Learning to Learn to Disambiguate: Meta-Learning for Few-Shot Word Sense Disambiguation

Nithin Holla
ILLC, University of Amsterdam
nithin.holla7@gmail.com

Pushkar Mishra
Facebook AI
pushkarmishra@fb.com

Helen Yannakoudakis
Dept. of Informatics, King's College London
helen.yannakoudakis@kcl.ac.uk

Ekaterina Shutova
ILLC, University of Amsterdam
e.shutova@uva.nl

Abstract

Deep learning methods typically rely on large amounts of annotated data and do not generalize well to few-shot learning problems where labeled data is scarce. In contrast to human intelligence, such approaches lack versatility and struggle to learn and adapt quickly to new tasks. Meta-learning addresses this problem by training on a large number of related tasks such that new tasks can be learned quickly using a small number of examples. We propose a meta-learning framework for few-shot word sense disambiguation (WSD), where the goal is to disambiguate unseen words from only a few labeled instances. Meta-learning approaches have so far been typically tested in an N-way, K-shot classification setting where each task has N classes with K examples per class. Owing to its nature, WSD deviates from this controlled setup and requires the models to handle a large number of highly unbalanced classes. We extend several popular meta-learning approaches to this scenario, and analyze their strengths and weaknesses in this new challenging setting.

1 Introduction

Natural language is inherently ambiguous, with many words having a range of possible meanings. Word sense disambiguation (WSD) is a core task in natural language understanding where the goal is to associate words with their correct contextual meaning from a pre-defined sense inventory. WSD has been shown to improve downstream tasks such as machine translation (Chan et al., 2007) and information retrieval (Zhong and Ng, 2012). However, it is considered an AI-complete problem (Navigli, 2009) — it requires an intricate understanding of language, as well as real-world knowledge.

Approaches to WSD typically rely on (semi-) supervised learning (Zhong and Ng, 2010; Melamud et al., 2016; Kågebäck and Salomonsson, 2016; Yuan et al., 2016) or are knowledge-based (Lesk, 1986; Agirre et al., 2014; Moro et al., 2014). While supervised methods generally outperform knowledge-based ones (Raganato et al., 2017a), they require data manually annotated with word senses, which are expensive to produce. Supervised approaches also tend to learn a classification model for each word independently; however, this can perform poorly on words that have a limited amount of annotated data. Yet, alternatives that involve a single supervised model for all words (Raganato et al., 2017b) do not adequately solve the problem for rare words (Kumar et al., 2019). Humans, on the other hand, have a remarkable ability to learn from just a handful of examples (Lake et al., 2015). Modern deep learning methods, on the contrary, require large amounts of labeled data for training. Transfer learning (Caruana, 1993) has been proposed as a way to improve the models' data efficiency by transferring features between tasks. However, it still fails to generalize to new tasks in the absence of a considerable amount of task-specific data for fine-tuning (Yogatama et al., 2019).

Meta-learning, commonly referred to as *learning to learn* (Schmidhuber, 1987; Bengio et al., 1991; Thrun and Pratt, 1998), is an alternative learning paradigm that draws on previous experience in order to learn and adapt to new tasks quickly: the model is trained on a number of related tasks such that it can solve unseen tasks using a small number of training examples. A typical meta-learning setup consists of two components: a *learner* that adapts to each task from a small amount of training data pertaining to the task; and a *meta-learner* that guides the learner by acquiring knowledge that is common across all tasks.

Meta-learning has recently emerged as a promising approach to few-shot learning. It has achieved success in computer vision – image classification

(Triantafillou et al., 2019), image segmentation (Hendryx et al., 2019), image synthesis (Fontanini et al., 2019), tracking (Wang et al., 2020) – and reinforcement learning (Wang et al., 2016; Duan et al., 2016; Alet et al., 2020). As of recently, it has also started making its way to NLP – for sentence-level semantic tasks (Dou et al., 2019; Bansal et al., 2019), machine translation (Gu et al., 2018), relation classification (Obamuyide and Vlachos, 2019b), and text classification (Yu et al., 2018).

In this paper, we present a meta-learning framework for WSD. We propose models that learn to rapidly disambiguate new words with a small number of labeled examples. To the best of our knowledge, this is the first approach to few-shot WSD using meta-learning. Owing to its nature, WSD exhibits inter-word dependencies within sentences, has a large number of classes, and inevitable class imbalances; all of which present new challenges compared to the controlled setup in most current meta-learning approaches. To address these challenges we extend three popular meta-learning approaches to this task: Prototypical Networks (Snell et al., 2017), Model-Agnostic Meta-Learning (MAML) (Finn et al., 2017) and a hybrid thereof — ProtoMAML (Triantafillou et al., 2019). We investigate meta-learning using three underlying model architectures, namely recurrent networks, fully-connected networks/multi-layer perceptrons (MLP) and transformers (Vaswani et al., 2017), and experiment with varying number of sentences available for task-specific fine-tuning. We evaluate the model's rapid adaptation ability by testing on a set of new, unseen words, thus demonstrating that the model is able to learn new word senses from a small number of examples.

As there are no few-shot WSD datasets available for our task formulation, we create a few-shot version of a publicly available WSD dataset for our experiments. We release our code as well as the scripts used to generate our few-shot dataset setup to further facilitate research in the field [1].

2 Background and Related Work

2.1 Meta-learning

In contrast to "traditional" machine learning approaches, meta-learning involves a different paradigm known as *episodic learning*. The training set and test set in meta-learning are called *meta-training set* ($\mathcal{D}_{meta-train}$) and *meta-test set* ($\mathcal{D}_{meta-test}$) respectively. Both sets consist of *episodes* rather than individual data points. Each episode constitutes a task $\mathcal{T}_i$, comprising a small number of training examples for adaptation, called the *support set* $\mathcal{D}^{(i)}_{support}$, and a separate set of test examples for evaluation, called the *query set* $\mathcal{D}^{(i)}_{query}$. A typical setup for meta-learning is the balanced N-way, K-shot setting where each episode has N classes with K examples per class in its support set.

Meta-learning algorithms are broadly categorized into three types: *metric-based* (Koch et al., 2015; Vinyals et al., 2016; Sung et al., 2017; Snell et al., 2017), *model-based* (Santoro et al., 2016; Munkhdalai and Yu, 2017), and *optimization-based* (Ravi and Larochelle, 2017; Finn et al., 2017; Nichol et al., 2018). Metric-based methods first embed the examples in each episode into a high-dimensional space typically using a neural network. Next, they obtain the probability distribution over labels for all the query examples based on a kernel function that measures the similarity with the support examples. Model-based approaches try to achieve rapid learning directly through their architectures. They typically employ external memory so as to remember key examples encountered in the past and thus avoid forgetting. Optimization-based approaches explicitly include generalizability in their objective function and optimize for the same.

In this paper, we experiment with metric-based and optimization-based approaches, as well as a hybrid thereof.

2.2 Meta-learning in NLP

Meta-learning in NLP is still in its nascent stages. Gu et al. (2018) apply meta-learning to the problem of neural machine translation where they meta-train on translating high-resource languages to English and meta-test on translating low-resource languages to English. Obamuyide and Vlachos (2019b) use meta-learning for relation classification whereas Obamuyide and Vlachos (2019a) utilize meta-learning in a lifelong learning setting of relation extraction. Chen et al. (2019) consider relation learning by using meta-learning to do few-shot link prediction in knowledge graphs. Dou et al. (2019) perform meta-training on certain high-resource tasks from the GLUE benchmark (Wang et al., 2018) and meta-test on certain low-resource tasks from the same benchmark. Bansal

[1] PlaceholderURL

et al. (2019) propose a softmax parameter generator component that can enable a varying number of classes in the meta-training tasks. They choose the tasks in GLUE along with SNLI (Bowman et al., 2015) for meta-training, and use entity typing, relation classification, sentiment classification, text categorization, and scientific NLI as the test tasks. Meta-learning has also been explored for few-shot text classification (Yu et al., 2018; Geng et al., 2019; Jiang et al., 2018; Sun et al., 2019). Wu et al. (2019) employ meta-reinforcement learning techniques for multi-label classification, with experiments on entity typing and text classification. Hu et al. (2019) adopt meta-learning to learning good representations of out-of-vocabulary words by framing it as a regression task.

2.3 Supervised WSD

Early supervised systems for WSD relied on handcrafted features extracted from the context words to train a machine learning classifier (Lee and Ng, 2002; Navigli, 2009; Zhong and Ng, 2010). Word embeddings were later used as features to train classifiers (Taghipour and Ng, 2015; Rothe and Schütze, 2015; Iacobacci et al., 2016). With the rise of deep learning, LSTM-based (Hochreiter and Schmidhuber, 1997) architectures were employed (Melamud et al., 2016; Kågebäck and Salomonsson, 2016; Yuan et al., 2016). While most work trained individual models per word, Raganato et al. (2017b) designed a single LSTM architecture to disambiguate all words, with the number of output units being equal to the sum of the number of words in the vocabulary and the total number of senses. Peters et al. (2018) performed WSD by nearest neighbour matching with contextualized ELMo (Peters et al., 2018) embeddings. Hadiwinoto et al. (2019) used pre-trained contextualized representations from BERT (Devlin et al., 2019) as features. Huang et al. (2019) fine-tune BERT for WSD while also incorporating sense definitions from WordNet (Miller et al., 1990) to obtain the current state-of-the-art F1 score of 77% on the benchmark by Raganato et al. (2017a).

3 Task and Dataset

We treat WSD as a few-shot word-level classification problem. As different words may have a different number of senses (classes) and sentences may have multiple ambiguous words, the standard setting of N-way, K-shot does not hold here. Specifically, different episodes can have a different number of classes and a varying number of examples per class – a setting which is considered to be more realistic (Triantafillou et al., 2019).

Dataset We use the SemCor corpus (Miller et al., 1994) manually annotated with senses from the New Oxford American Dictionary (NOAD) by Yuan et al. (2016),[2] which is one of the largest sense-annotated English corpora, with $37,176$ annotated sentences. The dataset is typically used only for model training (Raganato et al., 2017a) and thus does not include a train/validation/test split. We group the sentences in the corpus according to which word is to be disambiguated and randomly divide the words into disjoint meta-train, meta-validation and meta-test sets with a $60:15:25$ split. We consider three different settings with $|S|$ = 8, 16 and 32 sentences in the support set. A sentence may contain multiple word-level annotations. The statistics of the resulting dataset are shown in Table 1.

Episode generation For the meta-validation and meta-test sets, each episode corresponds to the task of disambiguating a single word. Thus, each episode has sentences containing annotations for a given word. The number of sentences in the support set is either 8, 16 or 32, whereas we allow the number of sentences in the query set to be equal to or less than each of these respectively since they are only used for evaluation. While splitting the sentences into support and query sets, we ensure that senses in the query set are already seen in the support set and we do not consider words with only one sense in its query set. Furthermore, we discard words that have fewer than a total of $|S|+1$ sentences since they cannot form a complete episode. For the meta-training set, both the support and query sets have 8, 16 or 32 sentences. Initial experiments with one-word-per-episode in the meta-training set yielded poor results due to an insufficient number of total episodes. Class imbalances and the presence of very frequent senses further hindered performance. To ameliorate these issues and design a suitable setup for meta-learning, we instead create training episodes with multiple annotated words in them. Specifically, each episode consists of 4 sampled words $\{s_j\}_{j=1}^{4}$ and $min(4, \nu(s_j))$ senses for each of those words,

[2] https://github.com/google-research-datasets/word_sense_disambigation_corpora

Support sentences	Split	No. of words	No. of episodes	No. of unique sentences	Average no. of senses
8	Meta-training	985	10000	27640	2.96
	Meta-validation	167	167	2303	3.20
	Meta-test	264	264	3561	3.28
16	Meta-training	799	10000	27973	3.07
	Meta-validation	146	146	3651	3.53
	Meta-test	197	197	4918	3.58
32	Meta-training	580	10000	27046	3.34
	Meta-validation	84	84	4051	3.94
	Meta-test	129	129	5836	3.52

Table 1: Statistics of our few-shot WSD dataset.

where $\nu(s_j)$ is the number of senses for word s_j. Sentences containing these senses are then sampled for the support and query sets such that the classes are as balanced as possible. Therefore, each task in the meta-training set is the disambiguation of 4 words between up to 16 senses. The labels for the senses are shuffled across episodes, i.e., one sense can have a different label when sampled in another episode. This is key in meta-learning as it prevents memorization (Yin et al., 2019). The advantage of our approach for constructing meta-training episodes is that it allows for generating a combinatorially large number of tasks that the model can be trained on. Herein, we use a total number of $10,000$ meta-training episodes.

4 Methods

All of our models consist of three parts: an encoder that takes all the words in a sentence as input and produces a representation for each of them, a hidden linear layer that projects the word representations to another space, and an output linear layer that produces the probability distribution over senses. The encoder and the hidden layer are shared across all tasks – we denote this block as $f_{\boldsymbol{\theta}}$ with shared parameters $\boldsymbol{\theta}$. The output layer is randomly initialized for each task $\mathcal{T}_i$ – we denote this as g_{ϕ_i} with parameters $\boldsymbol{\phi}_i$.

4.1 Model Architectures

We experiment with three different encoders: a single-layer bidirectional GRU (Cho et al., 2014) with GloVe embeddings (Pennington et al., 2014) as input that are not fine-tuned; ELMo (Peters et al., 2018) embeddings that are not fine-tuned (reducing the whole network to an MLP); and $\text{BERT}_{\text{BASE}}$ (Devlin et al., 2019) that is fine-tuned. We do not fine-tune ELMo but fine-tune BERT and therefore we work with two different resulting architectures – the MLP and transformer. The architecture of our three different models – GloVe+GRU, ELMo+MLP and BERT – is shown in Figure 1. The shared block $f_{\boldsymbol{\theta}}$ is meta-learned whereas the task-specific layer $g_{\boldsymbol{\phi}_i}$ is independently learned for each task $\mathcal{T}_i$.

4.2 Meta-learning Methods

4.2.1 Prototypical Networks

Proposed by Snell et al. (2017), Prototypical Networks is a metric-based approach making use of the idea of clustering as well as nearest neighbor classification. It consists of an embedding network $f_{\boldsymbol{\theta}}$ parameterized by $\boldsymbol{\theta}$ that is used to produce a prototype vector for every class as the mean vector of the embeddings of all the support data points for that class. Suppose S_c denotes the subset of the support set containing examples from class $c \in C$, the prototype $\boldsymbol{\mu_c}$ is:

$$\boldsymbol{\mu_c} = \frac{1}{|S_c|} \sum_{(x_i, y_i) \in S_c} f_{\boldsymbol{\theta}}(x_i)$$

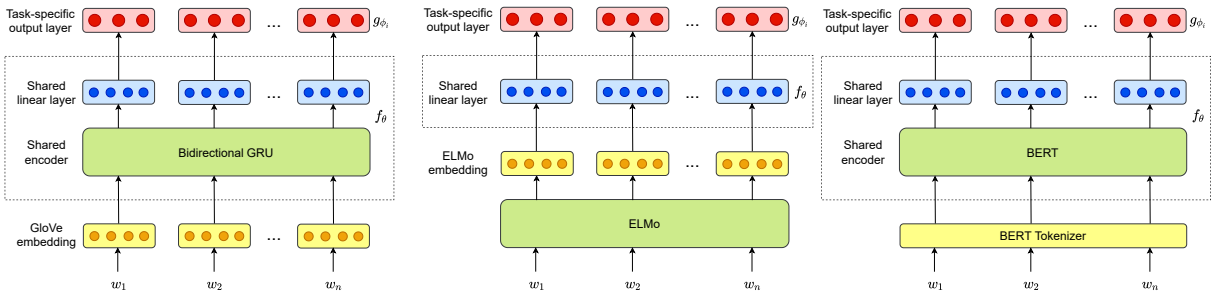
Given a distance function d defined on the embedding space, the distribution over classes for a query point x is calculated as a softmax over negative distances to the prototypes:

$$p(y = c|x) = \frac{\exp(-d(f_{\boldsymbol{\theta}}(x), \boldsymbol{\mu_c}))}{\sum_{c' \in C} \exp(-d(f_{\boldsymbol{\theta}}(x), \boldsymbol{\mu'_c}))} \quad (1)$$

The method is applicable to any distance function so long as it is differentiable. The training loss is the negative log likelihood of the true class c^*:

$$J(\boldsymbol{\theta}) = -\log p(y = c^*|x)$$

We generate the prototypes (one per sense) from the output of the shared block $f_{\boldsymbol{\theta}}$ for the support



(a) Bi-GRU encoder with GloVe input. (b) MLP with ELMo input. (c) Entire BERT model as encoder.

Figure 1: Model architecture showing the shared encoder, the shared linear layer and the task-specific linear layer. The inputs are words $w_1, w_2, ..., w_n$ of a sentence.

examples. The probability distribution over senses for the query examples is obtained as in Equation 1. Thus, we do not specifically use g_{ϕ_i} here. Parameters $\boldsymbol{\theta}$ are updated after every episode using the Adam optimizer (Kingma and Ba, 2015):

$$\boldsymbol{\theta} \leftarrow \text{Adam}(\mathcal{L}^q_{\mathcal{T}_i}, \boldsymbol{\theta}, \beta) \tag{2}$$

where $\mathcal{L}^q_{\mathcal{T}_i}$ is the cross-entropy loss on the query set and β is the *meta learning rate*.

4.2.2 Model-Agnostic Meta-Learning (MAML)

MAML is a purely optimization-based approach proposed by Finn et al. (2017) and designed for the N-way, K-shot setting. The optimization goal is to train a model's initial parameters such that it can perform well on a new task after only a few gradient steps on a small amount of data from that new task. In other words, it seeks to build internal representations that are suitable to many related tasks so that a new task can be learned by fine-tuning on a small number of examples. During meta-training, tasks are drawn from a distribution of tasks $p(\mathcal{T})$. The model's parameters are adapted from $\boldsymbol{\theta}$ to a task $\mathcal{T}_i$ using $D^{(i)}_{support}$ to yield $\boldsymbol{\theta}'_i$. The update is performed using one or several steps of gradient descent. This step is referred to as inner-loop optimization. With m gradient steps, the update is:

$$\boldsymbol{\theta}'_i = U(\mathcal{L}^s_{\mathcal{T}_i}, \boldsymbol{\theta}, \alpha, m), \tag{3}$$

where U is an optimizer such as SGD, α is the inner-loop learning rate and $\mathcal{L}^s_{\mathcal{T}_i}$ is the loss for the task computed on $D^{(i)}_{support}$. Thus, each task $\mathcal{T}_i$ has an updated model $f_{\boldsymbol{\theta}'_i}$. The meta-objective is to have $f_{\boldsymbol{\theta}'_i}$ generalize well across tasks from $p(\mathcal{T})$, i.e.:

$$\begin{aligned} J(\boldsymbol{\theta}) &= \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}^q_{\mathcal{T}_i}(f_{\boldsymbol{\theta}'_i}) \\ &= \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}^q_{\mathcal{T}_i}(f_{U(\mathcal{L}^s_{\mathcal{T}_i}, \boldsymbol{\theta}, \alpha, m)}). \end{aligned}$$

To achieve generalization, the losses $\mathcal{L}^q_{\mathcal{T}_i}$ are computed on $D^{(i)}_{query}$. The optimization is over $\boldsymbol{\theta}$ even though the losses are obtained from the updated parameters $\boldsymbol{\theta}'_i$, which effectively optimizes for the model's initial parameters so that it can undergo a few steps of gradient descent and still perform well. The meta-optimization, also called outer-loop optimization, does the update with the outer-loop learning rate β:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \beta \nabla_{\boldsymbol{\theta}} \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}^q_{\mathcal{T}_i}(f_{\boldsymbol{\theta}'_i})$$

It can be seen that the meta-optimization involves computing second-order gradients, i.e., the backward pass works through the update step in Equation 3, resulting in a computationally expensive process. Finn et al. (2017) propose a first-order approximation, called FOMAML, which ignores the contribution from second-order terms. It computes the gradients with respect to the updated parameters $\boldsymbol{\theta}'_i$ rather than the initial parameters $\boldsymbol{\theta}$. The outer-loop optimization step thus reduces to:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \beta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \nabla_{\boldsymbol{\theta}'_i} \mathcal{L}^q_{\mathcal{T}_i}(f_{\boldsymbol{\theta}'_i})$$

FOMAML does not generalize outside the N-way, K-shot setting, since it assumes a fixed number of classes across tasks. We therefore extend it with output parameters $\boldsymbol{\phi}_i$ that are adapted per task.

During the inner-loop for each task, the optimization is done as follows:

$$\boldsymbol{\theta}'_i, \boldsymbol{\phi}'_i \leftarrow \text{SGD}(\mathcal{L}^s_{\mathcal{T}_i}, \boldsymbol{\theta}, \boldsymbol{\phi}_i, \alpha, \gamma, m) \quad (4)$$

where $\mathcal{L}^s_{\mathcal{T}_i}$ is the cross-entropy loss on the support set, α and γ are the learning rates for the shared block and output layer respectively, and m is the number of update steps. We introduce different learning rates for the shared block and the output layer — the output layer is randomly initialized per task and thus needs to learn aggressively, whereas the shared block already has past information and can thus learn slower. We refer to α as the *learner learning rate* and γ as the *output learning rate*. The outer-loop optimization uses Adam:

$$\boldsymbol{\theta} \leftarrow \text{Adam}\left(\sum_i \mathcal{L}^q_{\mathcal{T}_i}(\boldsymbol{\theta}'_i, \boldsymbol{\phi}'_i), \beta\right) \quad (5)$$

where the gradients of the query cross-entropy losses $\mathcal{L}^q_{\mathcal{T}_i}$ are computed with respect to the updated parameters $\boldsymbol{\theta}'_i$, β is the *meta learning rate*, and the sum over i is for all tasks in the batch of tasks considered.

4.2.3 ProtoMAML

Snell et al. (2017) show that if Euclidean distance is used, Prototypical Networks are equivalent to a linear model with a particular parameterization. The distance can be expressed as:

$$-||f_{\boldsymbol{\theta}}(x) - \boldsymbol{\mu_c}||^2 = -f_{\boldsymbol{\theta}}(x)^T f_{\boldsymbol{\theta}}(x) + 2\boldsymbol{\mu_c}^T f_{\boldsymbol{\theta}}(x) - \boldsymbol{\mu_c}^T \boldsymbol{\mu_c}$$

The first term is constant with respect to class c, so it does not affect the softmax probabilities and can thus be dropped:

$$2\boldsymbol{\mu_c}^T f_{\boldsymbol{\theta}}(x) - \boldsymbol{\mu_c}^T \boldsymbol{\mu_c} = \boldsymbol{w_c}^T f_{\boldsymbol{\theta}}(x) + b_c$$
$$\boldsymbol{w_c} = 2\boldsymbol{\mu_c}, b_c = -\boldsymbol{\mu_c}^T \boldsymbol{\mu_c} \quad (6)$$

where $\boldsymbol{w_c}$ and b_c are the weights and biases for the output unit corresponding to class c. Triantafillou et al. (2019) combine the strengths of Prototypical Networks and MAML by initializing the final layer of the learner classifier in each episode with these Prototypical Network-equivalent weights and biases and continue to learn with MAML, proposing thus a hybrid approach referred to as ProtoMAML. While updating $\boldsymbol{\theta}$, they allow the gradients to flow through the linear layer initialization. In a similar manner, using FOMAML would yield ProtoFOMAML.

Here, too, we construct the prototypes from the output from $f_{\boldsymbol{\theta}}$ for the support examples. The output layer parameters $\boldsymbol{\phi}_i$ are initialized as per Equation 6. The learning then proceeds as in (FO)MAML, i.e., inner-loop optimization as in Equation 4 and outer-loop optimization as in Equation 5; the only difference being that γ need not be too high owing to the good initialization. Proto(FO)MAML thus supports a varying number of classes per task.

4.3 Baseline Methods

Majority sense baseline This baseline predicts the sense that occurs with the highest frequency in the support set. Hereafter, we refer to it as **MajoritySenseBaseline**.

Nearest neighbor classifier This model identifies the sense of a query instance as the sense of its nearest neighbor from the support set in terms of cosine distance. We perform nearest neighbor matching with the ELMo embeddings of the words as well as with the corresponding BERT outputs but not with GloVe embeddings since they are the same for all senses. We refer to this baseline as **NearestNeighbor**.

Non-episodic baseline This baseline is a single model that is trained on all tasks without any distinction between them – it treats the support and query sets as mini-batches. The output layer is thus not task-dependent and the number of output units is equal to the total number of senses in the dataset. The softmax at the output layer is taken only over the relevant classes within the mini-batch. Instead of $\boldsymbol{\phi}_i$ per task, we now have a single $\boldsymbol{\phi}$. During training, the parameters are updated per mini-batch as follows:

$$\boldsymbol{\theta}, \boldsymbol{\phi} \leftarrow \text{Adam}(\mathcal{L}_{\mathcal{T}_i}, \boldsymbol{\theta}, \boldsymbol{\phi}, \alpha)$$

where α is the learning rate. During the meta-testing phase, we independently fine-tune the trained model on the support sets of each of the tasks (in an episodic fashion) as follows:

$$\boldsymbol{\theta}'_i, \boldsymbol{\phi}'_i \leftarrow \text{SGD}(\mathcal{L}_{\mathcal{T}_i}, \boldsymbol{\theta}, \boldsymbol{\phi}, \alpha, \gamma, m)$$

where the loss is computed on the support examples, α is the *learner learning rate* as before and γ is the *output learning rate*. We use SGD for fine-tuning because we only have m update steps

and do not need to track gradients over multiple episodes. We refer to this model as **NE-Baseline**.

Episodic fine-tuning baseline In addition to our meta-learning methods, we also include a variant that only performs meta-testing starting from a randomly initialized model. This is equivalent to training from scratch on the support examples of each episode. We prepend the prefix **EF-** to the meta-learning methods to denote this baseline variant.

5 Experiments and Results

5.1 Experimental setup

We use the meta-validation set to choose the best hyperparameters for the models. The chosen evaluation metric is the average of the macro F1 scores across all words in the meta-validation set. We report the same metric on the meta-test set. We employ early stopping by terminating training if F1 does not improve on the meta-validation set over two epochs. The size of the hidden state in GloVe+GRU is 256, and the size of the shared linear layer is 64, 256 and 192 for the GloVe+GRU, ELMo+MLP and BERT models respectively. The shared linear layer's activation function is *tanh* for GloVe+GRU, and *ReLU* for ELMo+MLP and BERT. For FOMAML, ProtoFOMAML and ProtoMAML, the batch size is set to 16 tasks. A detailed specification of all the hyperparameters is provided in Appendix A.1. The output layer in the meta-learning methods is initialized anew in every episode and every epoch, whereas in the NE-Baseline it has a fixed number of 5612 units, which is the total number of senses in our dataset. Our implementation is based on PyTorch (Paszke et al., 2019) with the MAML variants implemented using the `higher` package (Grefenstette et al., 2019).

5.2 Results

In Table 2, we report macro F1 scores averaged over all words in the meta-test set. We report the means and standard deviations from five independent runs for every model and every value of $|S|$. We note that the results are not directly comparable across $|S|$ setups as, by their formulation, they involve different meta-test episodes.

GloVe+GRU In Table 2, it can be seen that all the meta-learning methods perform better than their EF counterparts, indicating successful utilization of the meta-training set. However, FOMAML fails to outperform NE-Baseline as well as the EF versions of the other meta-learning models. Interestingly, solely running meta-testing is better than fine-tuning the NE-Baseline model which shows that the latter does not effectively transfer knowledge from the meta-training set. ProtoNet, a rather simple metric-based approach, is the best-performing model across all three setups of $|S|$. It even surpasses ProtoFOMAML which incorporates the strength of ProtoNet into FOMAML.

ELMo+MLP The scores for the nearest neighbor classifier, the baseline and the EF methods are much higher compared to GloVe-based models which can be attributed to the input embeddings being contextualized. ProtoNet and ProtoFOMAML still produce improvements over their EF counterparts by utilizing the meta-training set. Like before, FOMAML performs poorly. The difference between ProtoNet and ProtoFOMAML is now smaller, with the latter achieving the best performance for $|S| = 8, 16$ and the former for $|S| = 32$.

BERT The F1 scores for all the BERT-based models are higher than the previous architectures, except for NE-Baseline and FOMAML that now have a lower performance. In line with the earlier observations, FOMAML is comparatively weak. BERT-based ProtoNet is overall the best performing model and outperforms all other approaches for all values of $|S|$. Overall, across architectures, we see that NE-Baseline and FOMAML consistently underperform whereas ProtoNet is often the most effective approach.

Effect of second-order gradients In order to investigate the effect of including second-order gradients in optimization-based meta-learning methods, we further experiment with ProtoMAML, given that ProtoFOMAML performed considerably better than FOMAML. In Table 3, we report the F1 scores alongside ProtoNet and ProtoFOMAML when using GloVe+GRU and ELMo+MLP; however, we exclude the BERT variant (fine-tuned) due to its high computational cost. From the results, we can observe that second-order gradients lead to improved scores compared to ProtoFOMAML in all cases. The improvements are however less than 2%, indicating the effectiveness of the first-order approximation. With GloVe+GRU and $|S| = 8$, ProtoMAML outperforms ProtoNet while ProtoFOMAML does not. With ELMo+MLP and $|S| = 8, 16$, both the first and second-order methods outperform ProtoNet. However, for all $|S|$

Embedding/ Encoder	Method	Average macro F1 score $\|S\| = 8$	$\|S\| = 16$	$\|S\| = 32$
-	MajoritySenseBaseline	0.259	0.264	0.261
GloVe+GRU	NearestNeighbor	–	–	–
	NE-Baseline	0.507 ± 0.005	0.479 ± 0.004	0.451 ± 0.009
	EF-ProtoNet	0.539 ± 0.009	0.538 ± 0.003	0.562 ± 0.005
	EF-FOMAML	0.341 ± 0.002	0.321 ± 0.004	0.303 ± 0.005
	EF-ProtoFOMAML	0.529 ± 0.010	0.540 ± 0.004	0.553 ± 0.009
	ProtoNet	**0.601 ± 0.003**	**0.633 ± 0.008**	**0.654 ± 0.004**
	FOMAML	0.418 ± 0.005	0.392 ± 0.007	0.375 ± 0.005
	ProtoFOMAML	0.599 ± 0.005	0.617 ± 0.004	0.627 ± 0.004
ELMo+MLP	NearestNeighbor	0.641	0.645	0.654
	NE-Baseline	0.640 ± 0.012	0.633 ± 0.001	0.614 ± 0.008
	EF-ProtoNet	0.635 ± 0.004	0.661 ± 0.004	0.683 ± 0.003
	EF-FOMAML	0.414 ± 0.006	0.383 ± 0.003	0.352 ± 0.003
	EF-ProtoFOMAML	0.621 ± 0.004	0.623 ± 0.008	0.611 ± 0.005
	ProtoNet	0.688 ± 0.004	0.709 ± 0.006	**0.731 ± 0.006**
	FOMAML	0.589 ± 0.010	0.587 ± 0.012	0.575 ± 0.016
	ProtoFOMAML	**0.689 ± 0.007**	**0.711 ± 0.004**	0.726 ± 0.004
BERT	NearestNeighbor	0.704	0.716	0.741
	NE-Baseline	0.599 ± 0.023	0.539 ± 0.025	0.473 ± 0.015
	EF-ProtoNet	0.655 ± 0.004	0.682 ± 0.005	0.721 ± 0.009
	EF-FOMAML	0.522 ± 0.007	0.450 ± 0.008	0.393 ± 0.002
	EF-ProtoFOMAML	0.662 ± 0.006	0.654 ± 0.009	0.665 ± 0.009
	ProtoNet	**0.750 ± 0.008**	**0.755 ± 0.002**	**0.766 ± 0.003**
	FOMAML	0.550 ± 0.011	0.476 ± 0.010	0.436 ± 0.014
	ProtoFOMAML	0.731 ± 0.004	0.739 ± 0.008	0.744 ± 0.005

Table 2: Average macro F1 scores of the meta-test words.

Embedding/ Encoder	Method	Average macro F1 score $\|S\| = 8$	$\|S\| = 16$	$\|S\| = 32$
GloVe+GRU	ProtoNet	0.601 ± 0.003	**0.633 ± 0.008**	**0.654 ± 0.004**
	ProtoFOMAML	0.599 ± 0.005	0.617 ± 0.004	0.627 ± 0.004
	ProtoMAML	**0.617 ± 0.005**	0.629 ± 0.006	0.633 ± 0.006
ELMo+MLP	ProtoNet	0.688 ± 0.004	0.709 ± 0.006	**0.731 ± 0.006**
	ProtoFOMAML	0.689 ± 0.007	0.711 ± 0.004	0.726 ± 0.004
	ProtoMAML	**0.699 ± 0.006**	**0.722 ± 0.007**	0.729 ± 0.005

Table 3: Average macro F1 scores of the meta-test words for second-order gradient model variants.

setups, BERT-based ProtoNet achieves the highest performance.

5.3 Analysis

Effect of the number of meta-training episodes The total number of possible meta-training episodes that can be generated using our proposed setup is combinatorially large (see Section 3). We now seek to investigate the following: do more episodes always translate to higher performance? In order to answer that question, we plot the average macro F1 score for our best-performing model – ProtoNet with BERT – as the number of meta-training episodes increases (Figure 2). The shaded region shows one standard deviation from the mean, obtained over five runs. Different $|S|$ setups reach peaks at different meta-training data sizes; however, overall, the largest gains in performance come with a minimum of around 4000 episodes.

Effect of number of senses To investigate the relation between the macro F1 score and the number of senses for a word, in Figure 3, we plot the macro F1 scores averaged over words with a given number of senses in the meta-test set, obtained from our best model — ProtoNet with BERT. Overall, we see a trend where the macro F1 score reduces as the number of senses increase. Furthermore, words with a larger number of senses seem to benefit from a larger number of sentences in the support set. For a word with 8 senses, the $|S| = 32$ case becomes roughly a 4-shot problem whereas it is roughly a 2-shot and 1-shot problem for $|S| = 16$ and $|S| = 8$ respectively. In this view, the disambiguation of

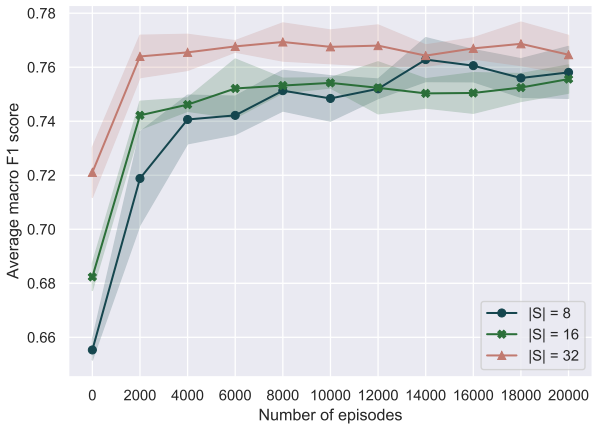


Figure 2: Average macro F1 score of ProtoNet+BERT as the number of meta-training episodes increases.

words with a larger number of senses gets better with $|S|$ due to an increase in the effective number of shots.

Challenging cases In Table 4, we present a set of 10 words with the lowest macro F1 scores (in increasing order of the score) obtained from ProtoNet with GloVe+GRU. We perform the analysis on this model to investigate challenging cases without the effects of, and advantages offered by ELMo and BERT. We note that, for $|S| = 8$, most of the words in the list have predominantly verb senses, showing that they are the most challenging ones to disambiguate. Even for $|S| = 16$, there is a large proportion of verb senses, whereas for $|S| = 32$, the number of such cases drops, indicating that disambiguation of verbs improves as the number of sentences for fine-tuning increases. We present a detailed distribution of macro F1 scores across words in Appendix A.3.

$\|S\| = 8$	$\|S\| = 16$	$\|S\| = 32$
bad	move	independent
work	appearance	gather
give	in	north
clear	green	square
settle	fix	do
bloom	establishment	bond
draw	note	proper
check	drag	pull
break	cup	problem
gather	bounce	language

Table 4: Words with the lowest macro F1 scores for ProtoNet with GloVe+GRU.

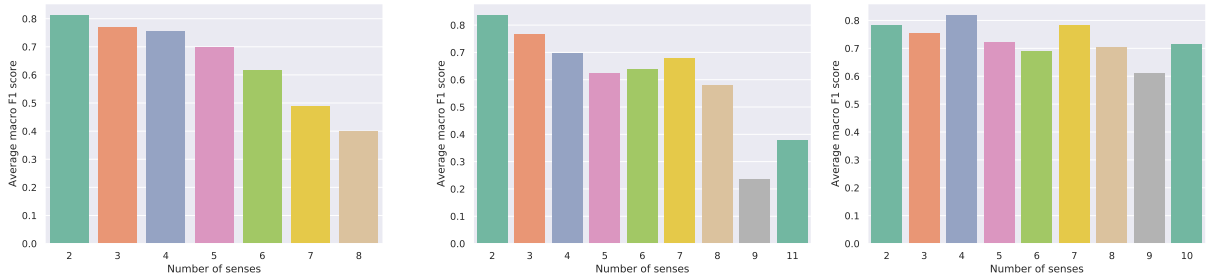
6 Discussion

Our NE-Baseline model trains on all words in the meta-training set followed by fine-tuning on the meta-test words. Our experiments demonstrate that episodic training with meta-learning produces much better few-shot performance than fine-tuning a model trained in a non-episodic fashion, a finding consistent for all $|S|$ setups.

The success of meta-learning is particularly evident in our experiments with GloVe+GRU. GloVe embeddings do not distinguish across the senses of a word and, yet, ProtoNet, ProtoFOMAML and ProtoMAML produce high F1 scores. In fact, their scores come quite close to the nearest neighbor classifier with ELMo embeddings as input, even though ELMo is better able to represent properties of our task. With both ELMo and BERT, the task starts from an improved initialization, owing to their strong pre-training.

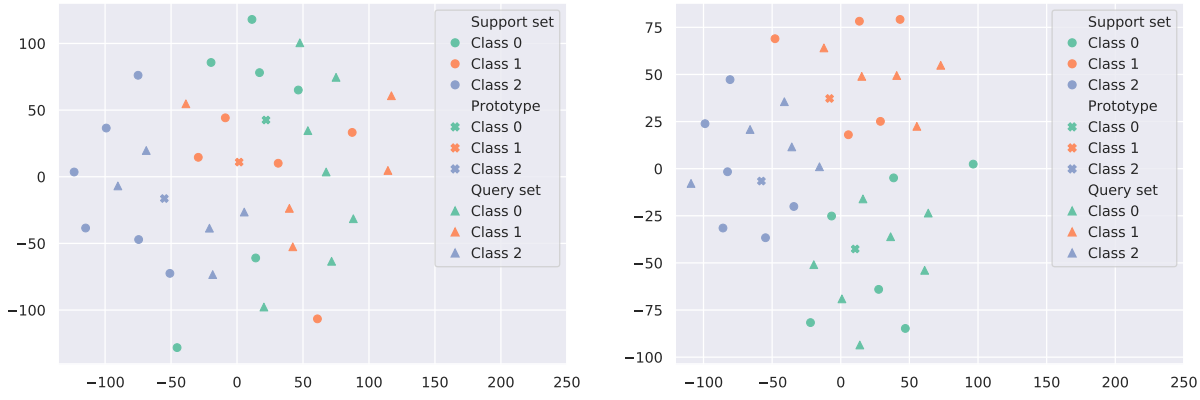
Even though contextualized representations from ELMo and BERT already contain information relevant to our task, integrating them into a meta-learning framework allows these models to further and substantially improve performance. In order to illustrate the advantage that meta-learning brings over contextualized representations, we provide example t-SNE visualizations (van der Maaten and Hinton, 2008) of the original ELMo embeddings and those generated by ProtoNet with ELMo embeddings as input (Figure 4). It can be observed that the representations from ProtoNet are better clustered with respected to the senses compared to the original ELMo representations. ProtoNet thus effectively learns to disambiguate new words — separate the senses into clusters — thereby improving upon using ELMo embeddings. We provide more t-SNE visualizations in Appendix A.4.

Overall, we find that ProtoNet performs better than ProtoFOMAML. This is likely because in ProtoFOMAML, outer-loop backpropagation does not occur through the initialization of the output layer. The gradients are obtained, not with respect to the initial parameters $\boldsymbol{\theta}$, but the updated parameters $\boldsymbol{\theta}'_i$. As a result, $\boldsymbol{\theta}$ is not optimized to explicitly serve as a good output layer initialization. ProtoMAML overcomes this limitation and does better than ProtoNet in some cases. However, this is not a consistent trend, likely because the inner-loop updates do not always improve upon the initial parameters. Sense inference in ProtoNet is similar to some of the traditional approaches to WSD based



(a) $|S| = 8$ (b) $|S| = 16$ (c) $|S| = 32$

Figure 3: Barplot of macro F1 scores averaged over words with a given number of senses.



(a) ELMo embedding - 'field' (b) Word representation - 'field'

Figure 4: t-SNE visualizations comparing ELMo embeddings (left) against word representations generated by ProtoNet with ELMo+MLP (right).

on distances in feature space (Navigli, 2009). A primary difference here is that the representations are optimized for the few-shot setting via episodic training.

Our setup further highlights the weakness of FO-MAML when applied beyond the N-shot, K-way setting. This could be due to the fact that, in each episode, the number of "new" output parameters is much greater than the number of support examples. For a shared linear layer of size 64, a word with 4 senses, for instance, yields $64 \times 4 + 4 = 260$ parameters. Training this number of parameters with 8, 16, 32 examples would likely be sub-optimal. Good output layer initialization is therefore important for effective learning in such scenarios. A similar solution is also used by Bansal et al. (2019), where they design a differentiable parameter generator for the output layer.

We note that, for our models with the GRU encoder, the total number of parameters that need to be trained from scratch is much higher than the number of training examples. Investigating sub-networks with fewer parameters that can perform roughly the same as the original one (e.g., lottery tickets (Frankle and Carbin, 2019; Yu et al., 2020)) is an interesting avenue for future work.

7 Conclusion

Few-shot learning is a key capability for AI to reach human-like performance. Although current meta-learning algorithms do not provide the perfect recipe for few-shot learning, they provide a viable solution when a large number of tasks are available for training. We demonstrated the ability of meta-learning to disambiguate new words when only a handful of labeled examples are available. Considering the typical data scarcity in WSD, we believe that meta-learning can yield a more general disambiguation model than traditional approaches. Interesting avenues to explore further would be whether such a meta-trained model generalizes to disambiguation in a different domain, to a multilingual scenario or to an altogether different yet related task.

References

Eneko Agirre, Oier López de Lacalle, and Aitor Soroa. 2014. Random walks for knowledge-based word sense disambiguation. *Computational Linguistics*, 40(1):57–84.

Ferran Alet, Martin F. Schneider, Tomas Lozano-Perez, and Leslie Pack Kaelbling. 2020. Meta-learning curiosity algorithms.

Trapit Bansal, Rishikesh Jha, and Andrew McCallum. 2019. Learning to few-shot learn across diverse natural language classification tasks.

Y. Bengio, S. Bengio, and J. Cloutier. 1991. Learning a synaptic learning rule. In *IJCNN-91-Seattle International Joint Conference on Neural Networks*, volume ii, pages 969 vol.2–.

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 (EMNLP)*. Association for Computational Linguistics.

Rich Caruana. 1993. Multitask learning: A knowledge-based source of inductive bias. In *Machine Learning, Proceedings of the Tenth International Conference, University of Massachusetts, Amherst, MA, USA, June 27-29, 1993*, pages 41–48. Morgan Kaufmann.

Yee Seng Chan, Hwee Tou Ng, and David Chiang. 2007. Word sense disambiguation improves statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 33–40, Prague, Czech Republic. Association for Computational Linguistics.

Mingyang Chen, Wen Zhang, Wei Zhang, Qiang Chen, and Huajun Chen. 2019. Meta relational learning for few-shot link prediction in knowledge graphs. 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 4217–4226, Hong Kong, China. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

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

Zi-Yi Dou, Keyi Yu, and Antonios Anastasopoulos. 2019. Investigating meta-learning algorithms for low-resource natural language understanding 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 1192–1197, Hong Kong, China. Association for Computational Linguistics.

Yan Duan, John Schulman, Xi Chen, Peter L. Bartlett, Ilya Sutskever, and Pieter Abbeel. 2016. Rl^2: Fast reinforcement learning via slow reinforcement learning.

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, International Convention Centre, Sydney, Australia. PMLR.

Tomaso Fontanini, Eleonora Iotti, Luca Donati, and Andrea Prati. 2019. Metalgan: Multi-domain label-less image synthesis using cgans and meta-learning.

Jonathan Frankle and Michael Carbin. 2019. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Ruiying Geng, Binhua Li, Yongbin Li, Xiaodan Zhu, Ping Jian, and Jian Sun. 2019. Induction networks for few-shot text classification. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3904–3913, Hong Kong, China. Association for Computational Linguistics.

Edward Grefenstette, Brandon Amos, Denis Yarats, Phu Mon Htut, Artem Molchanov, Franziska Meier, Douwe Kiela, Kyunghyun Cho, and Soumith Chintala. 2019. Generalized inner loop meta-learning.

Jiatao Gu, Yong Wang, Yun Chen, Victor O. K. Li, and Kyunghyun Cho. 2018. Meta-learning for low-resource neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3622–3631, Brussels, Belgium. Association for Computational Linguistics.

Christian Hadiwinoto, Hwee Tou Ng, and Wee Chung Gan. 2019. Improved word sense disambiguation using pre-trained contextualized word representations. 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 5297–5306, Hong Kong, China. Association for Computational Linguistics.

Sean M. Hendryx, Andrew B. Leach, Paul D. Hein, and Clayton T. Morrison. 2019. Meta-learning initializations for image segmentation.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Ziniu Hu, Ting Chen, Kai-Wei Chang, and Yizhou Sun. 2019. Few-shot representation learning for out-of-vocabulary words. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4102–4112, Florence, Italy. Association for Computational Linguistics.

Luyao Huang, Chi Sun, Xipeng Qiu, and Xuanjing Huang. 2019. GlossBERT: BERT for word sense disambiguation with gloss knowledge. 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 3509–3514, Hong Kong, China. Association for Computational Linguistics.

Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. 2016. Embeddings for word sense disambiguation: An evaluation study. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 897–907, Berlin, Germany. Association for Computational Linguistics.

Xiang Jiang, Mohammad Havaei, Gabriel Chartrand, Hassan Chouaib, Thomas Vincent, Andrew Jesson, Nicolas Chapados, and Stan Matwin. 2018. Attentive task-agnostic meta-learning for few-shot text classification. In *The Second Workshop on MetaLearning at NeurIPS*.

Mikael Kågebäck and Hans Salomonsson. 2016. Word sense disambiguation using a bidirectional LSTM. In *Proceedings of the 5th Workshop on Cognitive Aspects of the Lexicon (CogALex - V)*, pages 51–56, Osaka, Japan. The COLING 2016 Organizing Committee.

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

Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. 2015. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2. Lille.

Sawan Kumar, Sharmistha Jat, Karan Saxena, and Partha Talukdar. 2019. Zero-shot word sense disambiguation using sense definition embeddings. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5670–5681, Florence, Italy. Association for Computational Linguistics.

Brenden M. Lake, Ruslan Salakhutdinov, and Joshua B. Tenenbaum. 2015. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338.

Yoong Keok Lee and Hwee Tou Ng. 2002. An empirical evaluation of knowledge sources and learning algorithms for word sense disambiguation. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*, pages 41–48. Association for Computational Linguistics.

Michael Lesk. 1986. Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone. In *Proceedings of the 5th Annual International Conference on Systems Documentation*, SIGDOC 86, page 2426, New York, NY, USA. Association for Computing Machinery.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605.

Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. context2vec: Learning generic context embedding with bidirectional LSTM. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 51–61, Berlin, Germany. Association for Computational Linguistics.

George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine Miller. 1990. Wordnet: An on-line lexical database. *International Journal of Lexicography*, 3:235–244.

George A. Miller, Martin Chodorow, Shari Landes, Claudia Leacock, and Robert G. Thomas. 1994. Using a semantic concordance for sense identification. In *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*.

Andrea Moro, Alessandro Raganato, and Roberto Navigli. 2014. Entity linking meets word sense disambiguation: a unified approach. *Transactions of the Association for Computational Linguistics*, 2:231–244.

Tsendsuren Munkhdalai and Hong Yu. 2017. Meta networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2554–2563, International Convention Centre, Sydney, Australia. PMLR.

Roberto Navigli. 2009. Word sense disambiguation: A survey. *ACM Computing Surveys*, 41(2):1–69.

Alex Nichol, Joshua Achiam, and John Schulman. 2018. On first-order meta-learning algorithms. *CoRR*, abs/1803.02999.

Abiola Obamuyide and Andreas Vlachos. 2019a. Meta-learning improves lifelong relation extraction. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pages 224–229, Florence, Italy. Association for Computational Linguistics.

Abiola Obamuyide and Andreas Vlachos. 2019b. Model-agnostic meta-learning for relation classification with limited supervision. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5873–5879, Florence, Italy. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

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

Alessandro Raganato, Jose Camacho-Collados, and Roberto Navigli. 2017a. Word sense disambiguation: A unified evaluation framework and empirical comparison. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 99–110, Valencia, Spain. Association for Computational Linguistics.

Alessandro Raganato, Claudio Delli Bovi, and Roberto Navigli. 2017b. Neural sequence learning models for word sense disambiguation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1156–1167, Copenhagen, Denmark. Association for Computational Linguistics.

Sachin Ravi and Hugo Larochelle. 2017. Optimization as a model for few-shot learning. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Sascha Rothe and Hinrich Schütze. 2015. AutoExtend: Extending word embeddings to embeddings for synsets and lexemes. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1793–1803, Beijing, China. Association for Computational Linguistics.

Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. 2016. Meta-learning with memory-augmented neural networks. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1842–1850, New York, New York, USA. PMLR.

Jurgen Schmidhuber. 1987. Evolutionary principles in self-referential learning. on learning now to learn: The meta-meta-meta...-hook. Diploma thesis, Technische Universitat Munchen, Germany, 14 May.

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

Shengli Sun, Qingfeng Sun, Kevin Zhou, and Tengchao Lv. 2019. Hierarchical attention prototypical networks for few-shot text classification. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 476–485, Hong Kong, China. Association for Computational Linguistics.

Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip H. S. Torr, and Timothy M. Hospedales. 2017. Learning to compare: Relation network for few-shot learning.

Kaveh Taghipour and Hwee Tou Ng. 2015. Semi-supervised word sense disambiguation using word embeddings in general and specific domains. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 314–323, Denver, Colorado. Association for Computational Linguistics.

Sebastian Thrun and Lorien Pratt, editors. 1998. *Learning to Learn*. Kluwer Academic Publishers, USA.

Eleni Triantafillou, Tyler Zhu, Vincent Dumoulin, Pascal Lamblin, Utku Evci, Kelvin Xu, Ross Goroshin, Carles Gelada, Kevin Swersky, Pierre-Antoine Manzagol, and Hugo Larochelle. 2019. Meta-dataset: A

dataset of datasets for learning to learn from few examples.

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 I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

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

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, Brussels, Belgium. Association for Computational Linguistics.

Guangting Wang, Chong Luo, Xiaoyan Sun, Zhiwei Xiong, and Wenjun Zeng. 2020. Tracking by instance detection: A meta-learning approach.

Jane X Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Remi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick. 2016. Learning to reinforcement learn.

Jiawei Wu, Wenhan Xiong, and William Yang Wang. 2019. Learning to learn and predict: A meta-learning approach for multi-label 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 4354–4364, Hong Kong, China. Association for Computational Linguistics.

Mingzhang Yin, George Tucker, Mingyuan Zhou, Sergey Levine, and Chelsea Finn. 2019. Meta-learning without memorization.

Dani Yogatama, Cyprien de Masson d'Autume, Jerome Connor, Tomás Kociský, Mike Chrzanowski, Lingpeng Kong, Angeliki Lazaridou, Wang Ling, Lei Yu, Chris Dyer, and Phil Blunsom. 2019. Learning and evaluating general linguistic intelligence. *CoRR*, abs/1901.11373.

Haonan Yu, Sergey Edunov, Yuandong Tian, and Ari S. Morcos. 2020. Playing the lottery with rewards and multiple languages: lottery tickets in rl and nlp. In *International Conference on Learning Representations*.

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

Dayu Yuan, Julian Richardson, Ryan Doherty, Colin Evans, and Eric Altendorf. 2016. Semi-supervised word sense disambiguation with neural models. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1374–1385, Osaka, Japan. The COLING 2016 Organizing Committee.

Zhi Zhong and Hwee Tou Ng. 2010. It makes sense: A wide-coverage word sense disambiguation system for free text. In *Proceedings of the ACL 2010 System Demonstrations*, pages 78–83, Uppsala, Sweden. Association for Computational Linguistics.

Zhi Zhong and Hwee Tou Ng. 2012. Word sense disambiguation improves information retrieval. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 273–282, Jeju Island, Korea. Association for Computational Linguistics.

A Appendix

A.1 Hyperparameters

We performed hyperparameter tuning for all the models under the $|S| = 16$ setting. We obtain the best hyperparameters on the basis of the average macro F1 score on the meta-validation set. We trained the models with 5 seeds (42, 43, 44, 45, 46) and recorded the mean of the metric from the five runs to decide the best hyperparameters. For $|S| = 8$ and $|S| = 32$, we chose the best hyperparameters obtained from tuning. We employed early stopping with a patience of 2 epochs, i.e., we stop meta-training if the average validation macro F1 score does not improve over 2 epochs. Tuning over all the hyperparameters of our models is prohibitively expensive. Hence, for some of the hyperparameters we chose a fixed value. The size of the shared linear layer is 64, 256 and 192 for the GloVe+GRU, ELMo+MLP and BERT models respectively. The shared linear layer's activation function is tanh for GloVe+GRU and ReLU for ELMo+MLP and BERT. For FOMAML, Proto-FOMAML and ProtoMAML, the batch size is set to 16 tasks. For the BERT models, we perform learning rate warm-up for 100 steps followed by a constant rate. For GloVe+GRU and ELMo+MLP,

Embedding/ Encoder	Method	Output learning rate	Learner learning rate	Meta learning rate	Hidden size	No. of inner-loop updates	Size of shared linear layer
GloVe+GRU	NE-Baseline	1e−1	5e−4	-	256	5	64
	ProtoNet	-	-	1e−3	256	-	64
	FOMAML	1e−1	1e−2	1e−3	256	5	64
	ProtoFOMAML	1e−3	1e−3	1e−3	256	5	64
	ProtoMAML	1e−3	1e−3	1e−3	256	5	64
ELMo+MLP	NE-Baseline	1e−1	1e−3	-	-	7	256
	ProtoNet	-	-	1e−3	-	-	256
	FOMAML	1e−1	1e−2	5e−3	-	7	256
	ProtoFOMAML	5e−3	5e−3	5e−4	-	7	256
	ProtoMAML	1e−3	1e−3	1e−3	-	7	256
BERT	NE-Baseline	1e−1	5e−5	-	-	7	192
	ProtoNet	-	-	1e−6	-	-	192
	FOMAML	1e−1	1e−3	5e−5	-	7	192
	ProtoFOMAML	1e−3	1e−3	1e−4	-	7	192

Table 5: Hyperparameters used for training the models.

Embedding/ Encoder	Method	No. of GPUs used	Approximate training time per epoch
GloVe+GRU	Baseline	1	8 minutes
	ProtoNet	1	8 minutes
	FOMAML	1	15 minutes
	ProtoFOMAML	1	15 minutes
	ProtoMAML	1	9 hours 30 minutes
ELMo+MLP	Baseline	1	55 minutes
	ProtoNet	1	55 minutes
	FOMAML	1	1 hour
	ProtoFOMAML	1	1 hour
	ProtoMAML	1	1 hour 8 minutes
BERT	Baseline	1	35 minutes
	ProtoNet	1	35 minutes
	FOMAML	4	2 hours 35 minutes
	ProtoFOMAML	4	2 hours 35 minutes

Table 6: Approximate training time per epoch.

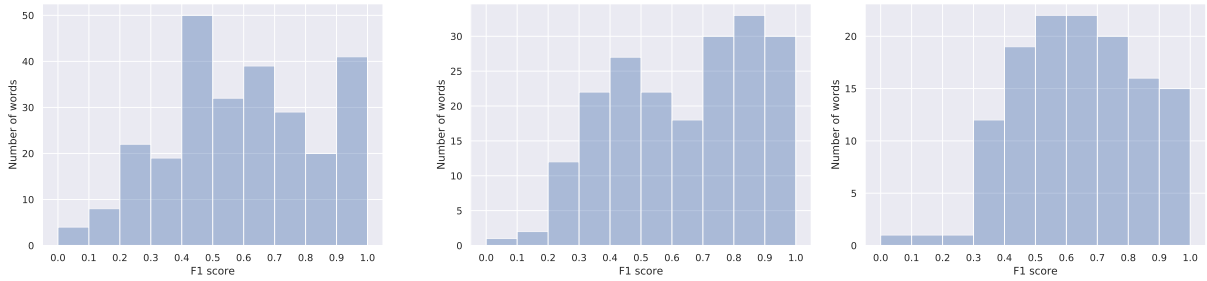
we decay the learning rate by half every 500 steps. We also experimented with two types of regularization – dropout for the inner-loop updates and weight decay for the outer-loop updates – but both of them yielded a drop in performance. The remaining hyperparameters, namely the output learning rate, learner learning rate, meta learning rate, hidden size (only for GloVe+GRU), and number of inner-loop updates were tuned. The best hyperparameters obtained are shown in Table 5.

A.2 Training times

We train all our models on TitanRTX GPUs. Our models vary in the total number of trainable parameters. Thus, the time taken to train each of them varies. To give an idea of how long it takes to train them, we provide an approximate time taken for one epoch for the $|S| = 16$ setup in Table 6. The training time would be slightly lower for $|S| = 8$ and slightly higher for $|S| = 32$. The training time for ProtoMAML with GloVe+GRU is extremely long (second-order derivatives for RNNs with the cuDNN backend was not supported in PyTorch at the time of writing and hence cuDNN had to be disabled).

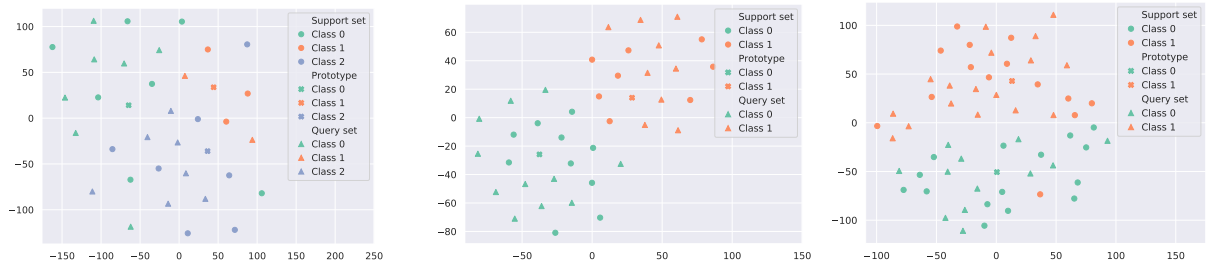
A.3 F1 score distribution

For ProtoNet with GloVe+GRU, we plot the distribution of macro F1 scores across the words in the meta-test set in Figure 5. The distribution is mostly right-skewed with very few words having scores in the range 0 to 0.2.



(a) $|S| = 8$ (b) $|S| = 16$ (c) $|S| = 32$

Figure 5: Distribution of macro F1 scores for ProtoNet with GloVe+GRU.



(a) Word representation - 'favor' (b) Word representation - 'tax' (c) Word representation - 'snow'

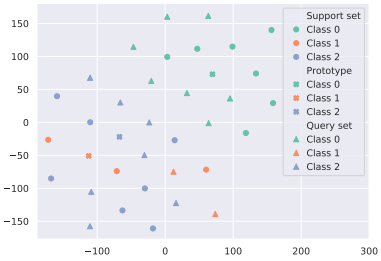
Figure 6: t-SNE visualizations of word representations generated by ProtoNet with GloVe+GRU.

A.4 t-SNE visualizations

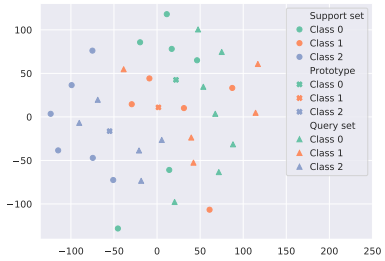
We provide a t-SNE visualization of the word representations generated by $f_{\boldsymbol{\theta}}$ of ProtoNet with GloVe+GRU for three words in the meta-test set in Figure 6. These three words achieved a macro F1 score of 1. Even though the model receives the same embedding for all senses as its input, it manages to separate the senses into clusters on the basis of the output representations. This occurs solely from the support examples without any fine-tuning on them. Moreover, the query examples also seem to be part of the same cluster and lie close to the prototypes.

ELMo embeddings, being contextual, already capture information in how the various senses are represented. In order to compare them against the representations generated by ProtoNet with ELMo+MLP, we again provide t-SNE visualizations. We plot the ELMo embeddings of three words in the meta-test test in Figure 7a, 7b and 7c. We also show the prototypes computed from these embeddings for illustration. For the same three words, we plot the representations obtained from $f_{\boldsymbol{\theta}}$ of ProtoNet with ELMo+MLP in Figure 7d, 7e and 7f. It can be observed that the ELMo embeddings alone are not clustered with respect to the senses. On the other hand, ProtoNet manages to separate the senses into clusters without any form of fine-tuning, which aids in making accurate predictions on the query set.

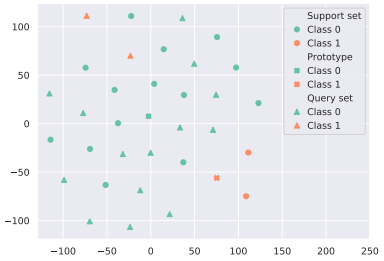
The visualizations of the word representations obtained from ProtoNet with both GloVe+GRU and ELMo+MLP further demonstrate ProtoNet's success in disambiguating new words.



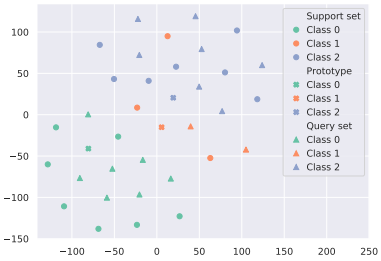
(a) ELMo embedding – 'favor'



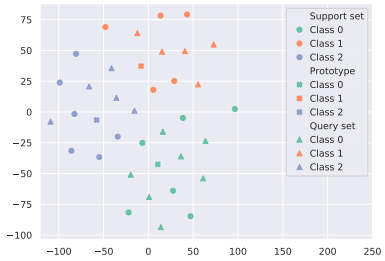
(b) ELMo embedding – 'field'



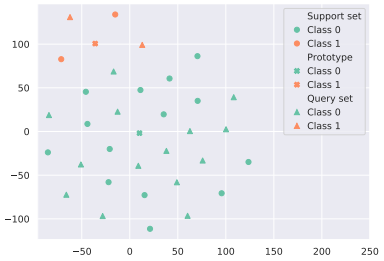
(c) ELMo embedding – 'factor'



(d) Word representation – 'favor'



(e) Word representation – 'field'



(f) Word representation - 'factor'

Figure 7: t-SNE visualizations comparing ELMo embeddings (top) against word representations generated by ProtoNet with ELMo+MLP (bottom).