# Strategies to Improve Few-shot Learning for Intent Classification and Slot-Filling

**Samyadeep Basu*, Karine lp Kiun Chong*, Amr Sharaf*, Alex Fischer, Vishal Rohra, Michael Amoake, Hazem El-Hammamy, Ehi Nosakhare, Vijay Ramani, Benjamin Han**
{sbasu, kaipkiun, amrsharaf, alex.fischer, virohra, miamoako, haelhamm, ehnosakh, vijayram, diha}@microsoft.com
**Microsoft AI**

## Abstract

Intent classification (IC) and slot filling (SF) are two fundamental tasks in modern Natural Language Understanding (NLU) systems. Collecting and annotating large amounts of data to train deep learning models for such systems are not scalable. This problem can be addressed by learning from few examples using fast supervised meta-learning techniques such as prototypical networks. In this work, we systematically investigate how contrastive learning and data augmentation methods can benefit these existing meta-learning pipelines for jointly modelled IC/SF tasks. Through extensive experiments across standard IC/SF benchmarks (SNIPS and ATIS), we show that our proposed approaches outperform standard meta-learning methods: contrastive losses as a regularizer in conjunction with prototypical networks consistently outperform the existing state-of-the-art for both IC and SF tasks, while data augmentation strategies primarily improve few-shot IC by a significant margin.

## 1 Introduction

NLU specific intent classification and slot-filling models often need to learn from only a few contextual examples given by the end user in industrial model deployment scenarios. Such models are often trained using meta-learning, a competitive few-shot learning strategy to learn from only a few examples. In this paper, we systematically dissect the existing meta-learning pipelines for jointly modelled few-shot Intent Classification (IC) and Slot Filling (SF) and identify practical training strategies to improve their performance by a significant margin. Precisely, we investigate how different data augmentation and contrastive learning strategies improve IC/SF performance, and show that our training approach outperforms state-of-the-art models for few-shot IC/SF. Given the user utterance: "*Book me a table for 6 at Lebanese Taverna*", an IC model identifies "*Restaurant Booking*" as the intent of interest, and an SF model identifies the slot types and values: *Party_Size:"6", Name: "Lebanese Taverna"*. These functionalities are typically driven by powerful deep learning models that rely on huge amounts of domain-specific training data. As such labeled data is rarely available, building models that can learn from only a few examples per class is inevitable.

Few-shot learning techniques (Krone et al., 2020; Ren and Xue, 2020; Geng et al., 2019, 2020; Liu et al., 2020b) have been recently proposed to address the problem of generalizing to unseen classes in IC/SF when only a few training examples per class are available. Krone et al. (2020) utilized meta-learning approaches such as prototypical networks (Snell et al., 2017) and MAML (Finn et al., 2017) to jointly model IC/SF. They showed that prototypical networks outperform other prevalent meta-learning techniques such as MAML as well as fine-tuning. Moreover, one primary benefit of prototypical networks is that it is computationally cheap during meta-testing, thus making it a good candidate for industrial few-shot learning systems. In this paper, we extend this powerful supervised meta-learning technique with unsupervised contrastive learning and data augmentation.

Rajendran et al. (2020) showed that meta-learners can be particular prone to overfitting which can be partially alleviated by data augmentation (Liu et al., 2020a). Data augmentation strategies in NLP have been shown to boost performance in general text classification settings (Wei and Zou, 2019b; Xie et al., 2019; Lee et al., 2021), however, there exists very little work on how data augmentation can be effectively used in the meta-learning pipeline specific to NLU tasks. To address this question, we first use a data augmentation strategy `slot-list values` for IC/SF tasks which generates synthetic utterances using dictionary-based slot-values. We note that similar dictionary based augmentation has been previously used in (Li et al., 2021), but in the context of dialogue state tracking, orthogonal to our use-case. Additionally, we investigate how state-of-the-art augmentation strategies such as backtranslation (Xie et al., 2019) and perturbation-based augmentations such as EDA – Easy Data Augmentation (Wei and Zou, 2019b) – can be used alongside prototypical networks.

We further investigate how contrastive learning (Chen et al., 2020) can be used as a regularizer during the meta-training stage to create better generalizable meta-learners. Contrastive learning is useful in creating improved prototypes as they pull similar representations together while pushing apart dissimilar ones. Through extensive experiments across SNIPS and ATIS, we show that meta-training with contrastive losses as a regularizer improves IC/SF performance for unseen classes with few examples. Our contributions include:

- We demonstrate the effectiveness of contrastive losses as a regularizer in meta-learning, by empirically showing how it improves few-shot IC/SF tasks across benchmark datasets.
- We illustrate the positive impact of data augmentation techniques such as `slot-list values`, backtranslation and EDA in the meta-learning pipeline.

## 2 Proposed Approaches

We follow the few-shot learning setup for IC/SF described in (Krone et al., 2020) with a few modifications. Instead of using a frozen backbone such as BERT or ELMo with a BiLSTM head, we use a more powerful pre-trained RoBERTa encoder. Additionally, in contrast to (Krone et al., 2020), we update our encoder during the meta-training stage. For a given utterance $x^i = \{x^i_1, x^i_2, ..., x^i_n\}$ with $n$ tokens, we first use the RoBERTa model denoted by $f_\phi$ to encode the utterance resulting in $h^i = \{h^i_{<cls>}, h^i_1, ..., h^i_n\}$. We use the `<cls>` token embedding to denote the utterance level embedding which we use for intent classification. For slot filling, we use each of the token embeddings $\{h^i_j\}^n_{j=1}$ of the $i^{th}$ utterance. Given a support set $S$, assuming $S_l$ consists of utterances belonging to the intent class $c_l$ and $S_a$ consists of tokens from the slot class $c_a$, we first compute the class prototypes for intents ($c_l$) and slots ($c_a$):

$$c_l = \frac{1}{|S_l|} \sum_{x^i \in S_l} f_\phi(x^i) \qquad (1)$$

$$c_a = \frac{1}{|S_a|} \sum_{x^i_j \in S_a} f_\phi(x^i_j) \quad \forall x^i \in S \qquad (2)$$

Given a query example $\mathbf{z}$ and a distance function $d$, a distribution over the different classes is computed using the softmax of the distances to the different class prototypes. Specifically we denote the intent specific log likelihood loss as:

$$L_{IC}(\phi, \mathbf{z}) = -\log\{\frac{\exp(-d(f_\phi(\mathbf{z}), c_l))}{\sum_{l'} \exp(-d(f_\phi(\mathbf{z}), c_{l'}))}\} \qquad (3)$$

We use euclidean distance as the standard distance function. Similarly, we define the slot specific loss as $L_{Slots}(\phi, \mathbf{z})$. For a given query set $Q$, the cumulative loss for intents and slots is the log likelihood averaged across all the query samples and is denoted by $L_{Total}(\phi)$:

$$L_{Total}(\phi) = \sum_{\mathbf{z} \in Q} \frac{1}{|Q|} \{L_{IC}(\phi, \mathbf{z}) + L_{Slots}(\phi, \mathbf{z})\} \qquad (4)$$

### 2.1 Contrastive Learning

The general idea of contrastive learning (Chen et al., 2020) is to pull together the representations of similar samples while pushing apart the representations of dissimilar samples in an embedding space. In our work, we specifically incorporate the supervised contrastive loss as an added regularizer with the prototypical loss computation in Eq. (4). In particular we identify places in the meta-training pipeline where the incorporation of the contrastive loss is most beneficial for good generalization to few-shot classes. We devise two types of contrastive losses for the IC/SF tasks: (a) contrastive loss for intents $L_{contrastiveIC}(\phi)$ where the `<cls>` token embedding is used in the loss; (b) contrastive loss for slots $L_{contrastiveSF}(\phi)$ where the individual token embeddings are used in the loss. The regularized prototypical loss is the following:

$$L_{Total}(\phi) = \sum_{\mathbf{z} \in Q} \frac{1}{|Q|} \{L_{IC}(\phi, \mathbf{z}) + L_{Slots}(\phi, \mathbf{z})\} + \lambda_1 L_{contrastiveIC}(\phi) + \lambda_2 L_{contrastiveSF}(\phi) \qquad (5)$$

We provide more details about the two contrastive losses in the Appendix section.

### 2.2 Data Augmentation for Few-shot IC/SF

Prior works in computer vision (Liu et al., 2020a; Ni et al., 2020) have shown that data augmentation

| | Level | **SNIPS** (Kmax=20) | | **ATIS** (Kmax=20) | | **SNIPS** (Kmax=100) | | **ATIS** (Kmax=100) | |
|---|---|---|---|---|---|---|---|---|---|
| | | IC Acc | Slot F1 | IC Acc | Slot F1 | IC Acc | Slot F1 | IC Acc | Slot F1 |
| Krone et al. (2020) | - | 0.877 ± 0.01 | 0.597 ± 0.017 | 0.660 ± 0.02 | 0.340 ± 0.004 | 0.877 ± 0.01 | 0.621 ± 0.007 | 0.719 ± 0.01 | 0.412 ± 0.02 |
| *Baseline* (Ours) | - | 0.887 ± 0.06 | 0.597 ± 0.04 | 0.737 ± 0.06 | 0.74 ± 0.01 | 0.907 ± 0.05 | 0.593 ± 0.04 | 0.80 ± 0.04 | 0.70± 0.02 |
| CL (IC) | Support(m-train) | 0.905 ± 0.05 | 0.594 ± 0.04 | 0.75 ± 0.07 | 0.748 ± 0.02 | 0.912 ± 0.03 | 0.594 ± 0.04 | 0.802 ± 0.06 | 0.70 ± 0.02 |
| CL (IC) | Support,Query(m-train) | 0.908 ± 0.06 | 0.596 ± 0.04 | **0.76 ± 0.04** | 0.748 ± 0.02 | 0.93 ± 0.05 | 0.60 ± 0.03 | 0.829 ± 0.06 | 0.703 ± 0.03 |
| CL (IC + SF) | Support(m-train) | 0.903 ± 0.06 | 0.60± 0.04 | 0.757 ± 0.04 | 0.755 ± 0.02 | 0.92 ± 0.01 | 0.60 ± 0.04 | 0.826 ± 0.05 | 0.70 ± 0.03 |
| CL (IC + SF) | Support,Query(m-train) | **0.91 ± 0.04** | **0.60 ± 0.03** | 0.75 ± 0.07 | **0.756 ± 0.02** | **0.93 ± 0.03** | **0.60 ± 0.04** | **0.833 ± 0.05** | **0.71 ± 0.02** |
| CL (IC + SF), DA (Slot list) | Support,Query(m-train) | **0.921± 0.037** | **0.619± 0.037** | **0.803 ± 0.069** | 0.748 ± 0.019 | 0.923± 0.055 | **0.619± 0.035** | 0.821± 0.08 | **0.73± 0.02** |

Table 1: Few-shot classification accuracy with contrastive learning (CL) for prototypical networks. For CL (IC) only $L_{contrastiveIC}$ is used, whereas for CL (IC + SF), both $L_{contrastiveIC}$ and $L_{contrastiveSF}$ are used.

is very effective in meta-learning. In this section, we use various data augmentation strategies to improve the meta-learning pipeline for IC/SF tasks. Data augmentation for joint IC/SF tasks in NLU is particularly challenging as the augmentation is primarily possible at the level of intents. For intent level data augmentation, we use state-of-the-art techniques such as backtranslation (Xie et al., 2019) and EDA (Wei and Zou, 2019b) along with prototypical networks. We also introduce a novel data augmentation technique called `slot-list values` which effectively leverages the structure of joint IC/SF tasks. In particular, we investigate the effectiveness of these data augmentation techniques in the meta-learning pipeline at different levels such as: (a) support at meta-training; (b) support + query at meta-training; (c) support at meta-testing; (d) combination of those. We provide details about these augmentation methods below.

#### 2.2.1 `Slot-List Values` Augmentation

In IC/SF datasets, certain slot types often can take on values specified in a finite list. For example, in the SNIPS dataset the slot type *facility* can take on values from the list *["smoking room", "spa", "indoor", "outdoor", "pool", "internet", "parking", "wifi"]* . Specific to the discrete slot filling task, (Shah et al., 2019) used such values to learn an additional attention module for improving SF. Such lists can be created from the training dataset and be used for data augmentation. We leverage such lists to create synthetic utterances by replacing the values of slot types in a given utterance with other values from the list: e.g. given an utterance "*Book a table at a pool bar*", we synthesize another utterance "*Book a table at a indoor bar*".

#### 2.2.2 Augmentation by Backtranslation

Backtranslation is a technique of translating an utterance into an intermediate language and back to its original language using a neural machine translation model. Previous works (Edunov et al., 2018; Yu et al., 2018; Sennrich et al., 2015) showed that backtranslation is extremely effective as a data augmentation technique for NLP applications. In our paper in particular, we use a pre-trained `en-es` NMT model (Junczys-Dowmunt et al., 2018) for generating the augmented utterances. To ensure that the generated utterances are diverse, we follow the procedure in (Xie et al., 2019) in which we employ restricted sampling from the model output probability distribution instead of beam-search.

#### 2.2.3 EDA Data Augmentation

Adding small perturbations to the training data via random insertion, deletion, swapping and synonym replacement is one simple technique to generate synthetic data for data augmentation. Previous work by (Wei and Zou, 2019a) showed that EDA achieves state-of-the-art results on text-classification tasks. In our work, we use EDA to generate synthetic data to perform data augmentation at different stages of meta-learning.

## 3 Experiments

**Datasets:** We use two well-known IC/SF benchmarks: SNIPS (Coucke et al., 2018) and ATIS (Hemphill et al., 1990). SNIPS is a more challenging dataset as it contains intents from diverse domains whereas the ATIS dataset contains intents from only the *Airline* domain.

**Episode Construction:** We follow the standard episode construction technique described in (Krone et al., 2020; Triantafillou et al., 2020) where the number of classes and the shots per class in each episode are sampled dynamically.

**Few-shot Splits:** For the SNIPS dataset, we use 4 intent classes for meta-training and 3 intent classes for meta-testing. Similar to (Krone et al., 2020), we do not form a development split for SNIPS as there are only 7 intent classes and the episode construction process requires at least 3 classes in each split. For the ATIS dataset, we first select intent classes with more than 15 examples, then use 5 intent classes for meta-training and 7 intent classes for meta-testing. The rest of the

| | Level | SNIPS(Kmax=20) | ATIS (Kmax=20) | SNIPS (Kmax=100) | ATIS(Kmax=100) |
|---|---|---|---|---|---|
| | | IC Acc | IC Acc | IC Acc | IC Acc |
| (Krone et al., 2020) | - | 0.877 ± 0.01 | 0.660 ± 0.02 | 0.877 ± 0.01 | 0.719 ± 0.01 |
| *Baseline* (Ours) | - | 0.887 ± 0.06 | 0.737 ± 0.06 | 0.907 ± 0.05 | 0.80 ± 0.04 |
| DA (Slot-list) | Support(m-train) | 0.898 ± 0.061 | 0.735 ± 0.052 | 0.916 ± 0.055 | 0.810 ± 0.052 |
| DA (Slot-list) | Support,Query(m-train) | 0.919 ± 0.062 | **0.800 ± 0.054** | 0.917 ± 0.051 | 0.806 ± 0.066 |
| DA (Slot-list) | Support(m-train, m-test) | 0.905± 0.062 | 0.772 ± 0.044 | 0.922± 0.051 | 0.818± 0.056 |
| DA (Slot-list) | Support(m-test) | **0.926 ± 0.038** | 0.764 ± 0.073 | **0.931 ± 0.037** | **0.840± 0.047** |
| DA (Backtranslation) | Support(m-train) | 0.885 ± 0.03 | 0.77 ± 0.06 | 0.928 ± 0.029 | 0.79 ± 0.06 |
| DA (Backtranslation) | Support(m-train, m-test) | 0.881 ± 0.03 | 0.79 ± 0.05 | **0.931 ± 0.030** | 0.795 ± 0.06 |
| DA (Backtranslation) | Support(m-test) | 0.895 ± 0.036 | 0.71 ± 0.06 | 0.899 ± 0.06 | 0.77 ± 0.14 |
| DA (EDA) | Support(m-train) | 0.893 ± 0.062 | 0.787 ± 0.07 | 0.911 ± 0.04 | 0.805 ± 0.08 |
| DA (EDA) | Support(m-train,m-test) | 0.893 ± 0.047 | 0.761 ± 0.08 | 0.915 ± 0.04 | 0.808 ± 0.10 |
| DA (EDA) | Support(m-test) | 0.892 ± 0.047 | 0.731 ± 0.06 | 0.915 ± 0.05 | 0.78 ± 0.059 |

Table 2: Few-shot IC accuracy with Data Augmentation (DA) for prototypical networks; m-train refers to meta-training and m-test refers to meta-testing

classes are used as a development split. In (Krone et al., 2020), the intent classes for each split are *manually chosen*. This is not representative of realistic situations where the types of few-shot classes can vary considerably. To address this issue, we report our experiment results averaged over 5 seeds where in each run the intent classes for each split are randomly sampled. In each experiment run, we evaluate our results for 100 episodes sampled from the test-split. We refer to our re-implementation of (Krone et al., 2020) with this strategy as *Baseline*.

**Contrastive Learning Helps IC/SF tasks:** Table 1 shows the results of experiments adding contrastive losses as a regularizer to our baseline. Overall, we observe that across both SNIPS and ATIS datasets, using contrastive losses as a regularizer predominantly improves IC accuracy, while marginally improving SF F1 score. In particular, we notice that using contrastive losses as a regularizer with both the support and query during meta-training leads to the best performances.

**Impact of Data Augmentation is Dependent on Stage of Application:** Table 2 shows the results of adding data augmentation to the few-shot IC tasks. We find that the data augmentation techniques in general improve the performance of few-shot IC, depending on the stage in the meta-learning pipeline at which the data is augmented. More specifically, for SNIPS we notice up to 4% and 2% gain in IC accuracy for $Kmax = 20$ and $Kmax = 100$ respectively. With EDA, we find that augmentation during meta-training and meta-testing together leads to a noteworthy gain in few-shot IC performances across both SNIPS and ATIS. In comparison, backtranslation is effective in improving the few-shot IC performance for SNIPS, when the shots per class is higher such as in $Kmax = 100$. However for ATIS, we observe a significant gain in IC only for $Kmax = 20$.

**Slot-list Values Augmentation at Meta-Testing Helps:** We find that dictionary based augmentation techniques such as `slot-list values` generally show consistent gain in IC at all stages during meta-training and shots per class.

**Combination of Contrastive Learning and Data Augmentation Helps IC/SF tasks:** We find that the combination of contrastive losses and data augmentation via `slot-list values` outperforms models trained independently with only contrastive losses or data augmentation. We hypothesize that this is due to two independent effects working together in conjunction: (a) contrastive learning helps to create improved prototypes whereas (b) data augmentation helps mitigate meta-overfitting.

For SF, we find that data augmentation leads to only limited improvements when compared to IC (see Appendix C). We attribute this to the low shots per slot class, an artifact of the episodic sampling procedure (Krone et al., 2020), done per intent class in the joint IC/SF setting.

## 4 Conclusion

In this work, we systematically dissect meta-learning pipelines for few-shot IC/SF tasks and identify stages during meta-learning where contrastive learning and data augmentation can be effective. Empirically, we found that contrastive losses are effective regularizers during meta-training and outperform the current state-of-the-art few-shot joint IC/SF benchmarks across both SNIPS and ATIS. Impact of data augmentation in general is highly dependent on the stage at which it is applied during meta-learning. Notably, a combination of contrastive losses and data augmentation via `slot-list values` during meta-training leads to the best performances across both SNIPS and ATIS. These strategies for improving few-shot IC/SF tasks create a strong benchmark and open up possibilities on more stronger modes of meta-specific augmentation and contrastive learning.

## References


Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations.

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.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. *CoRR*, abs/1808.09381.

Steven Y. Feng, Varun Gangal, Jason Wei, Sarath Chandar, Soroush Vosoughi, Teruko Mitamura, and Eduard Hovy. 2021. A survey of data augmentation approaches for nlp.

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 of *Proceedings of Machine Learning Research*, pages 1126–1135. PMLR.

Mauajama Firdaus, Shobhit Bhatnagar, Asif Ekbal, and Pushpak Bhattacharyya. 2018. A deep learning based multi-task ensemble model for intent detection and slot filling in spoken language understanding. In *Neural Information Processing*, pages 647–658, Cham. Springer International Publishing.

Ruiying Geng, Binhua Li, Yongbin Li, Jian Sun, and Xiaodan Zhu. 2020. Dynamic memory induction networks for few-shot text classification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1087–1094, Online. Association for Computational Linguistics.

Ruiying Geng, Binhua Li, Yongbin Li, Yuxiao Ye, Ping Jian, and Jian Sun. 2019. Few-shot text classification with induction network. *CoRR*, abs/1902.10482.

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

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does BERT learn about the structure of language? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3651–3657, Florence, Italy. Association for Computational Linguistics.

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Tomasz Dwojak, Hieu Hoang, Kenneth Heafield, Tom Neckermann, Frank Seide, Ulrich Germann, Alham Fikri Aji, Nikolay Bogoychev, André F. T. Martins, and Alexandra Birch. 2018. Marian: Fast neural machine translation in C++. In *Proceedings of ACL 2018, System Demonstrations*, pages 116–121, Melbourne, Australia. Association for Computational Linguistics.

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. 2020. Supervised contrastive learning. *CoRR*, abs/2004.11362.

Jason Krone, Yi Zhang, and Mona Diab. 2020. Learning to classify intents and slot labels given a handful of examples.

Kenton Lee, Kelvin Guu, Luheng He, Tim Dozat, and Hyung Won Chung. 2021. Neural data augmentation via example extrapolation.

Shiyang Li, Semih Yavuz, Kazuma Hashimoto, Jia Li, Tong Niu, Nazneen Rajani, Xifeng Yan, Yingbo Zhou, and Caiming Xiong. 2021. Coco: Controllable counterfactuals for evaluating dialogue state trackers.

Jialin Liu, Fei Chao, and Chih-Min Lin. 2020a. Task augmentation by rotating for meta-learning.

Zihan Liu, Genta Indra Winata, Peng Xu, and Pascale Fung. 2020b. Coach: A coarse-to-fine approach for cross-domain slot filling. *CoRR*, abs/2004.11727.

Renkun Ni, Micah Goldblum, Amr Sharaf, Kezhi Kong, and Tom Goldstein. 2020. Data augmentation for meta-learning.

Janarthanan Rajendran, Alex Irpan, and Eric Jang. 2020. Meta-learning requires meta-augmentation.

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

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Improving neural machine translation models with monolingual data. *CoRR*, abs/1511.06709.

Darsh J. Shah, Raghav Gupta, Amir A. Fayazi, and Dilek Hakkani-Tür. 2019. Robust zero-shot cross-domain slot filling with example values. *CoRR*, abs/1906.06870.

Jake Snell, Kevin Swersky, and Richard S. Zemel. 2017. Prototypical networks for few-shot learning. *CoRR*, abs/1703.05175.

Dhanasekar Sundararaman, Vivek Subramanian, Guoyin Wang, Shijing Si, Dinghan Shen, Dong Wang, and Lawrence Carin. 2019. Syntax-infused transformer and BERT models for machine translation and natural language understanding. *CoRR*, abs/1911.06156.

Eleni Triantafillou, Tyler Zhu, Vincent Dumoulin, Pascal Lamblin, Utku Evci, Kelvin Xu, Ross Goroshin, Carles Gelada, Kevin Swersky, Pierre-Antoine Manzagol, and Hugo Larochelle. 2020. Meta-dataset: A dataset of datasets for learning to learn from few examples.

Jixuan Wang, Kai Wei, Martin Radfar, Weiwei Zhang, and Clement Chung. 2020. Encoding syntactic knowledge in transformer encoder for intent detection and slot filling.

Jason Wei and Kai Zou. 2019a. Eda: Easy data augmentation techniques for boosting performance on text classification tasks.

Jason W. Wei and Kai Zou. 2019b. EDA: easy data augmentation techniques for boosting performance on text classification tasks. *CoRR*, abs/1901.11196.

Qizhe Xie, Zihang Dai, Eduard H. Hovy, Minh-Thang Luong, and Quoc V. Le. 2019. Unsupervised data augmentation. *CoRR*, abs/1904.12848.

Wenxiu Xie, Dongfa Gao, Ruoyao Ding, and Tianyong Hao. 2018. A feature-enriched method for user intent classification by leveraging semantic tag expansion. In *Natural Language Processing and Chinese Computing*, pages 224–234, Cham. Springer International Publishing.

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. *CoRR*, abs/1804.09541.

Chenwei Zhang, Wei Fan, Nan Du, and Philip S. Yu. 2016. Mining user intentions from medical queries: A neural network based heterogeneous jointly modeling approach. In *Proceedings of the 25th International Conference on World Wide Web*, WWW '16, page 1373–1384, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee.

## A Hyperparameters

For the ATIS dataset, we use the development set to tune for $\lambda_1$ and $\lambda_2$ in Eq. (5). For the SNIPS dataset, we empirically set both $\lambda_1$ and $\lambda_2$ to be 0.06 due to the lack of a development set. In our experiments with the three data augmentation strategies, we generate synthetic utterances to exactly double the training data size for fair comparison throughout. Across all the experiments, we meta-train the models for 50 episodes and use a learning rate of $5e-5$.

## B On Contrastive Learning

In our work, we use two types of contrastive losses for IC/SF tasks: (a) contrastive loss for intents $L_{contrastiveIC}(\phi)$ where the `<cls>` token embedding is used in the loss; (b) contrastive loss for slots $L_{contrastiveSF}(\phi)$ where the individual token embeddings from the encoder are used in the loss. In particular, we use the supervised contrastive loss (Khosla et al., 2020) and leverage the label information present in the support or support + query set during meta-training. First we define the contrastive loss for intents $L_{contrastiveIC}(\phi)$: given a set of utterances with their corresponding intent labels $S_{intents} = \{(x_i, y_i)_{i=1}^m\}$, assume $P(i)$ to be a set consisting of examples from $S_{intents}$ with same labels as the $i^{th}$ example. Formally $P(i) : \{x_j : y_j = y_i \;\; \forall j \in [1, m] \;\; \& \;\; j \neq i\}$. The contrastive loss for the intents $L_{contrastiveIC}(\phi)$ is defined as the following:

$$\sum_{i=1}^{m} - \log \left\{ \frac{1}{|P(i)|} \sum_{z \in P(i)} \frac{\exp(f_\phi(x_i)^T f_\phi(z))/\tau}{\sum_{j=1, j\neq i}^{m} \exp(f_\phi(x_i)^T f_\phi(x_j))/\tau} \right\} \quad (6)$$

Here $f_\phi(x_i)$ denotes the `<cls>` embedding for the $i^{th}$ utterance. In case of slots, we first obtain the individual token embeddings in each utterance $x_i \;\; \forall i \in [1, m]$. Consider the total number of tokens to be $N$ in an episode and their associated embeddings' set to be $S_{slots} = \{(h_j, y_j^{'}), \;\; \forall j \in N\}$, where $y_j^{'}$ is the slot label for the $j^{th}$ token. Similar to the intents, we define the set $Q(i) : \{h_j : y_j^{'} = y_i^{'} \;\; \forall j \in [1, N] \;\; \& \;\; j \neq i\}$. Next we define the contrastive loss for the slots $L_{slots}(\phi)$ as:

$$\sum_{i=1}^{N} - \log \left\{ \frac{1}{|Q(i)|} \sum_{z \in Q(i)} \frac{\exp(h_i^T z)/\tau}{\sum_{j=1, j\neq i}^{N} \exp(h_i^T h_j)/\tau} \right\} \quad (7)$$

## C Impact of Data Augmentation for Slot Filling

Data augmentation for joint IC/SF tasks is challenging as augmentation is only possible at the level of intents. Although data augmentation leads to large improvements in few-shot IC performances, its impact on SF tasks is limited. From Table 3, across the different data augmentation methods such as backtranslation, EDA and `slot-list values`, we observe that there is no consistent improvements in SF performances across our different experiment settings. We hypothesize that as data augmentation does not provide any direct signal to the SF task, the improvements are insubstantial. To address this issue and provide a more direct signal to the SF task, we incorporate part-of-speech (POS) and noun-phrase information of the different slot values into the joint IC/SF model. In the next section, we discuss ways to incorporate these additional syntactic information into the meta-learning pipeline.

## D Beyond Semantic Information

Part-of-speech (POS) and noun-parser information can provide additional syntactic information about of an utterance, thus augmenting the semantic information from the encoded tokens. In particular, POS tags can help resolve decisions for ambiguous tokens or words. Previous work (Wang et al., 2020) has shown that prior information from POS tags helps in improving IC and SF tasks in the general supervised many shot setting. In our work, we use POS tags as an additional source of information particularly for the few-shot setting. We propose two primary ways to incorporate POS tags in the general meta-learning setting: (a) POS tag as an additional input feature; (b) Explicitly training the model to predict POS tags via a multi-task loss.

In addition to POS tags, we also augment information about noun-phrases as an additional input feature. Noun chunks or phrases have the potential to provide strong signals about possible spans of different slots to the underlying model, thus improving SF performance. For example, in the utterance "*book me a table for one at blue ribbon barbecue*"(with intent *BookRestaurant*, and slots: *party_size_number:"one", restaurant_name: "blue ribbon barbecue"*), *"blue ribbon barbecue"* is identified as a noun-chunk and the span information

| | Level | SNIPS(Kmax=20) | ATIS (Kmax=20) | SNIPS (Kmax=100) | ATIS(Kmax=100) |
|---|---|---|---|---|---|
| | | Slot F1 | Slot F1 | Slot F1 | Slot F1 |
| *Baseline* (Ours) | - | 0.599 ± 0.04 | 0.748 ± 0.01 | 0.593 ± 0.04 | 0.703 ± 0.02 |
| DA (Slot-list) | Support(m-train) | 0.603 ± 0.043 | 0.738 ± 0.020 | 0.609 ± 0.047 | 0.713 ± 0.025 |
| DA (Slot-list) | Support,Query(m-train) | **0.609 ± 0.043** | 0.74 ± 0.02 | 0.609 ± 0.03 | 0.715 ± 0.02 |
| DA (Slot-list) | Support(m-train, m-test) | 0.587± 0.045 | 0.712 ± 0.026 | 0.595 ± 0.042 | 0.686± 0.029 |
| DA (Slot-list) | Support(m-test) | 0.572 ± 0.036 | 0.697 ± 0.028 | 0.589 ± 0.042 | 0.684± 0.02 |
| DA (Backtranslation) | Support(m-train) | 0.595 ± 0.04 | 0.742 ± 0.01 | **0.611 ± 0.036** | **0.716 ± 0.02** |
| DA (Backtranslation) | Support(m-train, m-test) | 0.595 ± 0.04 | 0.742 ± 0.01 | **0.611 ± 0.03** | 0.716 ± 0.02 |
| DA(Backtranslation) | Support(m-test) | 0.598 ± 0.03 | 0.74 ± 0.01 | 0.60 ± 0.03 | 0.72 ± 0.01 |
| DA(EDA) | Support(m-train) | 0.585 ± 0.032 | 0.742 ± 0.02 | 0.596 ± 0.05 | 0.701 ± 0.03 |
| DA(EDA) | Support(m-train,m-test) | 0.593 ± 0.033 | 0.742 ± 0.02 | 0.594 ± 0.04 | 0.711 ± 0.005 |
| DA(EDA) | Support(m-test) | 0.586 ± 0.036 | 0.74 ± 0.01 | 0.593 ± 0.037 | **0.714 ± 0.02** |

Table 3: Few-shot Slot F1 with Data Augmentation (DA) for prototypical networks; m-train refers to meta-training and m-test refers to meta-testing

| | SNIPS (Kmax = 20) | | SNIPS(Kmax=100) | | ATIS(Kmax=20) | | ATIS(Kmax=100) | |
|---|---|---|---|---|---|---|---|---|
| | IC Acc | Slot F1 | IC Acc | Slot F1 | IC Acc | Slot F1 | IC Acc | Slot F1 |
| *Baseline* (Ours) | 0.887 ± 0.06 | 0.597 ± 0.04 | 0.907 ± 0.05 | 0.593 ± 0.04 | 0.737 ± 0.06 | 0.748 ± 0.02 | 0.801 ± 0.05 | 0.703 ± 0.02 |
| Multi-task POS loss | 0.905 ± 0.04 | **0.603 ± 0.03** | **0.929 ± 0.03** | 0.595 ± 0.03 | **0.769 ± 0.06** | 0.75 ± 0.01 | 0.807 ± 0.05 | 0.711 ± 0.02 |
| With POS-tag features | 0.896 ± 0.06 | 0.592 ± 0.04 | 0.926 ± 0.03 | 0.590 ± 0.04 | 0.745 ± 0.06 | 0.747 ± 0.01 | 0.793 ± 0.09 | 0.713 ± 0.02 |
| With noun-parser features | **0.912 ± 0.05** | 0.599 ± 0.04 | 0.897 ± 0.05 | **0.597 ± 0.03** | 0.764 ± 0.04 | **0.755 ± 0.02** | 0.805 ± 0.07 | **0.715 ± 0.02** |

Table 4: Effect of adding syntactic information into the joint IC/SF model

can potentially help with the SF task for the *restaurant_name* slot. Conversely, the POS tag for "*one*" is *NUM* and can help classify numeric words to the numeric slot *party_size_number*.

### D.1 Feature-Based Addition

Previous works have shown that adding POS tags as features improves IC (Zhang et al., 2016; Xie et al., 2018) as well SF performances (Firdaus et al., 2018) in many-shot settings. In this work we look into incorporating syntactic features in our meta-learning pipeline. A simple idea to incorporate POS or noun-chunk tags of an utterance is to concatenate a vector representation of them, $p^i_j$ and $\eta^i_j$ respectively, with the token embeddings $f_\phi(x^i_j)$. Formally, in our meta-learning pipeline, we revise Eq. (2) for our slot prototype:

$$c_a = \frac{1}{|S_a|} \sum_{x^i_j \in S_a} f_\phi(x^i_j) \oplus p^i_j \oplus \eta^i_j \quad \forall x^i \in S \quad (8)$$

### D.2 Multi-task POS Loss

Although training language models distills implicitly the structural knowledge of the underlying languages (Jawahar et al., 2019; Sundararaman et al., 2019) into the model, such knowledge can be imperfect. Explicitly training to learn structural knowledge such as POS tags (Wang et al., 2020), however, can help the model to improve on downstream tasks such as IC/SF. We treat POS tagging as a token level classification problem, similar to SF. Given a support set $S$, assume $S_l$ to consist of utterances belonging to the intent class $c_l$, $S_a$ to consist of tokens from the slot class $c_a$ and $S_{pos}$ to consist of POS tag tokens from the class $c_{pos}$. In addition to the intent class prototypes $c_l$ and slot class prototypes $c_a$, we define an additional class prototype $c_{pos}$ for the POS tags:

$$c_{pos} = \frac{1}{|S_{pos}|} \sum_{x^i_j \in S_{pos}} f_\phi(x^i_j) \quad \forall x^i \in S \quad (9)$$

Given a query example $\mathbf{z}$, we define the corresponding loss with the POS tag prototypes as:

$$L_{pos}(\phi, \mathbf{z}) = -\log\{\frac{\exp(-d(f_\phi(\mathbf{z}), c_{pos}))}{\sum_{pos'} \exp(-d(f_\phi(\mathbf{z}), c_{pos'}))}\} \quad (10)$$

For the query set $Q$, the composite loss function is the following:

$$L_{Total}(\phi) = \sum_{\mathbf{z} \in Q} \frac{1}{|Q|}\{L_{IC}(\phi, \mathbf{z}) + L_{Slots}(\phi, \mathbf{z}) + \beta L_{pos}(\phi, \mathbf{z})\} \quad (11)$$

where $\beta$ is a hyperparameter. For the ATIS dataset, we select $\beta$ by using a validation set. In case of the SNIPS dataset, we empirically set $\beta$ as 0.01 due to unavailability of a development set.

In Table 4, we observe an improvement in both IC and SF over the baseline with the addition of information from the POS tags as an auxilliary loss.

However, similar to feature-based addition, we notice only a marginal and small improvement for SF. To understand further this issue, we exmined the episodic sampling procedure used in (Krone et al., 2020). Across both the SNIPS and ATIS datasets, the average shots per class for intents are $\approx 5$ and $\approx 10$ for $Kmax = 20$ and $Kmax = 100$ respectively. However for slots, we find that the average shots per class are $\approx 1.3$ and $\approx 3$ for $Kmax = 20$ and $Kmax = 100$ respectively. We conjecture that as the shots per class for slots are much lesser in comparison to that of intents, it results in smaller improvements when compared to intents in the joint IC/SF setting.

## E Compute

For all our experiments we primarily use a V100-16GB GPU. For meta-training on ATIS for $Kmax = 100$ with data augmentation, we use V100-32GB GPU due to increased memory requirements.

## F Note on Data Augmentation Techniques

In our paper, we investigate only a limited number of data augmentation techniques specific to natural language processing. We note that in the recent years, a wide variety of augmentation techniques for NLP has been developed (See (Feng et al., 2021) for a good overview). However, we choose EDA, backtranslation and use a dictionary based `slot-list values` in our experiments due to it's inherent simplicity which can enable easy integration with existing meta-learning methods. Designing and adapting existing augmentation techniques to meta-learning is a future direction of research.