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

The success of deep learning methods hinges on the availability of large training datasets annotated for the task of interest. In contrast to human intelligence, these methods lack versatility and struggle to learn and adapt quickly to new tasks, where labeled data is scarce. Meta-learning aims to solve this problem by training a model on a large number of few-shot tasks, with an objective to learn new tasks quickly from a small number of examples. In this paper, we propose a meta-learning framework for few-shot word sense disambiguation (WSD), where the goal is to learn 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 the knowledge-based ones (Raganato et al., 2017a), they require data manually annotated with word senses, which are expensive to produce at a large scale. These methods also tend to learn a classification model for each word independently, and hence may 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) still 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). This inspired researchers to investigate techniques that would enable machine learning models to do the same. One such approach is transfer learning (Caruana, 1993), which aims 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, known as *learning to learn* (Schmidhuber, 1987; Bengio et al., 1991; Thrun and Pratt, 1998), is an alternative paradigm that draws on past 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 only a small number of training examples. A typical meta-learning setup consists of two components: a *learner* that adapts to each task from its small training data; 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 (Triantafillou et al., 2020; Fontanini et al., 2020; Hendryx et al., 2019; Wang et al., 2020) and reinforcement learning

Findings of the Association for Computational Linguistics: EMNLP 2020, pages 4517–4533
November 16 - 20, 2020.

(Wang et al., 2016; Duan et al., 2016; Alet et al., 2020). It has also recently made its way into NLP, and has been applied to machine translation (Gu et al., 2018), relation (Obamuyide and Vlachos, 2019b) and text (Yu et al., 2018) classification, and sentence-level semantic tasks (Dou et al., 2019; Bansal et al., 2019).

In this paper, we present the first meta-learning approach to WSD. We propose models that learn to rapidly disambiguate new words from only a few labeled examples. 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 algorithms 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., 2020). We investigate meta-learning using three underlying model architectures, namely recurrent 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 its ability to learn new word senses from a small number of examples.

Since there are no few-shot WSD benchmarks available, we create a few-shot version of a publicly available WSD dataset. We release our code as well as the scripts used to generate our few-shot data setup to facilitate further research.[1]

2 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 and test sets in meta-learning are referred to as *meta-training set* ($\mathcal{D}_{meta\text{-}train}$) and *meta-test set* ($\mathcal{D}_{meta\text{-}test}$) 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 – the *support set* $\mathcal{D}^{(i)}_{support}$, and a separate set of examples for evaluation – 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., 2018; 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 aim to achieve rapid learning directly through their architectures. They typically employ external memory so as to remember key examples encountered in the past. 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 and Obamuyide and Vlachos (2019a) for relation extraction in a lifelong learning setting. Chen et al. (2019) consider relation learning and apply meta-learning to 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 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

[1] https://github.com/Nithin-Holla/MetaWSD

(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) use meta-learning to learn representations of out-of-vocabulary words, framing it as a regression task.

2.3 Supervised WSD

Early supervised learning approaches to WSD relied on hand-crafted features extracted from the context words (Lee and Ng, 2002; Navigli, 2009; Zhong and Ng, 2010). Later work used word embeddings as features for classification (Taghipour and Ng, 2015; Rothe and Schütze, 2015; Iacobacci et al., 2016). With the rise of deep learning, LSTM (Hochreiter and Schmidhuber, 1997) models became popular (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 model with a large number of output units to disambiguate all words. 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. GlossBERT (Huang et al., 2019) incorporated sense definitions from WordNet (Miller et al., 1990) to form context-gloss pairs while fine-tuning BERT for WSD. By taking WordNet's graph structure into account, EWISER (Bevilacqua and Navigli, 2020) achieves the current state-of-the-art F1 score of 80.1% on the benchmark by Raganato et al. (2017a).

3 Task and Dataset

We treat WSD as a few-shot word-level classification problem, where a sense is assigned to a word given its sentential context. As different words may have a different number of senses and sentences may have multiple ambiguous words, the standard setting of N-way, K-shot classification does not hold in our case. Specifically, different episodes can have a different number of classes and a varying number of examples per class – a setting which is more realistic (Triantafillou et al., 2020).

Dataset We use the SemCor corpus (Miller et al., 1994) manually annotated with senses from the New Oxford American Dictionary by Yuan et al. (2016)[2]. With $37,176$ annotated sentences, this is one of the largest sense-annotated English corpora. We group the sentences in the corpus according to which word is to be disambiguated, and then randomly divide the words into disjoint meta-train, meta-validation and meta-test sets with a 60:15:25 split. A sentence may have multiple occurrences of the same word, in which case we make predictions for all of them. We consider four different settings with the support set size $|S|$ = 4, 8, 16 and 32 sentences. The number of distinct words in the meta-training / meta-test sets is $985/270$, $985/259$, $799/197$ and $580/129$ respectively. The detailed statistics of the resulting dataset are shown in Appendix A.1.

Training episodes In the meta-training set, both the support and query sets have the same number of sentences. Our initial experiments using one word per episode during meta-training yielded poor results due to an insufficient number of episodes. To overcome this problem and design a suitable meta-training setup, we instead create episodes with multiple annotated words in them. Specifically, each episode consists of r sampled words $\{z_j\}_{j=1}^{r}$ and $min(\lfloor |S|/r \rfloor, \nu(z_j))$ senses for each of those words, where $\nu(z_j)$ is the number of senses for word z_j. Therefore, each task in the meta-training set is the disambiguation of r words between up to $|S|$ senses. We set $r = 2$ for $|S| = 4$ and $r = 4$ for the rest. Sentences containing these senses are then sampled for the support and query sets such that the classes are as balanced as possible. For example, for $|S| = 8$, we first choose 4 words and 2 senses for each, and then sample one sentence for each word-sense pair. 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., 2020). The advantage of our approach for constructing meta-training episodes is that it allows for generating a combinatorially large number of tasks. Herein, we use a total number of $10,000$ meta-training episodes.

Evaluation episodes For the meta-validation and meta-test sets, each episode corresponds to the task of disambiguating a single word. While splitting the sentences into support and query sets, we ensure that senses in the query set are present

[2]`https://tinyurl.com/wsdcrp`. Note that the corpus does not have a standard train/validation/test split.

in the support set. Furthermore, we only consider words with two or more senses in their query set. The distribution of episodes across different number of senses is shown in Appendix A.1. Note that, unlike the meta-training tasks, our meta-test tasks represent a natural data distribution, therefore allowing us to test our models in a realistic setting.

4 Methods

Our models consist of three components: an encoder that takes the words in a sentence as input and produces a contextualized representation for each of them, a hidden linear layer that projects these 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 $\mathcal{T}_i$ (i.e. episode) – we denote this as $g_{\boldsymbol{\phi}_i}$ with parameters $\boldsymbol{\phi}_i$. $\boldsymbol{\theta}$ is meta-learned whereas $\boldsymbol{\phi}_i$ is independently learned for each task.

4.1 Model Architectures

We experiment with three different architectures: (1) a single-layer bidirectional GRU (Cho et al., 2014) with GloVe embeddings (Pennington et al., 2014) as input that are not fine-tuned; (2) ELMo (Peters et al., 2018) embeddings that are not fine-tuned, followed by an MLP; and (3) $\text{BERT}_{\text{BASE}}$ (Devlin et al., 2019) that is fine-tuned. The architecture of our three different models – GloVe+GRU, ELMo+MLP and BERT – is shown in Figure 1.

4.2 Meta-learning Methods

Prototypical Networks Proposed by Snell et al. (2017), prototypical networks is a metric-based approach. An embedding network $f_{\boldsymbol{\theta}}$ parameterized by $\boldsymbol{\theta}$ 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:

$$\boldsymbol{\mu_c} = \frac{1}{|S_c|} \sum_{\boldsymbol{x}_i \in S_c} f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) \tag{1}$$

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

We generate the prototypes (one per sense) from the output of the shared block $f_{\boldsymbol{\theta}}$ for the support examples. Instead of using $g_{\boldsymbol{\phi}_i}$, we obtain the probability distribution for the query examples based on the distance function. 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*.

Model-Agnostic Meta-Learning (MAML) MAML (Finn et al., 2017) is an optimization-based approach designed for the N-way, K-shot classification setting. The goal of optimization 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. Tasks are drawn from a distribution $p(\mathcal{T})$. The model's parameters are adapted from $\boldsymbol{\theta}$ to a task $\mathcal{T}_i$ using gradient descent on $D^{(i)}_{support}$ to yield $\boldsymbol{\theta}'_i$. 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 $D^{(i)}_{support}$. The meta-objective is to have $f_{\boldsymbol{\theta}'_i}$ generalize well across tasks from $p(\mathcal{T})$:

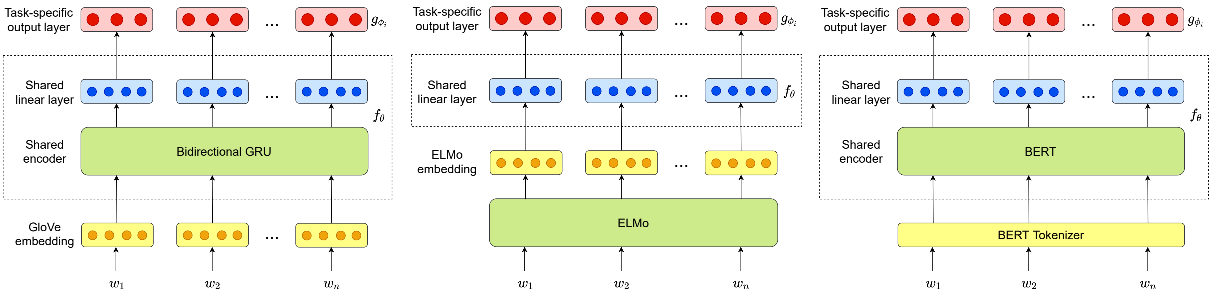
$$J(\boldsymbol{\theta}) = \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}^q_{\mathcal{T}_i}(f_{U(\mathcal{L}^s_{\mathcal{T}_i}, \boldsymbol{\theta}, \alpha, m)}). \tag{4}$$

where the loss $\mathcal{L}^q_{\mathcal{T}_i}$ is computed on $D^{(i)}_{query}$. The meta-optimization, or outer-loop optimization, does the update with the outer-loop learning rate β:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \beta \nabla_{\boldsymbol{\theta}} \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}^q_{\mathcal{T}_i}(f_{\boldsymbol{\theta}'_i}) \tag{5}$$

This involves computing second-order gradients, i.e., the backward pass works through the update step in Equation 3 – a computationally expensive process. Finn et al. (2017) propose a first-order approximation, called FOMAML, which computes the gradients with respect to $\boldsymbol{\theta}'_i$ rather than $\boldsymbol{\theta}$. The outer-loop optimization step thus reduces to:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \beta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \nabla_{\boldsymbol{\theta}'_i} \mathcal{L}^q_{\mathcal{T}_i}(f_{\boldsymbol{\theta}'_i}) \tag{6}$$



(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.

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 layer parameters $\boldsymbol{\phi}_i$ that are adapted per task. During the inner-loop for each task, the optimization is performed as follows:

$$\boldsymbol{\theta}'_i, \boldsymbol{\phi}'_i \leftarrow \text{SGD}(\mathcal{L}^s_{\mathcal{T}_i}, \boldsymbol{\theta}, \boldsymbol{\phi}_i, \alpha, \gamma, m) \quad (7)$$

where α and γ are the learning rates for the shared block and output layer respectively. We introduce different learning rates because 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:

$$\boldsymbol{\theta} \leftarrow \text{Adam}\left(\sum_i \mathcal{L}^q_{\mathcal{T}_i}(\boldsymbol{\theta}'_i, \boldsymbol{\phi}'_i), \beta\right) \quad (8)$$

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

ProtoMAML Snell et al. (2017) show that with Euclidean distance metric, prototypical networks are equivalent to a linear model with the following parameters: $\boldsymbol{w_c} = 2\boldsymbol{\mu_c}$ and $b_c = -\boldsymbol{\mu_c}^T\boldsymbol{\mu_c}$, where $\boldsymbol{w_c}$ and b_c are the weights and biases for the output unit corresponding to class c. Triantafillou et al. (2020) combine the strengths of prototypical networks and MAML by initializing the final layer of the classifier in each episode with these prototypical network-equivalent weights and biases and continue to learn with MAML, thus proposing a hybrid approach referred to as ProtoMAML. Similarly, using FOMAML would yield ProtoFOMAML. While updating $\boldsymbol{\theta}$, they allow the gradients to flow through the linear layer initialization.

We construct the prototypes from the output from $f_{\boldsymbol{\theta}}$ for the support examples. The parameters $\boldsymbol{\phi}_i$ are initialized as described above. The learning then proceeds as in (FO)MAML; 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 always predicts the most frequent sense in the support set. Hereafter, we refer to it as *MajoritySenseBaseline*.

Nearest neighbor classifier This model predicts 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 their BERT outputs but not with GloVe embeddings since they are the same for all senses. We refer to this baseline as *NearestNeighbor*.

Non-episodic training It is a single model that is trained on all tasks without any distinction between them – it merges support and query sets, and is trained using mini-batching. 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:

$$\boldsymbol{\theta}, \boldsymbol{\phi} \leftarrow \text{Adam}(\mathcal{L}_{\mathcal{T}_i}, \boldsymbol{\theta}, \boldsymbol{\phi}, \alpha) \quad (9)$$

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:

$$\boldsymbol{\theta}_i', \boldsymbol{\phi}_i' \leftarrow \text{SGD}(\mathcal{L}_{\mathcal{T}_i}, \boldsymbol{\theta}, \boldsymbol{\phi}, \alpha, \gamma, m) \quad (10)$$

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 refer to this model as *NE-Baseline*.

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

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 the metric does not improve 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 GloVe+GRU, ELMo+MLP and BERT 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. The output layer for these is initialized anew in every episode, whereas in NE-Baseline it has a fixed number of 5612 units. We use the `higher` package (Grefenstette et al., 2019) to implement the MAML variants.

5.2 Results

In Table 1, we report macro F1 scores averaged over all words in the meta-test set. We report the mean and standard deviation from five independent runs. We note that the results are not directly comparable across $|S|$ setups as, by their formulation, they involve different meta-test episodes.

GloVe+GRU All meta-learning methods perform better than their EF counterparts, indicating successful learning from the meta-training set. FOMAML fails to outperform NE-Baseline as well as the EF versions of the other meta-learning methods when $|S| = 8, 16, 32$. Interestingly, solely meta-testing is often better than NE-Baseline model which shows that the latter does not effectively transfer knowledge from the meta-training set. ProtoNet is the best-performing model (except when $|S| = 8$), with ProtoMAML being a close second.

ELMo+MLP The scores for NearestNeighbor, NE-Baseline and the EF methods are higher compared to GloVe-based models, which can be attributed to the input embeddings being contextual. ProtoNet and ProtoFOMAML still improve over their EF counterparts due to meta-training. ProtoFOMAML outperforms other methods for all $|S|$, and FOMAML is comparatively weak.

BERT The scores for all methods are higher than in case of the previous architectures, except for NE-Baseline and FOMAML. BERT-based ProtoNet, as well as ProtoFOMAML, outperform all other approaches for all $|S|$. Furthermore, ProtoFOMAML is superior to ProtoNet for $|S| = 4, 8$ and vice versa for $|S| = 16, 32$. Overall, across architectures, we see that NE-Baseline and FOMAML consistently underperform, whereas ProtoNet and ProtoFOMAML are the most effective methods. Moreover, they achieve a high disambiguation performance with as few as 4 training examples, which in many cases approaches a one-shot classification setting for individual senses (see Appendix A.1). The models are also relatively stable as indicated by the low standard deviations across runs.

Effect of second-order gradients We further experiment with ProtoMAML, including second-order gradients. In Table 2, we report its F1 scores alongside ProtoNet and ProtoFOMAML. For BERT, we train ProtoMAML while fine-tuning only the top layer and only for one inner-loop update step due to its high computational cost. We also train an equivalent ProtoFOMAML variant for a fair comparison. We can observe that ProtoMAML obtains scores similar to ProtoFOMAML in most cases, indicating the effectiveness of the first-order approximation. ProtoFOMAML achieves higher scores than ProtoMAML in some cases, perhaps due to an overfitting effect induced by the latter. In light of these results, we argue that first-order ProtoFOMAML suffices for this task.

5.3 Analysis

Effect of number of episodes We first investigate whether using more meta-training episodes always translates to higher performance. We plot the average macro F1 score for one of our high-scoring

Embedding/ Encoder	Method	Average macro F1 score $\|S\| = 4$	$\|S\| = 8$	$\|S\| = 16$	$\|S\| = 32$
-	MajoritySenseBaseline	0.247	0.259	0.264	0.261
GloVe+GRU	NearestNeighbor	–	–	–	–
	NE-Baseline	0.532 ± 0.007	0.507 ± 0.005	0.479 ± 0.004	0.451 ± 0.009
	EF-ProtoNet	0.522 ± 0.008	0.539 ± 0.009	0.538 ± 0.003	0.562 ± 0.005
	EF-FOMAML	0.376 ± 0.011	0.341 ± 0.002	0.321 ± 0.004	0.303 ± 0.005
	EF-ProtoFOMAML	0.519 ± 0.006	0.529 ± 0.010	0.540 ± 0.004	0.553 ± 0.009
	ProtoNet	$\mathbf{0.579 \pm 0.004}$	0.601 ± 0.003	$\mathbf{0.633 \pm 0.008}$	$\mathbf{0.654 \pm 0.004}$
	FOMAML	0.536 ± 0.007	0.418 ± 0.005	0.392 ± 0.007	0.375 ± 0.005
	ProtoFOMAML	0.577 ± 0.011	$\mathbf{0.616 \pm 0.005}$	0.626 ± 0.005	0.631 ± 0.008
ELMo+MLP	NearestNeighbor	0.624	0.641	0.645	0.654
	NE-Baseline	0.624 ± 0.013	0.640 ± 0.012	0.633 ± 0.001	0.614 ± 0.008
	EF-ProtoNet	0.609 ± 0.008	0.635 ± 0.004	0.661 ± 0.004	0.683 ± 0.003
	EF-FOMAML	0.463 ± 0.004	0.414 ± 0.006	0.383 ± 0.003	0.352 ± 0.003
	EF-ProtoFOMAML	0.604 ± 0.004	0.621 ± 0.004	0.623 ± 0.008	0.611 ± 0.005
	ProtoNet	0.656 ± 0.006	0.688 ± 0.004	0.709 ± 0.006	0.731 ± 0.006
	FOMAML	0.642 ± 0.009	0.589 ± 0.010	0.587 ± 0.012	0.575 ± 0.016
	ProtoFOMAML	$\mathbf{0.670 \pm 0.005}$	$\mathbf{0.700 \pm 0.004}$	$\mathbf{0.724 \pm 0.003}$	$\mathbf{0.737 \pm 0.007}$
BERT	NearestNeighbor	0.681	0.704	0.716	0.741
	NE-Baseline	0.467 ± 0.157	0.599 ± 0.023	0.539 ± 0.025	0.473 ± 0.015
	EF-ProtoNet	0.594 ± 0.008	0.655 ± 0.004	0.682 ± 0.005	0.721 ± 0.009
	EF-FOMAML	0.445 ± 0.009	0.522 ± 0.007	0.450 ± 0.008	0.393 ± 0.002
	EF-ProtoFOMAML	0.618 ± 0.013	0.662 ± 0.006	0.654 ± 0.009	0.665 ± 0.009
	ProtoNet	0.696 ± 0.011	0.750 ± 0.008	$\mathbf{0.755 \pm 0.002}$	$\mathbf{0.766 \pm 0.003}$
	FOMAML	0.676 ± 0.018	0.550 ± 0.011	0.476 ± 0.010	0.436 ± 0.014
	ProtoFOMAML	$\mathbf{0.719 \pm 0.005}$	$\mathbf{0.756 \pm 0.007}$	0.744 ± 0.007	0.761 ± 0.005

Table 1: Average macro F1 scores of the meta-test words.

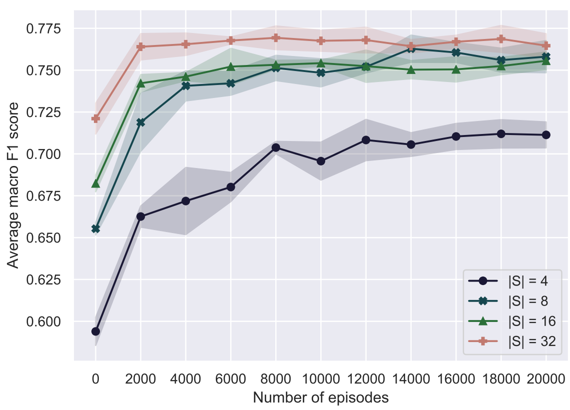


Figure 2: Average macro F1 score of ProtoNet+BERT as the number of meta-training episodes increases.

models – 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 data sizes; however, overall, the largest gains come with a minimum of around $8,000$ episodes.

Effect of number of senses To investigate the variation in performance with the number of senses, in Figure 3, we plot the macro F1 scores obtained from ProtoNet with BERT, averaged over words with a given number of senses in the meta-test set. We see a trend where the score reduces as the number of senses increase. Words with more senses seem to benefit from a higher support set size. For a word with 8 senses, the $|S| = 32$ case is 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 words with many senses improves with $|S|$ due to an increase in the effective number of shots.

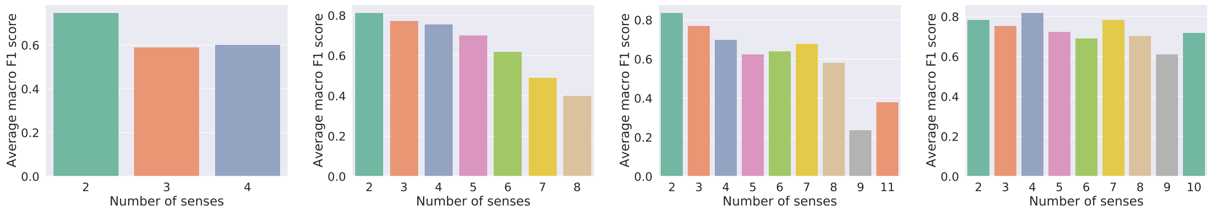
Challenging cases Based on the 10 words that obtain the lowest macro F1 scores with ProtoNet with GloVe+GRU (Appendix A.4), we see that verbs are the most challenging words to disambiguate without the advantage of pre-trained models and their disambiguation improves as $|S|$ increases.

6 Discussion

Our results demonstrate that meta-learning outperforms the corresponding models trained in a non-episodic fashion when applied in a few-shot learning setting – a finding consistent for all $|S|$ setups. Using the BERT-based models, we obtain up to 72% average macro F1 score with as few

Embedding/ Encoder	Method	Average macro F1 score $\|S\| = 4$	$\|S\| = 8$	$\|S\| = 16$	$\|S\| = 32$
GloVe+GRU	ProtoNet	**0.579 ± 0.004**	0.601 ± 0.003	**0.633 ± 0.008**	**0.654 ± 0.004**
	ProtoFOMAML	0.577 ± 0.011	0.616 ± 0.005	0.626 ± 0.005	0.631 ± 0.008
	ProtoMAML	**0.579 ± 0.006**	**0.617 ± 0.005**	0.629 ± 0.006	0.633 ± 0.006
ELMo+MLP	ProtoNet	0.656 ± 0.006	0.688 ± 0.004	0.709 ± 0.006	0.731 ± 0.006
	ProtoFOMAML	0.670 ± 0.005	0.700 ± 0.004	**0.724 ± 0.003**	**0.737 ± 0.007**
	ProtoMAML	**0.671 ± 0.006**	**0.702 ± 0.006**	0.722 ± 0.004	0.735 ± 0.008
BERT	ProtoNet	0.696 ± 0.011	**0.750 ± 0.008**	**0.755 ± 0.002**	**0.766 ± 0.003**
	ProtoFOMAML*	**0.697 ± 0.013**	**0.750 ± 0.005**	0.741 ± 0.007	0.754 ± 0.006
	ProtoMAML*	0.690 ± 0.003	0.736 ± 0.004	0.737 ± 0.006	0.752 ± 0.006

Table 2: Average macro F1 scores of the meta-test words for second-order gradient model variants as well as ProtoNet. (*Only the top layer fine-tuned and for only one inner-loop step)



(a) $|S| = 4$ (b) $|S| = 8$ (c) $|S| = 16$ (d) $|S| = 32$

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

as 4 examples, and closely approach the reported state-of-the-art performance[3] with $|S| = \{16, 32\}$.

The success of meta-learning is particularly evident with GloVe+GRU. GloVe embeddings are sense-agnostic and yet, ProtoNet, ProtoFOMAML and ProtoMAML approach the performance of some ELMo-based models, which enjoy the benefit of contextualization via large-scale pretraining.

Although contextualized representations from ELMo and BERT already contain information relevant to our task, integrating them into a meta-learning framework allows these models to substantially improve performance. To illustrate the advantage that meta-learning brings, we provide example t-SNE visualizations (van der Maaten and Hinton, 2008) of the original ELMo embeddings and those generated by ProtoNet based on ELMo (Figure 4). The representations from ProtoNet are more accurately clustered with respect to the senses than the original ELMo representations. ProtoNet thus effectively learns to disambiguate new words, i.e. separate the senses into clusters, thereby improving upon ELMo embeddings. We provide further t-SNE visualizations in Appendix A.6.

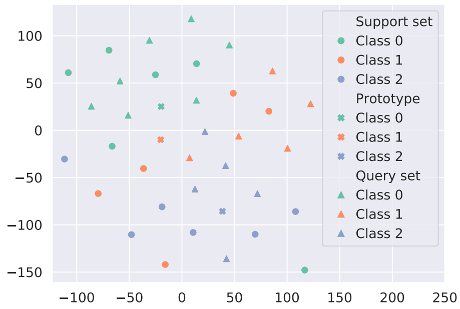
The success of ProtoNet and ProtoFOMAML can be in part attributed to the nature of the problem – WSD lends itself well to modeling approaches based on similarity (Navigli, 2009; Peters et al., 2018). Their relative ranking, however, depends on the architecture and the value of $|S|$. ELMo+MLP has the simplest architecture and ProtoFOMAML – an optimization-based method – performs best. For GloVe+GRU and BERT, which are more complex architectures, lower-shot settings benefit from ProtoFOMAML and higher-shot settings from ProtoNet. The reasons for this effect, however, remain to be investigated in future work.

Our experiments further highlight the weakness of FOMAML when applied beyond the N-way, K-shot setting. This may be due to the fact that the number of "new" output parameters in each episode is much greater than the number of support examples. Informed output layer initialization in Proto(FO)MAML is therefore important for effective learning in such scenarios. A similar problem with FOMAML is also pointed out by Bansal et al. (2019), who design a differentiable parameter generator for the output layer.

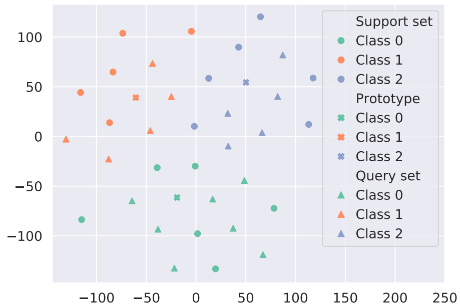
7 Conclusion

Few-shot learning is a key capability for AI to reach human-like performance. The development of meta-learning methods is a promising step in this direction. We demonstrated the ability of meta-learning to disambiguate new words when only a

[3]Not a direct comparison due to different data splits.



(a)



(b)

Figure 4: t-SNE visualizations comparing ELMo embeddings (left) against representations generated by ProtoNet with ELMo+MLP (right) for the word ‘field’.

handful of labeled examples are available. Given the data scarcity in WSD and the need for few-shot model adaptation to specific domains, we believe that meta-learning can yield a more general and widely applicable disambiguation model than traditional approaches. Interesting avenues to explore further would be a generalization of our models to disambiguation in different domains, to a multilingual scenario or to an altogether different 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. In *International Conference on Learning Representations*.

Trapit Bansal, Rishikesh Jha, and Andrew McCallum. 2019. Learning to few-shot learn across diverse natural language classification tasks. *arXiv preprint arXiv:1911.03863*.

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–.

Michele Bevilacqua and Roberto Navigli. 2020. Breaking through the 80% glass ceiling: Raising the state of the art in word sense disambiguation by incorporating knowledge graph information. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2854–2864, Online. Association for Computational Linguistics.

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. *arXiv preprint arXiv:1611.02779*.

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. 2020. MetalGAN: Multi-domain label-less image synthesis using cGANs and meta-learning. *Neural Networks*, 131:185–200.

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. *arXiv preprint arXiv:1910.01727*.

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. *arXiv preprint arXiv:1912.06290*.

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 24–26, 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. *arXiv preprint arXiv: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.

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 *Advances in Neural Information Processing Systems 30*, pages 4077–4087.

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

Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. 2018. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1199–1208.

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. 2020. Meta-dataset: A dataset of datasets for learning to learn from few examples. In *International Conference on Learning Representations*.

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

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. 2016. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems 29*, pages 3630–3638.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, 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. In *IEEE Conference on Computer Vision and Pattern Recognition (oral paper)*. IEEE.

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. *arXiv preprint arXiv:1611.05763*.

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. 2020. Meta-learning without memorization. In *International Conference on Learning Representations*.

Dani Yogatama, Cyprien de Masson d'Autume, Jerome Connor, Tomas Kocisky, Mike Chrzanowski, Lingpeng Kong, Angeliki Lazaridou, Wang Ling, Lei Yu, Chris Dyer, et al. 2019. Learning and evaluating general linguistic intelligence. *arXiv preprint arXiv:1901.11373*.

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 Data statistics

We report the number of words, the number of episodes, the total number of unique sentences and the average number of senses for the meta-training, meta-validation and meta-test sets for each of the four setups with different $|S|$ in Table 3. Additionally, in Figure 5 and Figure 6, we present bar plots of the number of meta-test episodes for different number of senses in the meta-test support and query sets respectively. It shows that the number of episodes drops quite sharply as the number of senses increases. In each episode, only words with a maximum of $|S|$ senses are considered so that all of them are accommodated in the support set.

A.2 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 five seeds (42 - 46) and recorded the mean of the metric from the five runs to identify the best hyperparameters. For $|S| = 4, 8, 32$, we chose the best hyperparameters obtained from this tuning.

We employed early stopping with a patience of 2 epochs, i.e., we stop meta-training if the validation metric 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, ProtoFOMAML 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, 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 search space for these is as follows:

- Output learning rate: 1e−1, 1e−2, 5e−3, 1e−3
- Learner learning rate: 1e−1, 5e−2, 1e−2, 5e−3, 1e−3, 5e−4, 1e−4, 5e−5, 1e−5
- Meta learning rate: 5e−3, 1e−3, 5e−4, 1e−4, 5e−5, 1e−5, 5e−6, 1e−6
- Hidden size: 64, 128, 256
- Number of inner-loop updates: 3, 5, 7

The best hyperparameters obtained are shown in Table 5.

A.3 Training times

We train all our models on TitanRTX GPUs. Our model architectures vary in the total number of trainable parameters. Thus, the time taken to train each of them varies. The number of meta-learned parameters $\boldsymbol{\theta}$ is as follows:

- GloVe+GRU: $889,920$
- ELMo+MLP: $262,404$
- BERT: $107,867,328$

To give an idea of how long it takes to train them, we provide the approximate time taken for one epoch for the $|S| = 16$ setup in Table 6. The training time would be slightly lower for $|S| = 4, 8$ 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 is not supported in PyTorch and hence cuDNN had to be disabled).

A.4 Challenging cases

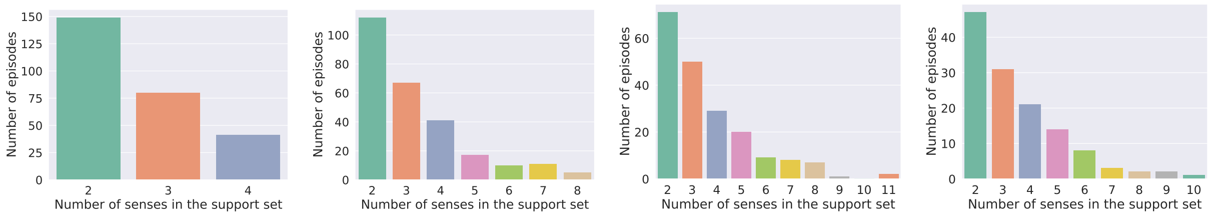
In Table 4, we present 10 words with the lowest macro F1 scores (in increasing order of the score) obtained from ProtoNet with GloVe+GRU. We perform the analysis on this model to investigate challenging cases without the contextualization advantage offered by ELMo and BERT. For $|S| = 4, 8, 16$, many words in the list have predominantly verb senses, showing that they are more challenging to disambiguate. The number of such cases drops in $|S| = 32$, indicating that disambiguation of verbs improves as $|S|$ increases.

A.5 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 7. The distribution is mostly right-skewed with very few words having scores in the range 0 to 0.2.

Support sentences	Split	No. of words	No. of episodes	No. of unique sentences	Average no. of senses
4	Meta-training	985	10000	27640	2.96
	Meta-validation	166	166	1293	2.60
	Meta-test	270	270	2062	2.60
8	Meta-training	985	10000	27640	2.96
	Meta-validation	163	163	2343	3.06
	Meta-test	259	259	3605	3.16
16	Meta-training	799	10000	27973	3.07
	Meta-validation	146	146	3696	3.53
	Meta-test	197	197	4976	3.58
32	Meta-training	580	10000	27046	3.34
	Meta-validation	85	85	4129	3.94
	Meta-test	129	129	5855	3.52

Table 3: Statistics of our few-shot WSD dataset.



(a) $|S| = 4$ (b) $|S| = 8$ (c) $|S| = 16$ (d) $|S| = 32$

Figure 5: Bar plot of number of meta-test episodes for different number of senses in the meta-test support set.

$\lvert S\rvert = 4$	ward, delicate, jam, lose, year, bounce, haul, introduce, guard, suffer
$\lvert S\rvert = 8$	bad, work, give, clear, settle, bloom, draw, check, break, gather
$\lvert S\rvert = 16$	move, appearance, in, green, fix, establishment, note, drag, cup, bounce
$\lvert S\rvert = 32$	independent, gather, north, square, do, bond, proper, pull, problem, language

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

A.6 t-SNE visualizations

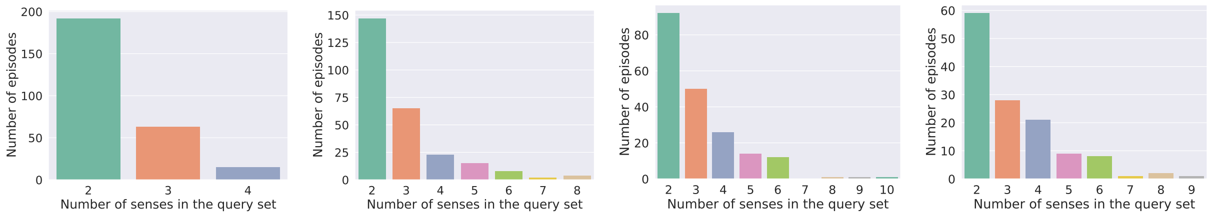
We provide t-SNE visualizations of the word representations generated by f_θ of ProtoNet with GloVe+GRU for three words (with macro F1 score of 1) in the meta-test set in Figure 8. Even though it receives the same input embedding for all senses, it manages to separate the senses into clusters on the basis of the representations of the support examples. This occurs even though ProtoNet does not perform any fine-tuning step on the support set. 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 9a, 9b and 9c. We also show the prototypes computed from these embeddings for illustration. For the same three words, we plot the representations obtained from f_θ of ProtoNet with ELMo+MLP in Figure 9d, 9e and 9f. It can be observed that the ELMo embeddings alone are not well-clustered with respect to the senses. On the other hand, ProtoNet manages to separate the senses into clusters, which aids in making accurate predictions on the query set.

These visualizations further demonstrate ProtoNet's success in disambiguating new words. From a learning to learn standpoint, the model has learned how to separate the senses in a high-dimensional space so as to disambiguate them. Proto(FO)MAML often improves upon this good initialization during its inner-loop updates.

A.7 Results on the meta-validation set

To facilitate reproducibility, we provide the results on the meta-validation set for all the methods that involved hyperparameter tuning in Table 7.



(a) $|S| = 4$ (b) $|S| = 8$ (c) $|S| = 16$ (d) $|S| = 32$

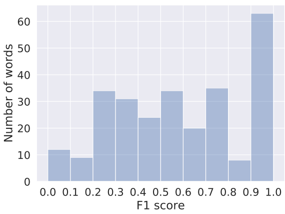
Figure 6: Bar plot of number of meta-test episodes for different number of senses in the meta-test query set.

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	1e−3	1e−3	5e−4	–	7	256
	ProtoMAML	1e−3	1e−3	5e−4	–	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

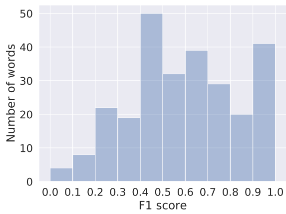
Table 5: Hyperparameters used for training the models.

Embedding/ Encoder	**Method**	**No. of GPUs used**	**Approximate training time per epoch**
GloVe+GRU	NE-Baseline	1	8 minutes
	ProtoNet	1	8 minutes
	FOMAML	1	15 minutes
	ProtoFOMAML	1	18 minutes
	ProtoMAML	1	9 hours 30 minutes
ELMo+MLP	NE-Baseline	1	55 minutes
	ProtoNet	1	55 minutes
	FOMAML	1	1 hour
	ProtoFOMAML	1	1 hour
	ProtoMAML	1	1 hour 2 minutes
BERT	NE-Baseline	1	35 minutes
	ProtoNet	1	35 minutes
	FOMAML	4	2 hours 35 minutes
	ProtoFOMAML	4	4 hours 18 minutes
	ProtoFOMAML*	1	41 minutes
	ProtoMAML*	1	47 minutes

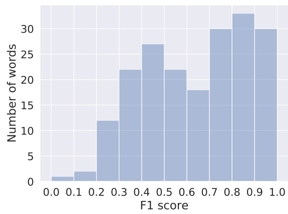
Table 6: Approximate training time per epoch. (*Only the top layer fine-tuned and only for one inner-loop step.)



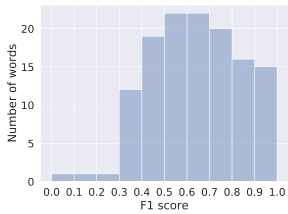
(a) $|S| = 4$



(b) $|S| = 8$

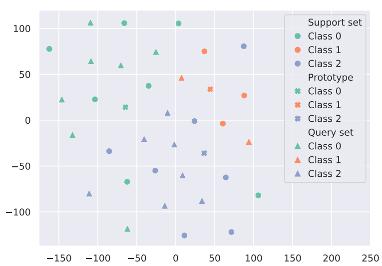


(c) $|S| = 16$

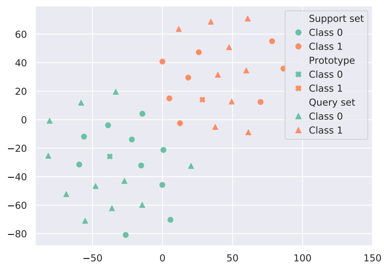


(d) $|S| = 32$

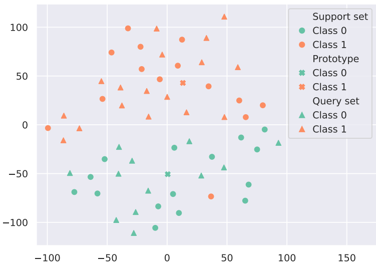
Figure 7: Distribution of macro F1 scores for ProtoNet with GloVe+GRU.



(a) Word representation – 'favor'

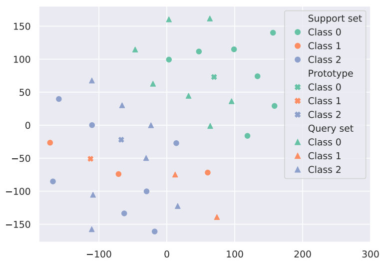


(b) Word representation – 'tax'

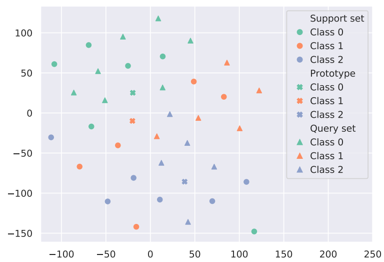


(c) Word representation – 'snow'

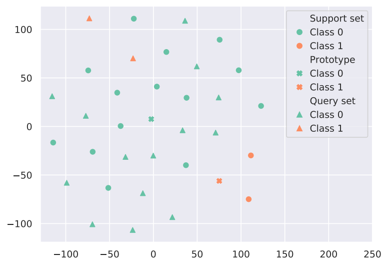
Figure 8: t-SNE visualizations of word representations generated by ProtoNet with GloVe+GRU.



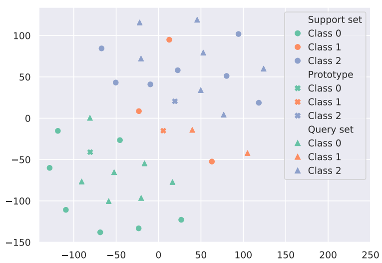
(a) ELMo embedding – 'favor'



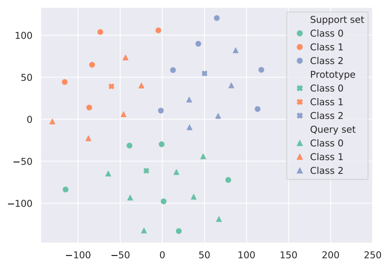
(b) ELMo embedding – 'field'



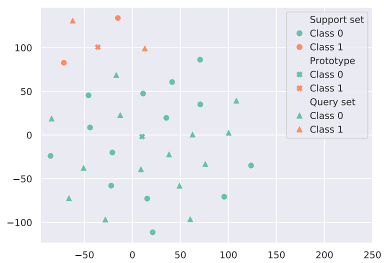
(c) ELMo embedding – 'factor'



(d) Word representation – 'favor'



(e) Word representation – 'field'



(f) Word representation – 'factor'

Figure 9: t-SNE visualizations comparing ELMo embeddings (top) against word representations generated by ProtoNet with ELMo+MLP (bottom).

Embedding/ Encoder	Method	Average macro F1 score $\lvert S\rvert = 4$	$\lvert S\rvert = 8$	$\lvert S\rvert = 16$	$\lvert S\rvert = 32$
GloVe+GRU	NE-Baseline	0.557 ± 0.015	0.563 ± 0.011	0.590 ± 0.008	0.541 ± 0.018
	ProtoNet	0.591 ± 0.008	0.615 ± 0.001	0.638 ± 0.007	0.634 ± 0.006
	FOMAML	0.540 ± 0.011	0.410 ± 0.006	0.405 ± 0.007	0.351 ± 0.007
	ProtoFOMAML	**0.604 ± 0.016**	**0.622 ± 0.010**	**0.642 ± 0.005**	0.626 ± 0.015
	ProtoMAML	0.599 ± 0.004	**0.622 ± 0.010**	0.641 ± 0.005	**0.627 ± 0.013**
ELMo+MLP	NE-Baseline	0.659 ± 0.016	0.685 ± 0.005	0.728 ± 0.004	0.693 ± 0.007
	ProtoNet	0.682 ± 0.008	0.701 ± 0.007	**0.741 ± 0.007**	0.722 ± 0.011
	FOMAML	0.670 ± 0.007	0.609 ± 0.011	0.598 ± 0.017	0.566 ± 0.011
	ProtoFOMAML	**0.702 ± 0.002**	**0.728 ± 0.007**	0.740 ± 0.003	0.732 ± 0.005
	ProtoMAML	**0.702 ± 0.007**	0.726 ± 0.008	**0.741 ± 0.003**	**0.738 ± 0.006**
BERT	NE-Baseline	0.466 ± 0.160	0.601 ± 0.006	0.616 ± 0.009	0.569 ± 0.006
	ProtoNet	**0.742 ± 0.007**	0.759 ± 0.013	0.786 ± 0.004	**0.770 ± 0.009**
	FOMAML	0.702 ± 0.005	0.553 ± 0.019	0.506 ± 0.014	0.418 ± 0.020
	ProtoFOMAML	0.740 ± 0.010	0.756 ± 0.008	**0.770 ± 0.009**	0.734 ± 0.014
	ProtoFOMAML*	0.738 ± 0.016	**0.763 ± 0.003**	0.769 ± 0.006	0.744 ± 0.006
	ProtoMAML*	0.737 ± 0.012	0.760 ± 0.007	0.764 ± 0.005	0.736 ± 0.009

Table 7: Average macro F1 scores of the meta-validation words. (*Only the top layer fine-tuned and for only one inner-loop step)