we tuned on the development sets in our RoBERTa-based experiments. For a single-domain experiment, we take a hyper-parameter set and apply it to the ten different runs to select the threshold in Section 4.2 on the development set. We then select the best hyper-parameter set along with the corresponding threshold, which achieves the best $J_{\text{in\_oos}}$ in Equation (9) on the development set, among all the possible hyper-parameter sets. Finally, we apply the selected model and the threshold to the test set. We follow the same process for the all-domain experiments, except that we run each experiment five times. Table 8 and Table 9 summarize the hyper-parameter settings used for the evaluation on the test sets. We note that each model was not very sensitive to the different hyper-parameter settings, as long as we have a large number of training iterations.

B Data Augmentation

We describe the details about the classifier baselines with the data augmentation techniques in Section 4.3.

EDA Classifier-EDA uses the following four data augmentation techniques in Wei and Zou (2019): synonym replacement, random insertion, random swap, and random deletion. We follow the publicly available code.[14] For every training example, we empirically set one augmentation based on every technique. We apply each technique separately to the original sentence and therefore every training example will have four augmentations. The probability of a word in an utterance being edited is set to 0.1 for all the techniques.

BT For classifier-BT, we use the English-German corpus in Negri et al. (2018), which is widely used in an annual competition for automatic post-editing research on IT-domain text (Chatterjee et al., 2019). The corpus contains about 7.5 million translation pairs, and we follow the *base* configuration to train a transformer model (Vaswani et al., 2017) for each direction. Based on the initial trial in our preliminary experiments to generate diverse examples, we decided to use a temperature sampling technique instead of a greedy or beam-search strategy. More specifically, logit vectors during the machine translation process are multiplied by τ to distort the output distributions, where we set $\tau = 5.0$. For each training example in the intent detection

[13] https://github.com/huggingface/transformers/blob/master/src/transformers/tokenization_roberta.py.

[14] https://github.com/jasonwei20/eda_nlp.

	Single domain			All domains		
	Learning rate	**Epoch**	**Run**	**Learning rate**	**Epoch**	**Run**
Classifier	{1e-4, 2e-5, 5e-5}	{15, 25, 35}	10	{1e-4, 5e-5}	{15, 25, 35}	5
Emb-kNN	{1e-4, 2e-5, 3e-5}	{7, 10, 20, 25, 35}	10	{2e-5, 5e-5}	{3, 5, 7}	5
DNNC	{1e-5, 2e-5, 3e-5, 4e-5}	{7, 10, 15}	10	{2e-5, 5e-5}	{3, 5, 7}	5

Table 7: Some hyper-parameter settings for a few models.

	5-shot	**10-shot**
Classifier	{bs: 50, ep: 25.0, lr: 5e-05}	{bs: 50, ep: 35.0, lr: 5e-05}
Emb-kNN	{bs: 200, ep: 7.0, lr: 2e-05}	{bs: 200, ep: 5.0, lr: 2e-05}
DNNC	{bs: 900, ep: 7.0, lr: 2e-05}	{bs: 1800, ep: 5.0, lr: 2e-05}

Table 8: Best hyper-parameter settings for a few models on the all-domain experiments, where `bs` is batch size, `ep` represents epochs, `lr` is learning rate.

dataset, we first translate it into German and then translate it back to English. We repeat this process to generate up to five unique examples, and use them to train the classifier model. Table 10 shows such examples, and we will release all the augmented examples for future research.

C Additional Results

Visualization Figure 5 shows the same curves in Figure 3 along with the corresponding 10-shot results. We can see that the 10-shot results also exhibit the same trend. Figure 6 shows more visualization results with respect to Figure 1. Again, the 10-shot visualization shows the same trend.

Figure 7 and Figure 8 show 5-shot and 10-shot confidence levels on the test sets of the banking domain and all domains, respectively. Both Classifier and Emb-kNN cannot perform well to distinguish the in-domain examples from the OOS examples, while DNNC has a clearer distinction between the two.

Faster inference Figure 9 shows the same curves in Figure 4 also for the 10-shot setting. We can see the same trend with the 10-shot results.

Case studies Table 11 shows four DNNC prediction examples from the development set of the banking domain. For the first example, the input utterance is correctly predicted with a high confidence score, and it has a similarly matched utterance to the input utterance; for the second example, the input utterance is predicted incorrectly with a high confidence score, where the matched utterance is related to money but it has a slightly different meaning with the input utterance. For the third example, the model gives a very low confidence score to predict an OOS user utterance as an in-domain intent; the last example is an incorrect case where the input utterance and the matched utterance have a topically similar meaning, resulting in a high confidence score for the wrong label, "bill due." Based on these observations, it is an important direction to improve the model's robustness (even with the large-scale pre-trained models) towards such confusing cases.

	5-shot	10-shot	5-shot	10-shot
	Banking		**Credit cards**	
Classifier	{bs: 15, ep: 25.0, lr: 5e-05}	{bs: 15, ep: 35.0, lr: 5e-05}	{bs: 15, ep: 15.0, lr: 5e-05}	{bs: 15, ep: 25.0, lr: 5e-05}
Emb-kNN	{bs: 200, ep: 35.0, lr: 1e-05}	{bs: 200, ep: 25.0, lr: 2e-05}	{bs: 100, ep: 20.0, lr: 1e-05}	{bs: 100, ep: 10.0, lr: 1e-05}
DNNC	{bs: 370, ep: 15.0, lr: 1e-05}	{bs: 370, ep: 7.0, lr: 2e-05}	{bs: 370, ep: 15.0, lr: 2e-05}	{bs: 370, ep: 7.0, lr: 3e-05}
	Work		**Travel**	
Classifier	{bs: 15, ep: 15.0, lr: 5e-05}	{bs: 15, ep: 15.0, lr: 5e-05}	{bs: 15, ep: 35.0, lr: 5e-05}	{bs: 15, ep: 25.0, lr: 1e-04}
Emb-kNN	{bs: 100, ep: 20.0, lr: 1e-05}	{bs: 100, ep: 7.0, lr: 2e-05}	{bs: 100, ep: 35.0, lr: 3e-05}	{bs: 100, ep: 20.0, lr: 1e-05}
DNNC	{bs: 370, ep: 7.0, lr: 3e-05}	{bs: 370, ep: 15.0, lr: 2e-05}	{bs: 370, ep: 7.0, lr: 2e-05}	{bs: 370, ep: 7.0, lr: 2e-05}

Table 9: Best hyper-parameter settings for a few models on the four single domains, where `bs` is batch size, `ep` represents epochs, `lr` is learning rate.

Original utterance	Augmented example	Intent label
can you block my chase account right away please	can you turn my chase account off directly	freeze account
do a car payment from my savings account	with my saving account, you can pay a car payment account	pay bill
when is my visa due	when is my visa to be paid	bill due

Table 10: Examples used to train clasifier-BT.

input utterance	transfer ten dollars from my wells fargo account to my bank of america account
matched utterance	transfer $10 from checking to savings
label of the input utterance	transfer
label of the matched utterance	transfer
confidence score	0.934
input utterance	what transactions have i accrued buying dog food
matched utterance	what have i spent on food recently
label of the input utterance	transactions
label of the matched utterance	spending history
confidence score	0.915
input utterance	who has the best record in the nfl
matched utterance	do i have enough in my boa account for a new pair of skis
label of the input utterance	OOS
label of the matched utterance	balance
confidence score	0.006
input utterance	how long will it take me to pay off my card if i pay an extra $50 a month over the minimum
matched utterance	how long do i have left to pay for my chase credit card
label of the input utterance	OOS
label of the matched utterance	bill due
confidence score	0.945

Table 11: Case studies on the development set of banking domain. The first two cases are in-domain examples from the banking domain, and the rest are OOS examples.

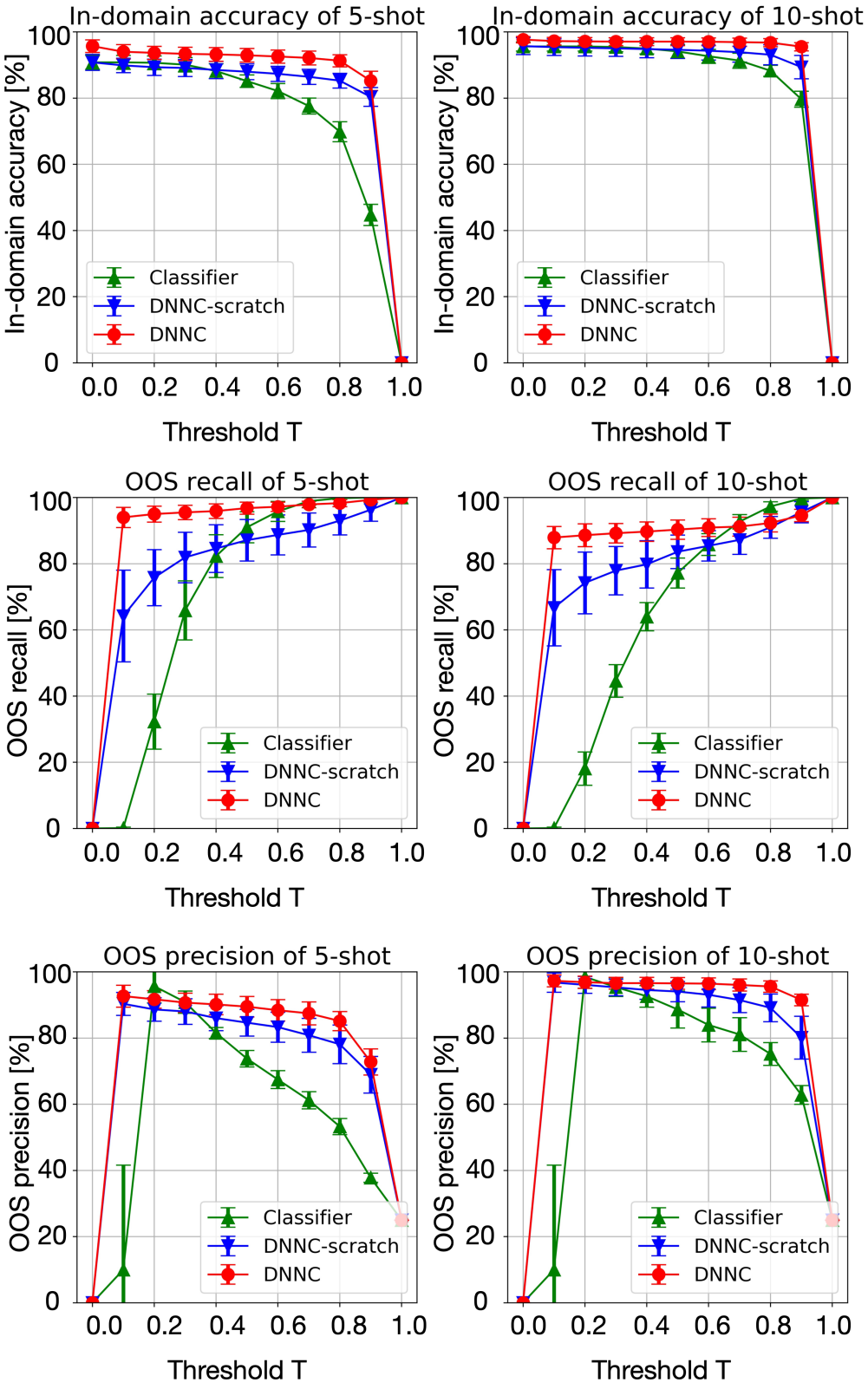

Figure 5: 5-shot and 10-shot development results on the banking domain. In this series of plots, a model with a higher area-under-the-curve is more robust.

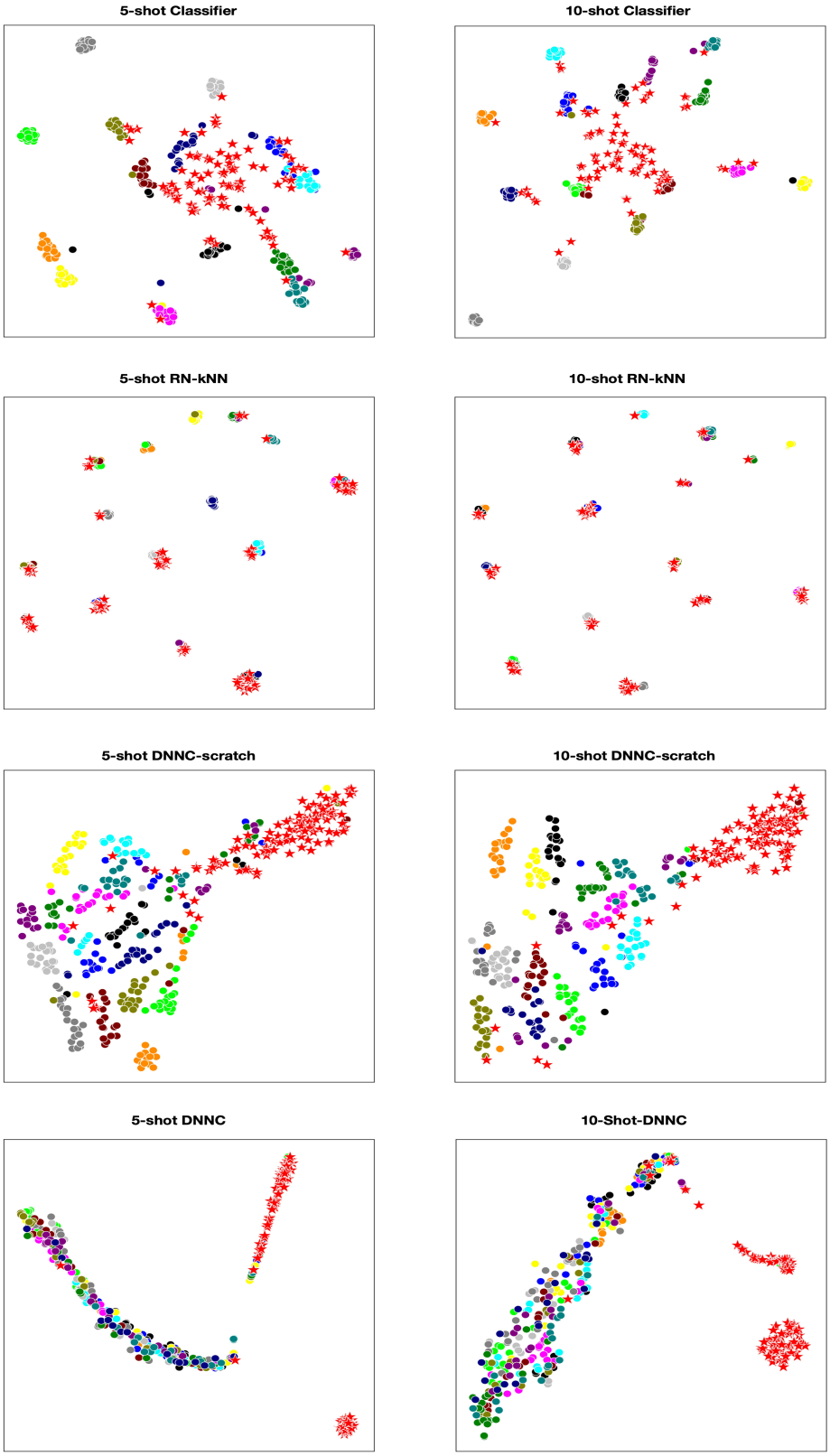

Figure 6: 5-shot and 10-shot tSNE visualizations on development set of the banking domain, where circles represent in-domain intent classes, and red stars represent out-of-scope intents.

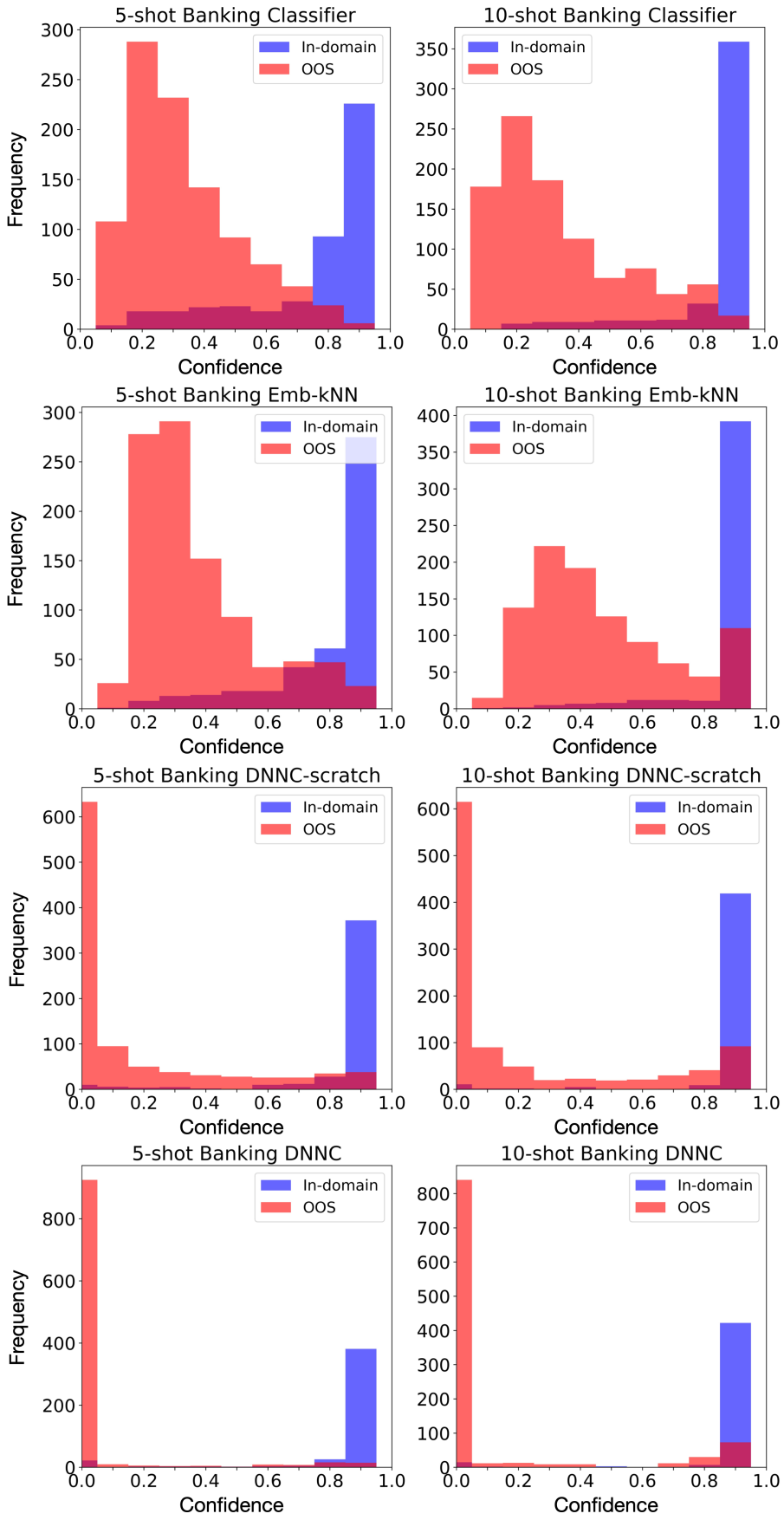

Figure 7: 5-shot and 10-shot confidence levels on test set of the banking domain. Best viewed in color.

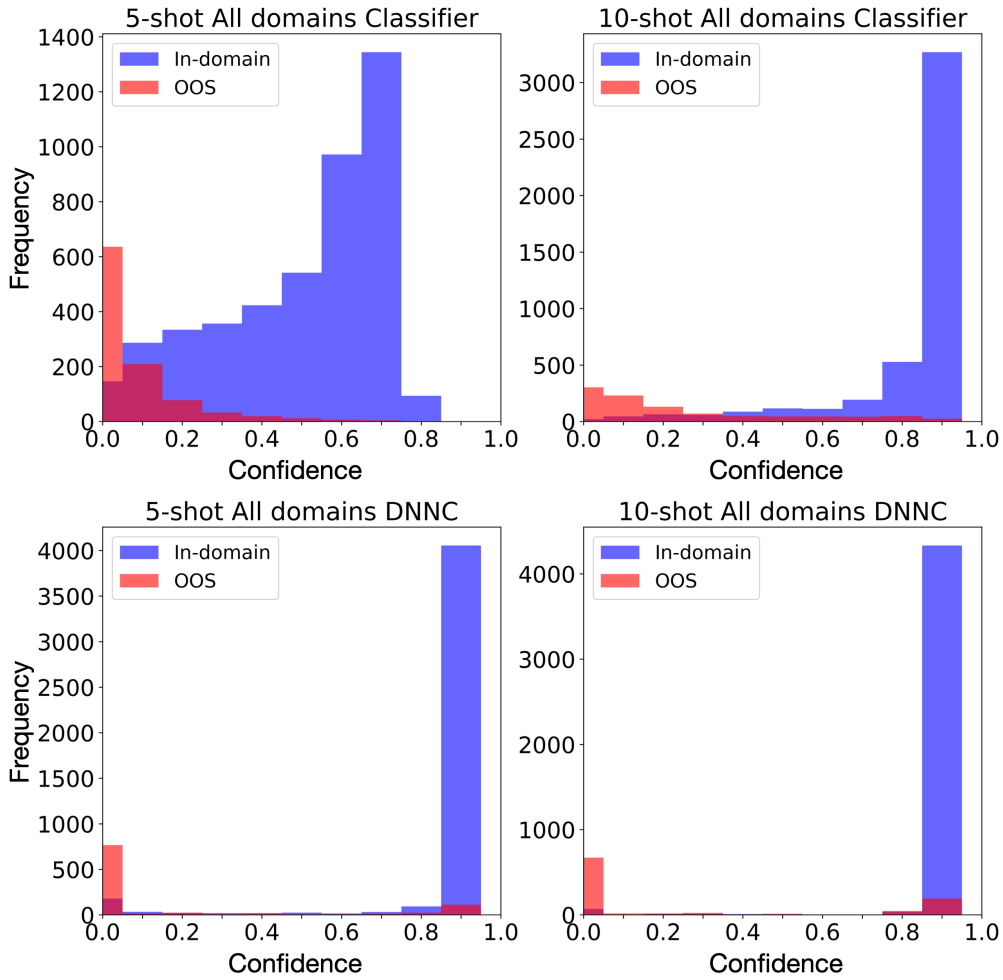

Figure 8: 5-shot and 10-shot confidence levels on test set of all domains. Best viewed in color.

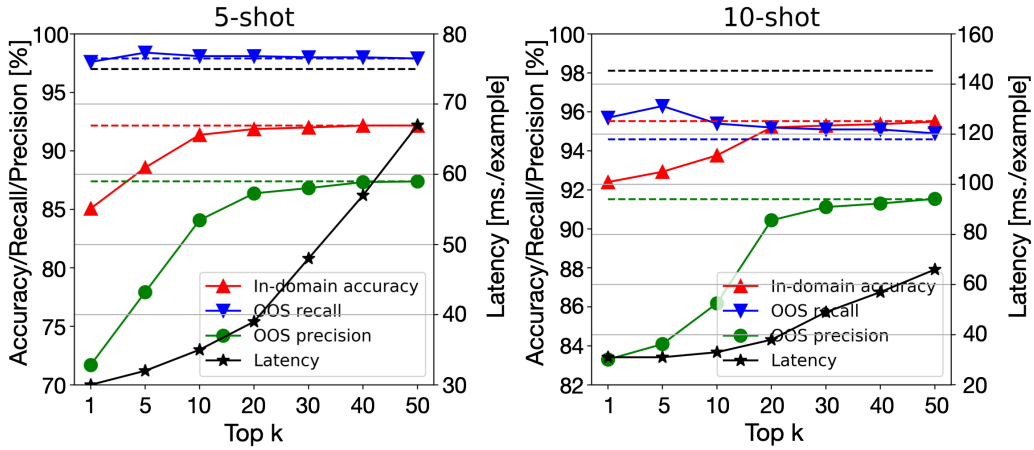

Figure 9: 5-shot and 10-shot DNNC-joint development results on the banking domain, where the dash lines are DNNC results.