Meta-Learning for Pragmatics and Social Meaning Tasks

Tamara Czinczoll 12858609
Christoph Hönes 12861944
Daniel Rodríguez Baleato 12799971
Ard Snijders 12854913

Abstract

A disadvantage of pre-trained task-agnostic models is that effective fine-tuning to downstream tasks often requires large amounts of data. This is problematic in scenarios where large datasets are scarce, which can lead to poor generalisation. Meta-learning aims to address this, with models explicitly trained for quicker adaption using only a few examples. We train two meta-learning models (ProtoNet & ProtoMAML) for generalisation to unseen tasks, in a few-shot setting, using a suite of tasks related to social meaning, a low-resource domain. We find that ProtoMAML outperforms both ProtoNet and a multi-task baseline. A subsequent ablation study suggests that gains achieved by ProtoMAML largely stem from its superior training paradigm.

1 Introduction

Recent developments in NLP, particularly with transformer-based architectures such as BERT, have led to a surge in popularity for fine-tuning large, pre-trained language models to a variety of downstream NLP tasks, beating the state-of-the-art on a number of important benchmarks (Devlin et al., 2019). However, a major drawback of these task-agnostic models is that fine-tuning requires large amounts of data to work effectively, which is problematic for scenarios where the collection of annotated data is costly or difficult. In such instances, such models tend to generalize poorly when only given a few examples (Yogatama et al., 2019). Meta-learning has recently emerged as a novel method which seeks to address this. Through use of a new learning paradigm called episodic learning, meta-learning methods are designed to "learn the learning algorithm", by training on related tasks such that models can learn to adapt to a new task efficiently, using only a few examples (Schmidhuber, 1987; Thrun and Pratt, 1998). In this work, we apply meta-learning across tasks in the domain of social language and pragmatics, testing on tasks such as irony detection and abusive language classification. To the best of our knowledge, there has been no previous work in which meta-learning methods were applied to this particular domain. Second, there is a lack of large annotated corpora for many tasks related to social meaning. This provides us with a realistic scenario in which a few-shot approach might be favourable to typical pre-trained transfer-based models. Finally, many tasks related to social language are notoriously difficult. For instance, in the case of figurative language such as irony and sarcasm, much of the ambiguity stems from the gaps between the literal and true interpretation of texts, and solving such tasks requires complex reasoning, employing both context and world-knowledge (Hee et al., 2018). In this respect, the inherent difficulty of the domain and its associated tasks could provide us with insights on the limits of current meta-learning methods.

We explore two popular meta-learning models: prototypical networks (Snell et al.), and ProtoMAML (Triantafillou et al., 2019). We compare these methods against a multi-task learning baseline. While multi-task learning methods lack an explicit meta-objective, they tend to have improved generalisation to unseen tasks when compared to single-task methods, due to having learned more general representations as a result of being trained with a multi-task objective (Wang et al., 2019). Multi-task methods can thus provide us with a competitive and meaningful baseline to compare our meta-learning models against. Similar to Bansal et al. (2019), we evaluate on generalisation, testing on unseen tasks with a few-shot setup. To this end, we seek to explore the following research question: *Can meta-learning adapt faster to new tasks related to pragmatics and social meaning, compared to multi-task learning?*

1.1 Related Work & Contribution

Meta-learning in NLP While a number of works investigating meta-learning have emerged in recent time, its applications to NLP are still relatively new. Previous applications include neural machine translation (Gu et al., 2018; Mi et al., 2019), few-shot text classification (Yu et al., 2018; Geng et al., 2019) and relation classification (Han et al., 2018). More recently, Bansal et al. (2019) proposed an optimization-based meta-learning framework suitable for tasks with varying numbers of classes, outperforming both pre-trained and multi-task learned models across a diverse range of NLP tasks.

Few-Shot Learning For Social Meaning Wu et al. (2018) perform irony detection with a multi-task approach - however, their related tasks are constructed from the same dataset as their main task (i.e. missing hashtag prediction), and they thus do not consider performance on other datasets, or unseen tasks. Stappen et al. (2020) study cross-lingual hate speech detection for low-resource languages, in a zero- and few-shot setting, using a frozen Transformer Language Model. However, their framework is specifically tailored towards hate speech, and they do not train for generalisation.

Contribution We contribute to the existing body of work by exploring two meta-learning methods in the domain of social language and pragmatics, evaluating on few-shot generalisation by testing on unseen tasks, some of which include unseen classes and/or are sourced from a different domain than what was trained on. We study the strengths and weaknesses of both models with respect to each other, and a multi-task baseline. We assess how holding out train tasks might affect performance on a Sentiment Analysis task. We find that meta-learning models consistently outperform MTL across all tasks. Subsequent ablation experiments suggest that performance differences mostly stem from ProtoMAML's superior training paradigm.

2 Models

2.1 Base architecture

All models use the same encoder for embedding examples. For this, we use BERT-base, a 12-layer, uncased transformer-based model (Devlin et al., 2019) which has shown great success in learning general linguistic structure across different domains. The lower, more general layers are kept frozen at all times; only the last two layers are fine-tuned on our tasks. On top of the BERT model, every model has one additional linear classification layer, which is trained from scratch.

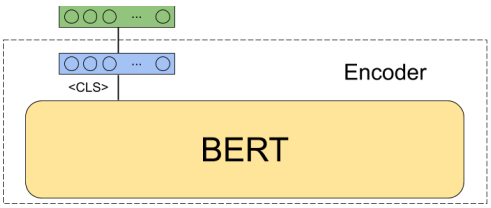

Figure 1: Model architecture. The CLS token representation from BERT (Base uncased) is funnelled through a fully connected layer (in blue). For ProtoMAML one additional fully-connected layer was added to serve as output layer (in green). The Multi-task model uses multiple output layers, one per task; whereas the Prototype network uses none.

2.2 Multi-task Baseline

For our multi-task learner (MTL) baseline, we use hard parameter sharing for our base architecture, and a task-specific linear output layer which is separately optimized for each task. Typically, the loss of the auxiliary tasks is down-scaled with respect to the loss of the target task. However, our main goal is to learn general representations with good generalisation to unseen tasks. To this end, we optimize our MTL model using a uniform multi-task objective, where the loss of each task is weighted equally. The MTL model is trained with a cross-entropy loss.

2.3 Meta-Learning Models

2.3.1 Episodic Learning

An important difference between meta-learning and more conventional machine learning concerns the learning paradigm. Instead of learning from a collection of examples, we instead learn from a collection of few-shot tasks, which are called episodes. Each episode has its own training and test set, which we call the support set and query set, respectively. The model is trained on the support set and then evaluated on the query set. The query set's performance depends on how successful the few-shot training on the support set was (i.e. how quickly the model was able to adapt to the episode's task). The meta model's loss is then calculated based on the query set's performance, and the model is updated.

2.4 ProtoNet & ProtoMAML

In this work we further explore two distinct types of meta-learning: metric-based, and optimization-based meta-learning. With metric-based methods, an embedding function encodes input examples into vector representations, after which a probability distribution over labels is computed for all query examples, based on their similarity with examples from the support set, as measured by some distance function. Optimization-based approaches, on the other hand, encourage the ability to quickly adapt to a new task in the objective function.

2.4.1 Prototypical Networks

The Prototypical Network (hereafter: ProtoNet) is a meta-learning technique proposed by (Snell et al.). It uses a *Nearest-Neighbour classifier* for its predictions. The weights are trained to produce meaningful embeddings of the input that result in good separability by the NN-classifier. The model computes prototypes for each label by averaging their respective embeddings obtained from the support set. These are used as the centroids of the NN-classifier. The examples in the query set are then classified by assigning them the label of the closest prototype. The prototypes circumvent the problem of finding a suitable initialization of an output layer for a new unobserved task.

2.4.2 ProtoMAML

ProtoMAML (Triantafillou et al., 2019) combines the advantages of ProtoNets (Snell et al.) with the MAML algorithm (Finn et al., 2017). The MAML algorithm is designed to make a model learn general representations from which it can rapidly adapt to any task with a limited number of examples. Moreover, this algorithm is applicable to any arbitrary model architecture. In each iteration a batch of training tasks is sampled. For each of those tasks, a small number of gradient updates (as shown in eq. 1) is performed on the support set of the episode, to fine-tune the parameters to perform well on this task (inner loop).

$$\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta) \tag{1}$$

θ are the original model parameters, α is the inner loop learning rate and $\mathcal{L}_{\mathcal{T}_i}$ is the task loss with respect to model predictions f_θ. Consequently, the success of the inner loop updates is evaluated on the query set for each task (outer loop), and the sum of these query set losses is used to update the original model parameters (before the inner loop updates), as shown in eq. 2.

$$\theta \longleftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}(f_{\theta'}) \tag{2}$$

This objective encourages the model to learn representations from which the model can quickly adapt to different tasks. However, there are some drawbacks of this algorithm. The standard MAML algorithm uses second-order gradients, which are costly to compute, and can make training with limited computing resources infeasible. Inspired by Finn et al. (2017), we instead use the first-order approximation of the MAML algorithm, which simply omits the second-order gradients (introduced by ∇_θ in eq. 2) and only uses the first-order gradients with respect to θ' from the query set evaluation.

Furthermore, the original algorithm is limited to work only with tasks that have the same number of output classes, as the output layer is shared across all tasks. We can overcome this by employing prototypes as calculated by a ProtoNet. These can be reinterpreted as the weights and biases of a linear layer (**?**), as shown in eq. 3.

$$W_k = 2p_k \qquad b_k = -||p_k||^2 \tag{3}$$

W_k corresponds to the k-th row of the output weight matrix and b_k is the k-th entry of the bias vector. p_k is the prototype of label k with $k = 0, \ldots, N-1$ where N is the number of class labels. Computing one prototype for each label ensures the right number of output classes. Additionally, they serve as good initializations for the output layer. A ProtoMAML is roughly a ProtoNet trained with MAML. However, a difference for the classification is that ProtoMAML uses the output layer initialization shown in eq. 3 and fine-tunes these parameters in the inner loop, instead of using a distance metric for classification.

3 Tasks & Datasets

We use a total of seven datasets: three for training, one for validation and four for testing. Our datasets cover a variety of pragmatics and social meaning tasks, and are mostly sourced from English Twitter data (except for the Abuse task, which is based on forum data). A compact summary of the datasets can be seen in Table 1. We test on 4 tasks, which stem from three different datasets. As these datasets do not feature a *train/dev/test* split, we apply a $70/15/15$ split.

3.1 Training tasks

Emotion Classification: We choose to use the Sem-Eval 2018 Emotion dataset, which is comprised of 10.983 tweets labelled according to the emotion(s) they express (Mohammad et al., 2018). There are 11 classes in total, and tweets can be assigned multiple labels. We re-purpose this dataset from a single, 11-way classification task, into 11 binary classification tasks - we motivate this with the hypothesis that meta-learning models will learn more easily when trained on a larger set of tasks. Due to its fine-grained nature, we expect that this task is useful in the training phase, as the recognition of different emotions could have valuable carry-over to other tasks, such as sentiment analysis and abusive language classification.

Sarcasm: We use a dataset as provided by the Second Workshop on Figurative Language Processing (on Figurative Language Processing, 2020). The data consists of 5000 tweets with binary labels. In some instances, context tweets are provided when target tweets were written in response to other tweets - here, we choose to append responses to the main tweet prior to encoding.

Offensive Language: For this task, we train on the OffensEval task from Sem-Eval 2019, which is based on the OLID dataset (Zampieri et al., 2019). It consists of 14.101 tweets with binary labels, indicating whether a tweet is offensive or not.

3.2 Validation

Sentiment Analysis: For validating models, we evaluate on a tweet-based Sentiment Analysis task (Sem) with 2.726 examples. Originally, this dataset has five labels of fine-grained sentiment labels. We choose to only consider examples labeled as *positive* or *negative*, thereby turning it into a binary classification task. We motivate our choice to evaluate with this task with the assumption that sentiment analysis is a rather general task, and in this respect provides good means to test whether our models have learned good general representations, which are broadly suitable for a wide array of tasks.

3.3 Testing Tasks

Irony: We test on two different irony detection tasks. Both tasks stem from Sem-Eval 2018, task 3, which are constructed with the same dataset of 3.817 tweets (Van Hee et al., 2018). Subtask A is a binary irony detection task; subtask B is a more fine-grained irony classification task; here, the examples labelled as ironic from subtask A are split into three, more specific distinctions: verbal irony as arising from polarity contrast, other verbal irony, and situational irony. We motivate our choice for testing on irony as it is known to be a difficult task - to this end, testing on this task might give us meaningful insights on the limits of current few-shot approaches.

Abusive Language: We use a Twitter dataset with 16.202 tweets, labelled according to expression of sexism, racism, or neither (Waseem and Hovy, 2016). While the models have seen a similar task during training (offensive language detection), it should be noted that the abusive language task introduces new, previously unseen classes, distinguishing between two special cases of abusive language. Furthermore, while this dataset is similarly tweet-based, its examples were sourced differently [1]. In this respect, we argue that using this task during testing can provide us with insights on the ability of models to transfer knowledge from the highly related OLID dataset, and how the difference in underlying distributions of both tasks might affect performance outcomes.

Politeness: For this task, we employ the Stanford Politeness corpus (Danescu-Niculescu-Mizil et al., 2013), where 4.353 examples are labelled according to expressing politeness, impoliteness, or neutral. Whereas all the other datasets are based on Twitter data, this dataset is comprised of requests from Wikipedia editors' talk pages. Possibly, few-shotting on this task enables us to observe how out-of-domain tasks might affect performance across models differently, compared to testing on tasks based on data seen by the model already (which would be the case for the tweet-based test tasks)

4 Experimental Set-Up

4.1 Data Sampling

To ensure a fair comparison between our three models, we use the same data sampling strategy for all of them. To account for the differences in size for different datasets, training tasks are sampled with a probability proportional to the square root of the number of training examples. This ensures

[1] Waseem and Hovy (2016) specifically collected data with an emphasis on sexism and racism; The OLID dataset was constructed by querying the Twitter API with general offensive terms

that tasks with a higher number of samples are still represented more strongly, without becoming too dominant. For the episodic training of both ProtoNet and ProtoMAML, the training set is split into two halves; From one half we draw the support set, and from the other the query set. Each support and query set contains a number of examples equivalent to the batch size. The multi-task model receives a single batch per iteration. If possible (given the batch size and number of classes), the data sampler returns a batch with an equal number of examples per label.

4.2 Hyperparameters

For all models we use the Cross-Entropy loss with a softmax module. For optimizing BERT parameters we use the AdamW (Loshchilov and Hutter, 2017) optimizer; the additional linear layers are optimized with Adam (Kingma and Ba, 2014). On top of BERT, we use a 768-dim additional linear layer, followed by a ReLU activation function for the MTL and ProtoMAML. We employ a cosine learning rate scheduler for all models, with a warm-up of 200 iterations for the BERT parameters. For a full hyperparameter overview, see 4 in the Appendix. We train MTL, Proto-Net and ProtoMAML for $10,000$, $10,000$ and $3,000$ iterations respectively, all the while validating the model on Sentiment Analysis every 200 to 300 iterations. We evaluate models on Sentiment Analysis in an 8-shot setting, and select the best performing models on this task as our final testing models.

4.3 K-Shot Setup

Our main experiment concerns few-shot testing on the four testing tasks, which are unseen for all models. The idea is to simulate having only a few labeled examples and check how well the models can adapt to the new task. For each test task, we create 32 episodes with 4, 8 and 16 examples per label. We then perform 100 updates per example, and evaluate their performance on the whole test set.

Before model comparison, we need to decide what learning rate to use for each model in this phase, as the one used to produce the best model during training does not necessarily provide the strongest results for k-shot testing. In order to provide a fair comparison between all models, we conduct a parameter search to find the best learning rate following the procedure described in the previous paragraph. Figure 3 shows this exploratory analysis. During the k-shot testing we use a learning rate of 0.025, 1, and 0.01 for the Multi-task baseline, the ProtoNet and the ProtoMAML respectively. Note that for the meta-learning models, the output layer is comprised of prototypes which are learned from the support set. For the Multi-task model, the output layer is initialized randomly. We test 32 episodes per task and perform 3 runs for every episode, after which we take the average for our final result.

We also conduct a set of peripheral experiments where we only fine-tune our base architecture (BERT-base and the additional layer + an output layer) on each test task separately, in a conventional single-task fashion. We argue that obtaining such scores can provide us with an indicator of how well a non-few-shot model would perform. This gives us a more informed notion of the 'true' difficulty of each task, in turn enabling us to make more meaningful comparisons with our few-shot architectures. Similarly to our methods in 3, we employ uniform sampling to obtain $train/dev/test$ splits for the test tasks.

4.4 Carousel Experiments

We also want to investigate the benefit of each task on the generalisation ability of each model. Ideally, we would set each task as test task once, and then inspect the change in performance when trained on each other task individually - however, such experiments fall outside of the scope of this project. Instead, during training we hold-out each task once, and compare the performance on our validation task. Our intuition is that Sentiment Analysis is a task which does well when the model encodes general information, which aligns with our goal of learning good general representations. We argue that if the performance of a model drops when a particular task is held out, it is likely that including that task improves generalization for that model.

5 Results and Analysis

5.1 K-shot Performance

Results for the K-shot experiments can be seen in Figure 2. As expected, meta-models seem to outperform the MTL model across most tasks. Furthermore, in line with our expectations, ProtoMAML outperforms ProtoNet on all tasks, except for IronyB: on this task, all models seem to perform approximately equal. It can also be noted that across the board, the performance on both irony

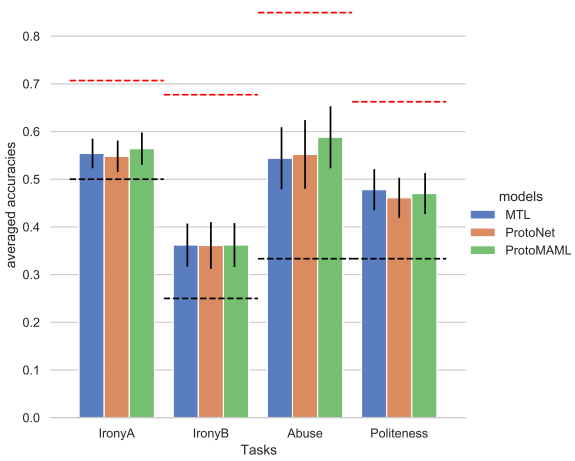

Figure 2: Averaged accuracy across all k-shot experiments. The black dotted line represents expected outcomes under random prediction. The red dotted line denotes the performance achieved by a single-task architecture which was fine-tuned on only that task. Black vertical dashes represents the standard deviation over multiple runs.

tasks is still comparatively poor, compared to both the random-guess and single-task baselines. Possibly, this suggests that such tasks still pose a difficult challenge in few-shot settings. The success of the ProtoMAML is most apparent in the Abuse task. Since the OLID dataset that was used for training is a similar task, but with different data and number of classes, this implies that the ProtoMAML best captures the general information of the offensive language detection task and can successfully transfer it to the Abuse task which focuses on a special case of offensive language. Curiously, the MTL model outperforms both meta-models on Politeness, despite the data stemming from another source. Theoretically, we would expect to see meta-models outperform MTL here, on the basis that by design, meta-models might be more adequately equipped to perform well on unseen tasks, or tasks with different underlying data. However, we find no evidence for that here.

Results for average performance across tasks, for different values of k-shots can be seen in Figure 3. In line with expectations, increasing the number of shots appears to correspond to a consistent increase in performance for all models, particularly for the meta-models, which seem to benefit comparatively more.

5.2 Learning Rate exploration for K-shot learning

See Figure 4 for results. As can be seen, the ProtoMAML outperforms the other models. Furthermore, it appears to be insensitive to the learning rate for smaller values. In contrast, MTL seems to be comparatively more sensitive to the learning rate value. The ProtoNet seems to be the least sensitive to changes in the learning rate. We also see that its performance increases at values higher than 1. This could indicate that the meta learning models learn broader local optima and are hence less sensitive to the learning rate.

5.3 Mixing model parameters

In an attempt to explain the observed differences in performance outcomes, we conduct several ablation studies. Particularly, we are interested in seeing whether the observed performance differences between models stem from a superior set of learned parameters. To this end, we re-run our initial k-shot experiments, although this time we initialize the base architecture of both the MTL and ProtoNet with the trained ProtoMAML parameters. Here we also vary learning rates for task-specific output layers, to see how this affects different models. See Figure 4 for results. As can be observed, re-initialising with ProtoMAML parameters improves the performance of both MTL and Protonet when compared to previous experiments, with accuracy gains of 1.28% and 2.56%, respectively. This suggests that the gains achieved by ProtoMAML stem mostly from its ability to learn a parameter set which is better-suited for generalisation to unseen tasks. Furthermore, it can be seen that ProtoNet achieves performance comparable to ProtoMAML even on high learning rates, which could point to a hybrid approach that reduces the LR dependence even further while maintaining the best performance.

We also test MTL (with its best-performing learning rate) using a ProtoMAML-like prototype initialization for the classifier. which scores an average accuracy of 47.35%, 1.11% less than its best value and 2.39% less than when initialised with ProtoMAML parameters. These results suggest that ProtoMAML boasts superior performance due to its training paradigm, in which it learns a set of parameters that can be more rapidly adapted to new tasks, and not specifically due to its centroid initialization.

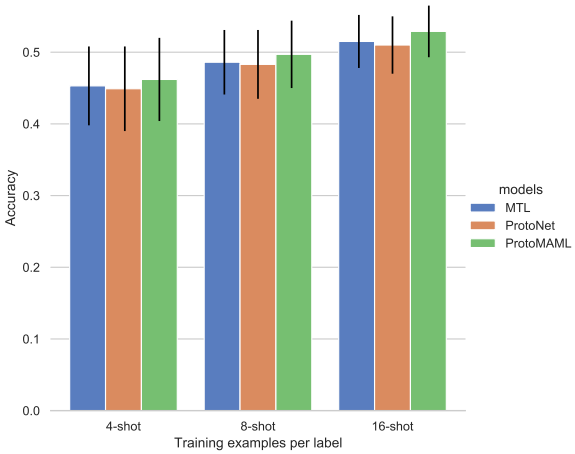

Figure 3: Classification performance averaged over all testing datasets for 4, 8- and 16-shot setting. Black vertical dashes are average standard deviations over multiple runs.

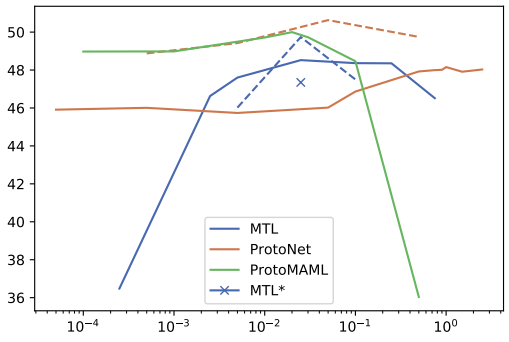

Figure 4: Differences in accuracy while varying the learning rate for 4/8/16-shot testing (averaged). Dotted lines are shown for models using ProtoMAML's initial parameters. MTL* uses the prototypes to initialise its classifier as ProtoMAML does.

5.4 Carousel Experiments

See Figure 5 for results. All models seem to rely on the Emotion prediction task the most. This is arguably to be expected; we evaluate on Sentiment Analysis, and some of the emotions explicitly denote happiness and anger/sadness, explaining the strong effect of holding out the Emotion tasks. For all other tasks, a positive or negative influence appears to be mostly model-dependent. Possibly, the lower amount of beneficial tasks for the meta-models indicate that on their loss surfaces, many of these tasks' optima are far removed to the optimum of the Sentiment Analysis task. In this respect, removing such tasks during training could "relax the pull" of that task's gradients pointing away from

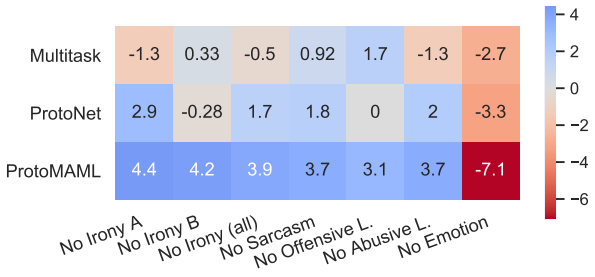

Figure 5: Change of accuracy on Sentiment Analysis when task X is not in the training tasks compared to when using all tasks. The redder the more the performance drops when that task is missing during training. Blue denotes an increase in performances.

Sentiment Analysis's optimum, and thus result in higher performance. This might also indicate that Sentiment Analysis is not as suitable for representing generalizability. However, these experiments were not run with the optimal hyperparameters. Therefore, they might not reflect the actual interdependencies of the tasks and thus have to be handled with caution. In this respect, more extensive testing with multiple test tasks might allow us to draw stronger conclusions.

5.5 Euclidean vs. Cosine Distance

In (Snell et al.), the original paper introducing prototypical networks, the authors use the Euclidean distance to calculate the distance of each query sample to the prototypes. However, it is often beneficial for high-dimensional spaces to instead use the cosine distance, since it only incorporates the angle between two vectors and is not dependent on their lengths and directions. We explore this further by conducting a small experiment, where we train one ProtoNet version with the Euclidean distance, and one with the cosine distance, and compare their performance. Although the accuracies in Figure 6 are comparable we observe that the standard deviation in Figure 7 is consistently lower when using the cosine similarity which we attribute to it being better suited to the high-dimensional embedding space of the ProtoNet.

6 Conclusion & Future Work

In this paper we have applied meta learning in the form of a Prototypical Network and a ProtoMAML model to social meaning tasks in a few-shot setup. We compared both models to a multitask baseline and found the ProtoMAML to be the most successful at adapting quickly to our unseen testing tasks.

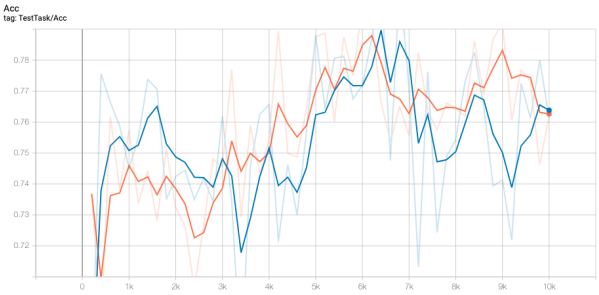

Figure 6: Accuracy on validation task during training on ProtoNet models with different distance metric: orange for euclidean distance and blue for cosine similarity.

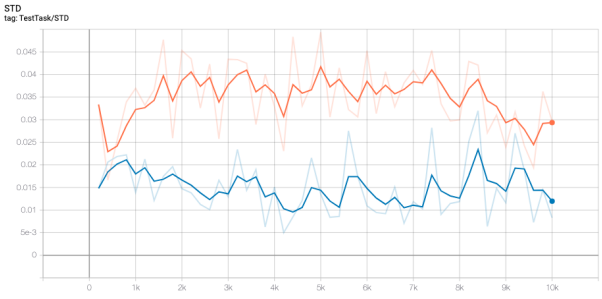

Figure 7: Standard deviation on validation task during training on ProtoNet models with different distance metric: orange for euclidean distance and blue for cosine similarity.

In a number of ablation experiments, we identified the source of the ProtoMAML's success to be its superior learning paradigm, leading to a better starting point on the loss surface when optimizing for a before-unseen task. Furthermore, we tested the interdependence of the tasks and found the benefit of their combinations to be highly model-dependent. On the fine-grained irony task, all the models barely achieved more than random performance indicating that the domain of social meaning and pragmatics is very hard to capture, especially since essential cues like tone of voice or facial expression cannot be conveyed in text.

In the future, we would like to increase the number of tasks and take the relation of the training tasks more into account. In that regard, we would also like to extend the analysis of the carousel tasks to test every combination of train and test tasks, to gain a deeper understanding of which combination of social meaning tasks is beneficial during training, especially for the ProtoMAML.

References

Semeval-2015, task 10: Sentiment analysis in twitter. http://alt.qcri.org/semeval2015/task10/#.

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

Cristian Danescu-Niculescu-Mizil, Moritz Sudhof, Dan Jurafsky, Jure Leskovec, and Christopher Potts. 2013. A computational approach to politeness with application to social factors. In *Proceedings of 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. Association for Computational Linguistics.

Second Workshop on Figurative Language Processing. 2020. Shared task on sarcasm detection. Data retrieved from https://github.com/EducationalTestingService/sarcasm/releases.

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*, ICML'17, page 1126–1135. JMLR.org.

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. Association for Computational Linguistics.

Jiatao Gu, Yong Wang, Yun Chen, Victor O. K. Li, and Kyunghyun Cho. 2018. Meta-learning for low-resource neural machine translation. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Xu Han, Hao Zhu, Pengfei Yu, Ziyun Wang, Yuan Yao, Zhiyuan Liu, and Maosong Sun. 2018. FewRel: A large-scale supervised few-shot relation classification dataset with state-of-the-art evaluation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4803–4809, Brussels, Belgium. Association for Computational Linguistics.

Cynthia Van Hee, Els Lefever, and Veronique Hoste. 2018. SemEval-2018 task 3: Irony detection in english tweets. In *Proceedings of The 12th Interna-*

tional Workshop on Semantic Evaluation. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. Cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

Ilya Loshchilov and Frank Hutter. 2017. Fixing weight decay regularization in adam. *CoRR*, abs/1711.05101.

Fei Mi, Minlie Huang, Jiyong Zhang, and Boi Faltings. 2019. Meta-learning for low-resource natural language generation in task-oriented dialogue systems. *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*.

Saif Mohammad, Felipe Bravo-Marquez, Mohammad Salameh, and Svetlana Kiritchenko. 2018. SemEval-2018 task 1: Affect in tweets. In *Proceedings of The 12th International Workshop on Semantic Evaluation*. Association for Computational Linguistics.

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.

Jake Snell, Kevin Swersky, and Richard S. Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems 30*.

Lukas Stappen, Fabian Brunn, and Björn Schuller. 2020. Cross-lingual zero- and few-shot hate speech detection utilising frozen transformer language models and axel.

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

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

Cynthia Van Hee, Els Lefever, and Véronique Hoste. 2018. SemEval-2018 task 3: Irony detection in English tweets. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 39–50. Association for Computational Linguistics.

Alex Wang, Jan Hůla, Patrick Xia, Raghavendra Pappagari, R. McCoy, Roma Patel, Najoung Kim, Ian Tenney, Yinghui Huang, Katherin Yu, Shuning Jin, Berlin Chen, Benjamin Durme, Edouard Grave, Ellie Pavlick, and Samuel Bowman. 2019. Can you tell me how to get past sesame street? sentence-level pretraining beyond language modeling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4465–4476.

Zeerak Waseem and Dirk Hovy. 2016. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In *Proceedings of the NAACL Student Research Workshop*, pages 88–93. Association for Computational Linguistics.

Chuhan Wu, Fangzhao Wu, Sixing Wu, Junxin Liu, Zhigang Yuan, and Yongfeng Huang. 2018. THU_NGN at SemEval-2018 task 3: Tweet irony detection with densely connected LSTM and multi-task learning. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 51–56. Association for Computational Linguistics.

Dani Yogatama, Cyprien de Masson d'Autume, Jerome Connor, Tomas Kocisky, Mike Chrzanowski, Lingpeng Kong, Angeliki Lazaridou, Wang Ling, Lei Yu, Chris Dyer, and Phil Blunsom. 2019. Learning and evaluating general linguistic intelligence.

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.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. Predicting the type and target of offensive posts in social media. 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 1415–1420. Association for Computational Linguistics.

Dataset	Task	Classes	Training size	Validation Size	Test Size
SemEval	Emotion Classification	11	6,838	886	3,259
SarcasmDetection	Sarcasm Detection	2	4,018	510	472
OLID/Offenseval	Offensive Language	2	13,241	-	860
SemEval2015	Sentiment Classification	2	1,908	409	409
SemEval2018	Task 3A: Irony	2	2,671	573	573
SemEval2018	Task 3B: Irony	4	2,671	573	573
WaseemHovy	Abusive language	3	11,341	2,430	2,431
Stanford Politeness	Politeness	3	3,047	653	653

Table 1: Datasets

	MTL	ProtoNet	ProtoMAML
BERT lr	$5e^{-5}$	$5e^{-5}$	$5e^{-5}$
MLP lr	$5e^{-5}$	$1e^{-4}$	$1e^{-4}$
Batch size	32	32	16
Iterations	$10e^{3}$	$10e^{3}$	$3e^{3}$
Inner optimizer	-	-	SGD
Inner lr	-	-	$1e^{-3}$
Meta batch size	-	-	32

Table 2: Selected hyperparameters for all models.

Model	Task	K	Accuracy	Std
MTL	IronySubtaskA	4	53.22%	0.032
MTL	IronySubtaskB	4	34.02%	0.056
MTL	Abuse	4	49.15%	0.079
MTL	Politeness	4	44.78%	0.055
MTL	IronySubtaskA	8	56.36%	0.027
MTL	IronySubtaskB	8	35.48%	0.046
MTL	Abuse	8	54.49%	0.072
MTL	Politeness	8	48.15%	0.034
MTL	IronySubtaskA	16	56.74%	0.033
MTL	IronySubtaskB	16	39.19%	0.032
MTL	Abuse	16	59.47%	0.043
MTL	Politeness	16	50.48%	0.039
MTL	*	*	48.46%	0.046
ProtoNet	IronySubtaskA	4	51.99%	0.032
ProtoNet	IronySubtaskB	4	33.76%	0.060
ProtoNet	Abuse	4	50.84%	0.086
ProtoNet	Politeness	4	42.91%	0.058
ProtoNet	IronySubtaskA	8	55.52%	0.031
ProtoNet	IronySubtaskB	8	35.60%	0.047
ProtoNet	Abuse	8	55.25%	0.080
ProtoNet	Politeness	8	46.85%	0.035
ProtoNet	IronySubtaskA	16	56.86%	0.037
ProtoNet	IronySubtaskB	16	38.96%	0.040
ProtoNet	Abuse	16	59.64%	0.050
ProtoNet	Politeness	16	48.65%	0.032
ProtoNet	*	*	48.07%	0.049
ProtoMAML	IronySubtaskA	4	54.37%	0.041
ProtoMAML	IronySubtaskB	4	33.39%	0.056
ProtoMAML	Abuse	4	53.34%	0.085
ProtoMAML	Politeness	4	43.85%	0.050
ProtoMAML	IronySubtaskA	8	56.91%	0.030
ProtoMAML	IronySubtaskB	8	35.67%	0.042
ProtoMAML	Abuse	8	58.89%	0.076
ProtoMAML	Politeness	8	47.20%	0.039
ProtoMAML	IronySubtaskA	16	57.98%	0.032
ProtoMAML	IronySubtaskB	16	39.41%	0.039
ProtoMAML	Abuse	16	64.31%	0.033
ProtoMAML	Politeness	16	50.03%	0.039
ProtoMAML	*	*	49.61%	0.047

Table 3: Reported performance for k-shot testing on 32 episodes for 3 runs averaged.

Model	Description	Avg. acc.	Avg. Std
MTL	Initialised from ProtoMAML	49.74%	0.04875
ProtoNet	Initialised from ProtoMAML	50.63%	0.0522
MTL*	Classifier from centroids	47.35%	0.0474

Table 4: Extra experiments.