FEW-SHOT NATURAL LANGUAGE PROCESSING BY META-LEARNING WITHOUT LABELED DATA

A Dissertation Presented

by

TRAPIT BANSAL

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

February, 2022

Robert and Donna Manning College of Information and Computer Sciences

FEW-SHOT NATURAL LANGUAGE PROCESSING BY META-LEARNING WITHOUT LABELED DATA

A Dissertation Presented

by

TRAPIT BANSAL

Approved as to style and content by:

Andrew K. McCallum, Chair

Kyunghyun Cho, Member

Subhransu Maji, Member

Patrick Flaherty, Member

Mohit Iyyer, Member

James Allan, Chair of the Faculty
Robert and Donna Manning College of Information and Computer Sciences

DEDICATION

To Mom, for teaching me more than she knew.

To Dad, for nurturing curiosity in me.

To Akansha, for believing in me when I didn't.

ACKNOWLEDGMENTS

This long and winding journey wouldn't have been possible without the help, support, and encouragement from a number of people. First and foremost, I will always be grateful to have Andrew McCallum as my advisor. Andrew's patient, loving, yet rigorous, style of advising has been a gift that made this journey truly enjoyable. While he helped me grow as a researcher, many of the lessons I learned from Andrew will also guide me throughout my life in my interactions with the research community, colleagues, friends, and family. I look forward to his continued support and mentorship in the next steps of my career.

I had the opportunity to do multiple internships throughout my Ph.D. and found some truly inspirational mentors and colleagues during them. I am grateful to have worked with Ilya Sutskever, whose creativity, passion, and grit have been a source of great inspiration. I am also grateful to Tsendsuren Munkhdalai, Da-Cheng Juan, Sujith Ravi, and Tong Wang for their support and mentorship, both during and after internships.

I am thankful to the thesis committee members for their valuable time and feedback on this thesis. I have had some amazing colleagues and lab members throughout the Ph.D. Among them, David Belanger, Arvind Neelakantan, Luke Vilnis, Pat Verga, and Nicholas Monath deserve special mention for being very generous with their time and help. I also want to acknowledge some people who helped me start my Ph.D. journey. Chiranjib Bhattacharyya, Ravi Kannan, and Mrinal Das provided me an opportunity to learn to do machine learning research that set me on this path and for which I am very grateful.

Finally, I want to acknowledge my family for their never-ending support, without which none of this was possible. My mom and dad, for loving me beyond measure and always providing me with the best of everything, much more than they themselves ever had. My brother, for being my support whenever I needed it. My wife, for being there through all the ups and downs, for always believing in me, and for inspiring me to always do better than yesterday.

ABSTRACT

FEW-SHOT NATURAL LANGUAGE PROCESSING BY META-LEARNING WITHOUT LABELED DATA

FEBRUARY, 2022

TRAPIT BANSAL

B.Sc & M.Sc., INDIAN INSTITUTE OF TECHNOLOGY KANPUR

M.S., UNIVERSITY OF MASSACHUSETTS AMHERST

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Andrew K. McCallum

Humans show a remarkable capability to accurately solve a wide range of problems efficiently – utilizing a limited amount of computation and experience. Deep learning models, by stark contrast, can be trained to be highly accurate on a narrow task while being highly inefficient in terms of the amount of compute and data required to reach that accuracy. Within natural language processing (NLP), recent breakthroughs in unsupervised pretraining have enabled reusable models that can be applied to many NLP tasks, however, learning of new tasks is still inefficient. This has led to research on few-shot learning, where the goal is to generalize to new tasks with very few labeled instances. Meta-learning, or learning to learn, treats the learning process itself as a learning problem from data with the goal of learning systems that can generalize to new tasks efficiently. This has the potential to produce few-shot learners that can accurately solve a wide range of new tasks. However, meta-learning requires a distribution over tasks with relevant labeled data that can be difficult to obtain, severely limiting the practical utility of meta-learning methods. In this dissertation, we develop methods to enable large-scale meta-learning from unlabeled text data and improve the few-shot generalization ability of NLP models.

We contribute methods that propose tasks synthetically created from unlabeled text, allowing for a large task distribution for meta-learning. This leads to rapid learning of new tasks by meta-learning from millions of self-supervised tasks and minimizes the train-test mismatch in few-shot learning by optimizing the pre-training directly for future fine-tuning with a few examples. Since real-world applications of NLP require learning diverse tasks with different numbers of classes, we first introduce an optimization-based meta-learning method that can learn from multiple NLP classification tasks with any number of classes. We then leverage the proposed self-supervised approach to create meta-training tasks, with a diverse number of classes, and meta-train models for few-shot learning using this task distribution. This leads to better representation learning, learning key hyper-parameters like learning rates, can be combined with supervised tasks to regularize supervised meta-learning, and leads to accurate few-shot learning on a diverse set of NLP classification tasks. We further explore the space of self-supervised tasks for meta-learning by considering important aspects like task diversity, difficulty, type, domain, and curriculum, and investigate how they affect meta-learning performance. Our analysis shows that all these factors meaningfully alter the task distribution, some inducing significant improvements in downstream few-shot accuracy of the meta-learned models.

Our findings yield accurate and efficient meta-learning methods that improve few-shot generalization to diverse tasks and should enable many future applications of meta-learning in NLP, such as hyper-parameter optimization, continual learning, efficient learning, learning in low-resource languages, and more.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 Motivation and Contributions

Neural models for natural language processing (NLP) have seen consistent improvement in their applicability and accuracy on many tasks. However, learning such models is still inefficient in terms of the amount of data required for good performance and their ability to generalize efficiently to new tasks and new domains. Meta-learning, or learning to learn, considers the problem of utilizing experience from previously seen tasks in order to more efficiently learn new tasks. An example of such an approach is to utilize a collection of tasks to learn a model initialization that enables quick learning of new tasks with few steps of gradient descent. While this has led to research on few-shot learning, especially in the domains of computer vision and reinforcement learning, it's application to NLP problems has been limited. Moreover, meta-learning typically requires a distribution over relevant diverse tasks with labeled data which can be difficult to obtain for many NLP applications, severely limiting its practical utility. In this dissertation, we develop practically usable meta-learning methods for NLP that can leverage unlabeled data for learning and improve the generalization ability of neural NLP models to new tasks and new domains with very few labeled examples.

We consider optimization-based meta-learning methods (Finn et al., 2017), which learn an initialization as well as a gradient-based optimizer for efficient adaptation to new tasks. Such methods leverage a useful inductive bias of gradient-based learning without compromising on representational capacity and are highly suitable for learning of new tasks. However, since NLP tasks vary greatly in their output spaces, existing methods are not applicable to learn from a diverse set of tasks. We first develop (Bansal et al., 2020a) an optimization-based method that can learn from tasks with diverse, potentially disjoint, output spaces and be applied to new unseen tasks at test time. This approach learns to generate task-specific parameters for few-shot learning, model initialization primed for few-shot learning, and key hyper-parameters of the optimization process employed at test time, such as per-layer learning rates, for efficient model adaptation with limited labeled data. We investigate self-supervised learning, multi-task learning, existing meta-learning methods, and our proposed approach for few-shot generalization to new tasks and domains.

While our supervised meta-learning approach improves few-shot accuracy, it is limited to utilizing a fixed set of supervised tasks which can be difficult to obtain and limits its practical utility. This can also lead to overfitting to the training task distribution further limiting generalization to new tasks. We thus consider meta-learning from unlabeled data. Recently, unsupervised pre-training of transformers using language modeling objectives (Devlin et al., 2019) has shown tremendous success in learning generalizable representations. Motivated by the success of these cloze-style objectives (Taylor, 1953), we propose an approach (Bansal et al., 2020b) to create separate multi-class classification tasks by gathering tokens-to-be blanked from among only a handful of vocabulary terms. This provides a large, rich, meta-learning task distribution from unlabeled corpora, with number of tasks exponential in the size of the vocabulary, enabling large-scale self-supervised meta-learning of NLP models. In addition to the advantages of the supervised meta-learning approach, self-supervised meta-learning enables better representation learning, and ameliorates meta-overfitting when combined with supervised tasks for meta-learning. We show that this approach improves few-shot generalization over language-model pre-training

(Devlin et al., 2019) or multi-task learning (Caruana, 1997; Liu et al., 2019a), while combining this with supervised tasks also improves over supervised meta-learning.

We further extend the idea of self-supervised task distributions by exploring the space of tasks from the prespectives of task diversity, difficulty, type of task, domain of tasks and task curriculum during meta-learning. The previously proposed self-supervised approach relied on random sampling of words for task generation. This can be wasteful in terms of utilization of unlabeled text for meta-training, leads to easy training tasks and limits diversity in the training task distribution. We propose modifications to task sampling that further enrich this task distribution, leading to more efficient meta-training. We also contrast the cloze-style approach with another approach to task creation that is based on clustering sentence embeddings. We find that the sentence clustering approach is sub-optimal compared to the cloze-style approaches proposed in this work. Moreover, an interesting aspect of being able to synthetically generate tasks is that we can control the order in which tasks are presented during training. We explore curriculum-based learning (Bengio et al., 2009) over self-supervised tasks in order to make meta-training more efficient. We compare and contrast all these proposed self-supervised task distributions for their utility in improving few-shot learning by meta-learning models on these task distributions. Our analysis shows that all of these considerations meaningfully alter task distributions and induce significant changes in downstream few-shot accuracy.

The findings presented in this thesis improve the generalization of NLP models to new tasks and new domains with very little labeled data. The proposed self-supervised approaches to generate meta-learning tasks remedy a long-standing problem of lack of sufficient training data for meta-learning methods in NLP and have further implications for developing meta-learning methods for other meta-problems in NLP, such as hyper-parameter optimization, continual learning, efficient learning, architecture search, learning in low-resource languages, and more.

The thesis is organized as follows. Chapter 2 discusses relevant background and related work for the content presented. Chapter 3 presents a meta-learning method that enables few-shot learning across tasks with diverse number of classes and establishes the evaluation methodology used throughout the dissertation. Chapter 4 then introduces the self-supervised approach to create task distributions and evaluates its utility in meta-learning models for few-shot learning under different settings. Chapter 5 takes these ideas further and explores diverse distributions of self-supervised tasks for meta-learning. Then chapter 6 takes a retrospective look at the methods presented in the thesis to compare and contrast them with some contemporaneous models like GPT-3 (Brown et al., 2020). Finally, chapter 7 concludes with a discussion of ideas for future research directions.

1.2 Declaration of Published Work

The main contributions of the thesis appeared in the following papers.

- Trapit Bansal, Rishikesh Jha, and Andrew McCallum. 2020a. Learning to few-shot learn across diverse natural language classification tasks. In *Proceedings of the 28th International Conference on Computational Linguistics (COLING)*, pages 5108–5123

- Trapit Bansal, Rishikesh Jha, Tsendsuren Munkhdalai, and Andrew McCallum. 2020b. Self-supervised meta-learning for few-shot natural language classification tasks. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 522–534

- Trapit Bansal, Karthick Gunasekaran, Tong Wang, Tsendsuren Munkhdalai, and Andrew McCallum. 2021. Diverse distributions of self-supervised tasks for meta-learning in NLP. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5812–5824

Earlier work on other methods for learning from limited labeled data, using multi-task and distantly supervised learning, appeared in the following papers. We don't provide an in-depth treatment of these methods but they are briefly discussed in the midst of related literature, in the background section 2.2.

- Trapit Bansal, David Belanger, and Andrew McCallum. 2016. Ask the GRU: Multi-task learning for deep text recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 107–114

- Nathan Greenberg, Trapit Bansal, Patrick Verga, and Andrew McCallum. 2018. Marginal likelihood training of BiLSTM-CRF for biomedical named entity recognition from disjoint label sets. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2824–2829

- Trapit Bansal, Pat Verga, Neha Choudhary, and Andrew McCallum. 2020c. Simultaneously linking entities and extracting relations from biomedical text without mention-level supervision. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7407–7414

- Dung Thai, Raghuveer Thirukovalluru, Trapit Bansal, and Andrew McCallum. 2021. Simultaneously self-attending to text and entities for knowledge-informed text representations. In *Proceedings of the 6th Workshop on Representation Learning for NLP (RepL4NLP-2021)*, pages 241–247, Online. Association for Computational Linguistics

The following work was also completed during the Ph.D. While not directly related, these helped motivate the content presented in this thesis.

- Trapit Bansal, Arvind Neelakantan, and Andrew McCallum. 2017. RelNet: End-to-end modeling of entities & relations. *NeurIPS Workshop on Automated Knowledge Base Construction (AKBC)*

- Trapit Bansal, Jakub Pachocki, Szymon Sidor, Ilya Sutskever, and Igor Mordatch. 2018. Emergent complexity via multi-agent competition. In *International Conference on Learning Representations*

- Maruan Al-Shedivat, Trapit Bansal, Yura Burda, Ilya Sutskever, Igor Mordatch, and Pieter Abbeel. 2018a. Continuous adaptation via meta-learning in nonstationary and competitive environments. In *International Conference on Learning Representations*

- Trapit Bansal, Da-Cheng Juan, Sujith Ravi, and Andrew McCallum. 2019. A2N: Attending to neighbors for knowledge graph inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4387–4392

- Vaishnavi Kommaraju, Karthick Gunasekaran, Kun Li, Trapit Bansal, Andrew McCallum, Ivana Williams, and Ana-Maria Istrate. 2020. Unsupervised pretraining for biomedical question answering. In *Working Notes of CLEF 2020 - Conference and Labs of the Evaluation Forum*, volume 2696 of *CEUR Workshop Proceedings*. CEUR-WS.org

CHAPTER 2
BACKGROUND

We start by discussing technical background relevant to the contributions made in this thesis. This thesis develops methods for accurate and efficient learning in the few-shot setting, that is when target tasks have very limited labelled data for efficient supervised learning. To enable learning of Natural Language Processing (NLP) tasks, we leverage neural network models that learn to represent words and sequences of words as vectors. We start by discussing these models that are used for the methods developed in this thesis. Then we provide some background and related work on the more general problem of learning from limited data, of which few-shot learning is one extreme case. Finally, we lay out the meta-learning problem setup, discuss various different meta-learning methods, its connection to other fields in transfer learning and survey existing meta-learning work in NLP.

2.1 Neural Models for NLP

2.1.1 Input Representations

The input for the methods considered in this work is text, which is a sequence of words. The discrete nature of text poses challenges for neural models, compared to images or audio, as neural methods are better suited for optimization techniques in the continuous space. Thus, representing the input in a continuous representation space is often required to learn neural networks that can operate on textual data. For this purpose, it is common to map discrete symbols in text to distributed representations (Bengio et al., 2003) which are subsequently processed by a neural model for learning high-level features for a task. Collobert and Weston (2008) first demonstrated the utility of such end-to-end learning for multiple NLP tasks, as well as the importance of pre-training such word representations from unlabelled data which was explored in more detail in subsequent works (Mikolov et al., 2013).

The first step in the NLP pipeline maps each token in text, segmented though some tokenization method, to discrete ids. Let $[x_1, \ldots, x_L]$ be a sequence of L tokens, where each $x_i \in [V]$ is a word type from a vocabulary of size V. The first layer in neural NLP models is then a lookup table which maps token ids to vectors, also called *word embeddings*, through a lookup operation. This sequence of word embeddings, $[\tilde{x}_1, \ldots, \tilde{x}_L]$, is then mapped to contextualized token representations through multiple layers of a neural network. Note that the word embeddings are part of the parameters of the neural networks which can be pre-trained separately on unlabelled data to use as initialization or fixed representations (Collobert et al., 2011; Mikolov et al., 2013), trained or fine-tuned on the target task (Collobert and Weston, 2008), or learned in an end-to-end manner along with other parameters on unlabelled data through self-supervised learning to initialize task-specific learning (Howard and Ruder, 2018; Peters et al., 2018).

Sub-word encoding. Note that classic word tokenization, along with the restriction to a fixed vocabulary, leads to problems with out-of-vocabulary words. As such, character-level modeling has been considered where neural models directly operate on character sequence (Mikolov et al., 2012; Sutskever et al., 2011; Graves, 2013; Peters et al., 2018, among others). An alternative to character-level models that has been highly successful is to use sub-word tokenization, which are in-between character-level and word-level. Examples of these are byte-pair encoding (BPE) (Sennrich et al., 2016) and word-piece (Schuster and Nakajima, 2012). BPE methods specify a target sub-word vocabulary size. Then beginning with each individual character in the language as the vocabulary, they will iteratively combine pairs of vocabulary items to add new units to the vocabulary based on a score, such as frequency, until the target vocabulary size is reached. Word-piece is a type of BPE method that chooses new vocabulary unit as the one that maximizes the language model likelihood of the training data when added to the vocabulary. These method provide a trade-off between using character-level and word-level modeling, ameliorate the out-of-vocabulary problem have been shown to be effective for many NLP problems (Sennrich et al., 2016; Wu et al., 2016; Vaswani et al., 2017; Devlin et al., 2019; Radford et al., 2019; Liu et al., 2019b, among others). The methods introduced in this work will use word-piece encoding, unless otherwise noted, which is also used by the BERT model (discussed in 2.2.4).

2.1.2 Contextual Text Encoders

Given a sequence of word embeddings as input, neural network models are then used to obtain context-informed token embeddings, also called as contextualized token representations. A number of choices exist for contextual encoders, which provide different trade-offs for modeling sequences. Common choices include: (1) convolutional neural networks (CNN) (Waibel et al., 1989; Kalchbrenner et al., 2014) and variants like dilated convolutions (Yu and Koltun, 2015); (2) recurrent neural networks (RNN) like long short term memory (Graves, 2013), gated recurrent units (Cho et al., 2014) and others; (3) transformers (Vaswani et al., 2017). Among these, recurrent neural network variants had been the default choice for NLP with state-of-the-art performance which changed with introduction of the transformers in Vaswani et al. (2017). Transformers rely entirely on attention operations (Bahdanau et al., 2014) and do away with recurrence which allows better modeling of long-range dependencies in text. Transformers have been applied successfully to numerous NLP tasks, often yielding state-of-the-art performance. The neural models in this thesis are based on transformers, however the methods are applicable to any other contextual text encoder like CNN and RNN. Next we discus the transformer model in more detail.

2.1.2.1 Transformers

We provide a brief overview of the transformer architecture. The reader is referred to the original paper (Vaswani et al., 2017) as well as the guide "the illustrated transformer" (Alammar, 2018) for a more exhaustive background. The model takes a sequence of L words as input which are converted into word embeddings in the first layer, $[\tilde{x}_1, \ldots, \tilde{x}_N]$, as described before. Transformer has no innate notion of position and the model relies on positional embeddings which are added to the input word embeddings. The positional embeddings are also learned parameters of the model. Sometimes, the input text is divided into multiple segments, for instance in natural language inference problems, and in such cases a segment embedding is also added to the input (Devlin et al., 2019).

Transformer (Vaswani et al., 2017) is made up of multiple layers. Each Transformer layer, denoted transformer$_k$, has its own set of parameters and consists of two components: multi-head self-attention followed by a series of feed-forward operations with non-linearity. Multi-head self-attention comprises of H blocks of self-attention in each layer. Let the input representations at the j-th layer be the matrix $R^{(j-1)} \in \mathbb{R}^{L \times d_{in}}$ comprising of L rows of representations for each token from the previous layer. Each head projects this input matrix into key, query and value: $K^{(j)}, Q^{(j)}, V^{(j)}$ of dimension $\mathbb{R}^{L \times d_k}$. The query vector at each token then attends to every other token, by a dot product with the key vector which is called the attention score, and the result of the attention is the aggregated value vectors based on the attention scores.

$$A^{(j)} = \text{softmax}\left(\frac{Q^{(j)} K^{(j)^T}}{\sqrt{d_k}}\right) V^{(j)}$$

The outputs of the individual attention heads are concatenated and projected back to dimension d_k, to give the output of multi-head attention $S^{(j)}$. This is followed by layer normalization (Ba et al., 2016), and a residual connection: $\tilde{S}^j = LN(S^{(j)} + R^{(j-1)})$. This output of multi-head self-attention is then followed by two feed-forward layers, which consist of two linear projections with a gelu non-linearity (Hendrycks and Gimpel, 2016). Finally, this output of the feed-forward layer, P^j is followed by another residual connection with the output of multi-head attention and layer-normalization to give the final output for the j-th transformer layer: $R^{(j)} = LN(P^j + \tilde{S}^j)$. This contextualized token-level representation then feeds into the next layer until we get to the final layer.

2.2 Learning Neural NLP Models with Limited Labeled Data

The focus of this thesis is on efficient few-shot learning which is an extreme case of learning from limited human-labeled data or supervision. A variety of different methods have been proposed throughout the history of machine learning for such scenarios. In order to be able to effectively learn from limited supervision, all methods will assume access to some additional resource, be it unlabelled data or task-specific knowledge. We briefly discuss the major areas of research under this umbrella before delving into an in-depth treatment of meta-learning, which is the matter of this thesis.

2.2.1 Multi-Task Learning

While regular supervised learning involves training on a single task, in multi-task learning (Caruana, 1997) a model is jointly trained on a set of tasks, typically with some shared parameters across tasks. The idea here is to share structure and statistical strength across related tasks to benefit learning of each (or a subset) of the tasks. This is also known as joint learning. For learning tasks with limited data, multi-task learning helps by using other related tasks which potentially have more data than the target task (Caruana, 1996). Multi-task learning often introduces trade-off between the various tasks being learned jointly based on the size, difficulty and relevance of the tasks as compared to the target tasks of interest.

For neural models, the joint learning often involves sharing the text encoder and using task-specific heads (i.e. additional layers) on top of the shared representations. Jointly learning over multiple related tasks has been used effectively in NLP. Collobert et al. (2011) showed the benefit of such joint learning over many low-level NLP tasks. McCann et al. (2018) framed multiple NLP tasks as question answering for multi-task learning. Highly related tasks, such as different information extraction tasks like entity and relation extraction, are often benefited from joint training (Miwa and Bansal, 2016; Katiyar and Cardie, 2017; Bekoulis et al., 2018). In Bansal et al. (2020c), we present a system that jointly extracts entities and relations in biomedical documents without needing exhaustive mention-level supervision, enabling learning such methods in low-resource settings. In Greenberg et al. (2018), we present a method for jointly learning across multiple datasets with disjoint or overlapping label sets for the task of named entity recognition, to enable learning systems that can perform labeling in the union of labels of the different datasets.

Multi-task learning helps in better representation learning of shared representations across tasks that has implications for better generalization to new tasks (Caruana, 1996), i.e. in the learning to learn setting (Thrun and Pratt, 2012). The benefit of such representation learning has also been explored from a theoretical perspective (Maurer et al., 2016). In Bansal et al. (2016), we demonstrated that multi-task learning of text representations by jointly predicting observed user-item interactions and keywords associated with the text can be effective for recommending new text items for which there is no interaction history, known as the cold-start recommendation problem. Multi-task learning on top of pre-trained models (discussed in 2.2.4) has also been helpful for adapting to new domains with limited data (Liu et al., 2019a). Refer to Ruder (2019) for a more detailed discussion and applications in NLP.

2.2.2 Weakly-Supervised Learning

The motivation of weakly-supervised methods is to learn machine learning model from limited supervised data by leveraging noisier or higher-level supervision that can be more easily or cheaply obtained. Such supervision is often obtained through subject matter experts or domain-specific databases or knowledge graphs. Ratner et al. (2016, 2017b) proposed the paradigm of data programming, where users express domain heuristics as labeling functions for creating a large weakly-supervised training set for learning. Mann and McCallum (2010) proposed the generalized expectation criterion that enables learning generative models such that the marginal label distributions satisfy some pre-defined constraints that serve as domain knowledge. Refer to Ratner et al. (2017a) for a detailed list of related methods.

Distant supervision (Mintz et al., 2009) is a type of weakly-supervised method that has been quite useful for information extraction tasks in NLP. Here higher-level supervision for a task is obtained through an external database, such as a knowledge graph. Relation extraction is an example of this approach where supervision for training can be obtained by taking an entity-relation tuple from the knowledge graph and classifying entity-pair mentions in text to have that relation. Since such annotation is noisy, various methods like multi-instance learning are incorporated (Riedel et al., 2010) to learn robust models. In Bansal et al. (2020c), we used such distant supervision about entities and relations at the document-level. Such annotations are readily available in biomedical knowledge graphs, where relations between biomedical entities are annotated by humans with the source research article where that relation is identified, without explicitly identifying where in the article the relation is mentioned. We develop models to enable extraction of entity-relation graphs from such research text using this type of distant supervision without needing exhaustive mention-level annotations which are often unavailable and expensive to obtain.

2.2.3 Semi-Supervised Learning

In semi-supervised learning, we have a small labeled data and a much larger, often task or domain specific, unlabeled data. Such methods utilize assumption about smoothness, low dimensional structure, or distance metrics to leverage the unlabeled data. Refer to Chapelle et al. (2009) for a review and Van Engelen and Hoos (2020) for a more recent survey covering modern neural network methods. Semi-supervised methods are often divided into inductive and transductive methods.

Inductive methods aim to construct classifiers that can generate predictions on any point in the input space. Unlabelled data is used for training the classifier, however the predictions for new unseen examples is independent of each other once the training is completed. Many different inductive learning methods exits. Self-training is a type of pseudo-labeling that can be considered as a wrapper around supervised learning. It works by training a classifier on labeled data and iteratively pseudo-labeling unlabeled data using the trained classifier and re-training the classifier on labeled as well as the most confident pseudo labeled data. Yarowsky (1995) proposed self-training for word-sense disambiguation in documents. More recently, Du et al. (2020b) demonstrated self-training methods on pre-trained transformers coupled with a nearest-neighbor retrieval mechanism can be effective for learning from limited data. Multi-view co-training (Blum and Mitchell, 1998) is a related method where multiple classifiers are trained on distinct views of the data. Other methods include generative models like variational auto-encoders (Kingma and Welling, 2013), generative-adversarial networks Springenberg (2015), and others (Van Engelen and Hoos, 2020).

Transductive methods don't have a clear distinction between training and testing phase. They typically define a graph over all data points, both labeled and unlabeled, which encodes similarity between data points. The objective function then enforces that predictions on labeled data points should match the true labels while similar data points should get the same label. Graph-based methods (Zhu et al., 2003) have been developed for specific NLP problems (Subramanya et al., 2010; Goldberg and Zhu, 2006). Graph convolution networks (Kipf and Welling, 2016) have been proposed for semi-supervised learning of classification problems on network data. See Van Engelen and Hoos (2020) for other related methods.

2.2.4 Self-Supervised Learning

Recently, self-supervised learning has gained increased popularity and success in transfer learning scenarios. Contrastive learning aims at learning representations by maximising similarity and dissimilarity over instances organized into groups of similar and dissimilar instances. Many methods formulate this as a mutual information maximization problem (Oord et al., 2018; Hjelm et al., 2018), where the idea is to maximize mutual information between two different views if the same data. More generally, such methods learn from unlabeled data by proposing a self-supervised task, also called as pretext task, that enables learning general purpose representations which are useful for a variety of downstream task. A large number of such self-supervised methods have been proposed for computer vision (Jing and Tian, 2020). Within this literature, recent works Su et al. (2020); Zhai et al. (2019) have explored the question of how self-supervised learning benefits few-shot learning. The models trained on self-supervised tasks are often used as *initializations* for downstream tasks on which they are further fine-tuned using task-specific labelled data. This is related to early methods on pre-training of deep networks which were shown to be an effective *prior* for supervised learning (Erhan et al., 2010). Pre-training followed by fine-tuning can also be considered as an inductive semi-supervised learning method.

In NLP, word embedding training methods like Word2Vec (Mikolov et al., 2013) are some of the earliest examples of self-supervised learning. Logeswaran and Lee (2018) trained sentence representation through a contrastive learning approach where the pretext task consists of maximizing similarity of a sentence to the next sentence in the document as compared with other random sentences. Pre-training using language model objectives has been shown to learn useful model initialization. Language modeling refers to predicting the next word conditioned on the context. Howard and Ruder (2018) showed language model pre-training followed by fine-tuning to work remarkably well for text classification tasks. Peters et al. (2018) pre-train a bi-directional language model with character embeddings and show that the contextualized token embeddings from these models are suitable for many downstream tasks. Radford et al. (2019) train a left to right autoregressive transformer language model and evaluate it for zero-shot learning for many NLP tasks. Yang et al. (2019) train an autoregressive transformer language model under all possible permutations of factorizing the word-order.

Devlin et al. (2019) pre-trained a transformed model on a masked language modeling (MLM) task. MLM can be considered a generalized version of langauge modeling that consists of masking out random words in a sentence which are to be predicted by the model. The masked out words are replace by a unqiue [MASK] token. In addition, they introduce a next sentence prediction task that is similar to Logeswaran and Lee (2018). They pre-process the input text to add [CLS] token at the start of the sentence as well as a [SEP] token to separate multiple sentences. The [CLS] is to learn a sentence-level embedding for sentence classification tasks. Their trained model, BERT, showed improvements in downstream performance, by fine-tuning the model, on a diverse range of tasks. Other recent work (Liu et al., 2019b; Clark et al., 2019; Lan et al., 2019) improved on pre-training of the BERT model through various modifications in the objective or the model architecture.

Recently, Brown et al. (2020) trained transformer language models with *175 billion* parameters on lots of common crawl data. They showed that the trained language model are effective few-shot learners when used along with a human added prompt to predict the label. Their model can be considered a model-based meta-learner (discussed in 2.3). Finally, recent theoretical work (Arora et al., 2019) has investigated why such self-supervised learning helps in solving downstream tasks. In particular, Saunshi et al. (2020) study why language modeling helps solving downstream tasks and provide error bounds showing that low language modeling test perplexity translated to lower downstream classification error.

2.3 Meta-Learning

This thesis explores meta-learning methods to learn models for natural language processing (NLP) tasks. This chapter surveys relevant technical background related to meta-learning and a discussion of existing work on meta-learning for NLP. Since the focus is on supervised learning, we restrict the discussion to this setting and don't discuss extensions to reinforcement learning problems.

2.3.1 Notation and Meta-Learning Setup

In supervised learning of a task T, we are given a dataset $\mathcal{D}_T = \{(x_j, y_j)\}$ sampled from a distribution: $\{(x_j, y_j)\} \sim P(\mathcal{X})P_T(\mathcal{Y}|\mathcal{X})$, where $\mathcal{X}$ is the input space, $\mathcal{Y}$ is the output space and $P_T(\mathcal{Y}|\mathcal{X})$ is the task-specific distribution over the output space given input. The aim is then to learn a parameterized model $f_\theta : x_j \to \hat{y}_j$, with parameters θ, that maps inputs x_j with true label y_j to a prediction $\hat{y}_j$. Learning is performed by (approximately) minimizing the task-specific loss $\mathcal{L}_T(\theta, D_T)$:

$$\hat{\theta} := \arg\min_{\theta} \mathcal{L}_T(\theta, D_T) \tag{2.1}$$

Note that the empirical loss is a function of both the parameters θ and the data D_T for the task. Global optimization is often computationally infeasible for complex models, such as neural networks, but one typically finds an approximate solution using an optimization procedure with some hyper-parameters such as learning-rate and parameter initialization.

In supervised meta-learning, there is a meta goal of learning across multiple tasks in order to enable rapid learning of new tasks. This assumes a distribution over tasks $P(\mathcal{T})$. Meta-learning methods then consider an approximate solution of equation 2.1:

$$\hat{\theta} := g_\omega(\theta, D_T) \tag{2.2}$$

where g_ω is some learning algorithm, with parameters ω, that has inputs as the model parameters and the training data. For example, g_ω can be a few steps of an optimization procedure like stochastic gradient descent with a learning rate ω (Finn et al., 2017), or a recurrent neural network with parameters ω (Younger et al., 2001). Note that θ are parameters of the *base learner*, for example a neural network, that is used to solve a task T.

Since the goal of meta-learning is to learn how to learn, the learning algorithm is not pre-specified as in typical supervised learning and will be the outcome of the meta-learning method. This can be formalized as optimizing the following objective:

$$\min_{\omega,\theta} \overbrace{\mathbb{E}_{T\sim\mathcal{P}(\mathcal{T})}\left[\mathcal{L}^{meta}\left(\underbrace{g_\omega(\theta, D_T^{train})}_{\text{Inner Loop}}, D_T^{val}\right)\right]}^{\text{Outer Loop}} \tag{2.3}$$

where L^{meta} is the meta-objective that measures the performance of task-specific learning on task T, $(D^{train}, D^{val}) \sim T$ are training and validation sets used for task specific learning and the evaluation of the meta-objective, respectively. Note that D^{train} is also referred to as the *support* set and D^{val} as the *query* set. The task-specific learning is also referred to as the *inner loop*, while the across task learning using the meta-objective is referred as the *outer loop*.

Episodic Framework. Training a meta-learning model often takes place in episodes. In each episode, a batch of tasks is samples from the task distribution $\mathcal{P}(\mathcal{T})$. Each task T is accompanied by a training data D_T^{train}, a validation data D_T^{val}, and a task-specific loss $\mathcal{L}_T$. D_T^{train} is then used for the inner-loop in equation 2.3, while D_T^{val} is used to compute the meta-objective in the outer loop which evaluates the learning outcome of the inner loop. This is also referred to as *meta-training* stage. After the model is trained, it is applied on a set of new tasks from $\mathcal{P}(\mathcal{T})$ that were not part of meta-training. Here is model can use is learned learning procedure from the meta-training to adapt to the test task's training data. This is referred to as *meta-testing* stage. One can also hold out a set of tasks with their (D^{train}, D^{val}), for validation of the meta-training and choosing hyper-parameters of the meta-learning methods. This is referred to as *meta-validation*.

Few-Shot Classification. As a concrete application, that is also germane to this work, lets consider supervised meta-learning for few-shot classification. The episodic learning framework for few-shot classification (Vinyals et al., 2016; Finn et al., 2017) minimizes train/test mismatch, thus meta-training tasks consist of k-shot data for small values of k. Formally, consider N-way k-shot classification, where N is the number of classes in the task and k is the number of examples per class. We are given a set of M training tasks $\{T_1, \ldots, T_M\}$. In order to simulate k-shot learning during training, in each episode (i.e. a training step) a batch of training tasks is sampled. Each task T_i consists a training set $\mathcal{D}_i^{tr} \sim T_i$, consisting of only k examples (per label) of the task and a validation set $\mathcal{D}_i^{val} \sim T_i$, containing q other examples of the same task. The model is then trained on $\mathcal{D}_i^{tr}$ using the task loss (cross-entropy), and then tested on $\mathcal{D}_i^{val}$. The model's performance on $\mathcal{D}_i^{val}$, for example measured using the classification cross-entropy loss, is then used to adjust the model parameters and the learning algorithm in equation 2.3. Here the validation error of the sampled tasks serves as the training error for the meta-learning process. At the end of training, the model is then evaluated on a new task $T_{M+1} \sim P(T)$ where again the train set of T_{M+1} consists of only k examples per label, and the model can use it's learning procedure to adapt to the task T_{M+1}. It is then evaluated on new test examples from this new task T_{M+1}.

2.3.2 Meta-Learning Approaches

Meta-learning methods are often categorized into model-based, metric-based, and optimization-based. We briefly discuss each of these categories. For a more thorough review of various meta-learning methods and applications in domains of computer vision or reinforcement learning, please refer to Hospedales et al. (2020) and Huisman et al. (2020).

2.3.2.1 Model-based Meta-Learning

Model-based meta-learning methods consider directly learning models that solve the meta-learning problem. Such methods, learn a model f_θ, such as a black-box neural network, that has inputs as the entire training data $\mathcal{D}_T^{train}$ as well as a new input $x \in \mathcal{D}_T^{val}$ for which we will like to make predictions: $p(y|x, \mathcal{D}_T^{train}) = f_\theta(x, \mathcal{D}_T^{train})$. The model f typically maintains a stateful internal representation of the task, often in the form of an external or recurrent memory. Memory-augmented neural networks (MANNs) (Santoro et al., 2016) use an external memory for storing representations of instances and a neural network controller for interacting with that memory. They process the support data and new query examples in a sequence. Recurrent meta-learners

(Duan et al., 2016; Wang et al., 2016) have been proposed which model f as recurrent neural networks for reinforcement learning problems. Meta networks (Munkhdalai and Yu, 2017) consider an architecture that is divided into a base learner, which performs fast task-specific learning, and meta-learner, which stores meta information useful for solving tasks. Mishra et al. (2018) model f with 1D temporal convolutions and a soft attention mechanism (Vaswani et al., 2017), processing examples in a sequence similar to Santoro et al. (2016).

In summary, model-based meta-learning methods make minimal assumption on the training dynamics of the model and instead learn it in the model weights. However, they are often outperformed by metric-based (2.3.2.2) or optimization-based (2.3.2.3) methods which have stronger inductive bias that enables better generalization (see, for eg., Satorras and Estrach (2018) and Finn et al. (2017)). Moreover, since they take the entire training data as input, they are less scalable to larger datasets than other methods, struggle to embed larger datasets degrading performance (Finn, 2018), and are worse at generalizing to out-of-distribution tasks than optimization-based methods (Finn, 2018).

2.3.2.2 Metric-based Meta-Learning

Metric-based methods try to learn a similarity metric that enable comparing any two instances. Predictions for new instances of a task are made by comparing the instance to all the instances in the training set using the similarity metric. Examples of metric-based methods include: matching networks (Vinyals et al., 2016), prototypical networks (Snell et al., 2017), siamese networks (Koch et al., 2015), relation networks (Sung et al., 2018), and more (Satorras and Estrach, 2018; Shyam et al., 2017). We look at matching networks and prototypical networks, which are two canonical examples of the metric-based methods.

Matching Networks. Matching networks (Vinyals et al., 2016) learn a parameterized model to embed all instances in a metric space and use a similarity weighted combination of all task training labels to make predictions on new instances for a task. These were proposed for classification tasks. Explicitly, it learns attention weights $a(\hat{x}, x_i)$ between two instances $\hat{x}$ and x_i which is used to make predictions for $\hat{x}$ as: $\hat{y} = \sum_{i=1}^{m} a(\hat{x}, x_i) y_i$. The attention weights are computed by measuring the cosine similarity between instances in the learned embedding space:

$$a(x, x_i) = \frac{\exp(f(x)^T h(x_i))}{\sum_{j=1}^{m} \exp(f(x)^T h(x_j))}$$

where f and h are neural networks with potentially shared parameters. Vinyals et al. (2016) explore extensions where these networks are also conditioned on the entire support set using an LSTM (Hochreiter and Schmidhuber, 1997).

Prototypical Networks. Prototypical networks (Snell et al., 2017) extends matching network, where the idea is to learn embeddings of instances such that they cluster around a single prototype representation for each class. They learn to embed data-points in an embedding space and use the few-shot training data as support to construct per class prototypes in this embedding space as the centroid representation for each class. The classification label for any new data point is then obtained as the class of the prototype that is closest to this data point in the embedding space. Concretely, the training dataset in an episode is first partitioned based on the class labels, $C_n = \{x_j | y_j = n\}$ where $n \in [N]$ and N is the number of classes in the current task. Then f_θ acts a neural encoder for data points in each C_n to generate a prototype for the n-th class as $\tilde{c}_n$. Now, given a query point $x^* \in \mathcal{D}_i^{val}$, the probability that x^* belongs to class n is given by computing the distance to the class prototype and passing the results through a softmax:

$$p(y^*|x^*, \mathcal{D}_i^{tr}) = \underset{n\in\{1,\ldots,N\}}{\text{softmax}} \{-d(\tilde{c}_n, f_\theta(x^*))\}$$
$$\tilde{c}_n = \frac{1}{|C_n|} \sum_{x_j \in C_n} f_\theta(x_j)$$

where $d(\cdot,\cdot)$ is the euclidean distance (Snell et al., 2017).

Summary. Metric-based methods learn embeddings of instances as well as similarity metrics that enable classifying new instances through comparing it with existing training examples. Connecting to equation 2.3, the inner-loop corresponds to only a forward pass through the embedding network to generate per-instance embeddings (Vinyals et al., 2016) or per-class prototypes (Snell et al., 2017). θ, ω are the parameters of the embedding networks. The outer-loop consists of generating the predictions on the validation set using the matching method and the meta-objective is then a cross-entropy loss on the true validation labels. Such methods have a similarity-based inductive bias, can lead to interpretable predictions and can be fast to apply when the target tasks are very small. However, when test tasks are more distant than the meta-training tasks, the methods do not absorb new task information and performance can degrade. Moreover, when task size increases, these methods may become computationally more expensive due to pairwise comparisons.

2.3.2.3 Optimization-based Meta-Learning

Optimization-based methods explicitly optimize for fast learning, using gradient-based optimization either directly as an inductive bias (Finn et al., 2017) or as motivation to learn fast optimization methods (Andrychowicz et al., 2016; Ravi and Larochelle, 2017; Li and Malik, 2016). These are often formulated as bi-level optimization, equation 2.3, where a base learner is updated using task-specific updates in the inner-loop (for example using SGD), and at the outer-loop the meta-parameters which affect inner-loop optimization are optimized across tasks.

Andrychowicz et al. (2016) introduced an LSTM optimizer that proposes parameter updates, using the gradient of a task loss, to replace the updates in regular stochastic gradient descent. Ravi and Larochelle (2017) proposed LSTM meta-learners where the parameters of the base learner are the cell state of the LSTM. Updates to the cell state in LSMT thus update the parameters of the base learner, analogous to gradient descent. They meta-learn an initialization for the cell state as well as the parameters of the LSTM that enable rapid learning. Li and Malik (2016) cast optimization as a reinforcement learning problem. We next look at model-agnostic meta-learning which uses a simpler gradient-based approach that is more scalable than these methods and can perform better for few-shot learning (Finn et al., 2017).

Model-Agnostic Meta-Learning (MAML). MAML (Finn et al., 2017) is an approach to optimization based meta-learning where the goal is to find a good initial point for model parameters θ, which through few steps of gradient descent, can be adapted to yield good performance on a new task. Learning in MAML consists of an *inner loop*, which consists of gradient-based learning on the task-specific objective, and an *outer-loop* which refines the initial point in order to enable fast learning across the set of tasks. Thus, given a task T_i with training datasets $\mathcal{D}_i^{tr}$ sampled during an episode, MAML's inner loop adapts the model parameters θ as:

$$\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_i(\theta, \mathcal{D}_i^{tr}) \tag{2.4}$$

Typically, more than one step of gradient update are applied sequentially. The hyperparameter α can also be meta-learned in the outer-loop (Li et al., 2017). The model parameters θ are then trained by back-propagating through the inner-loop adaptation across tasks, with the meta-objective of minimizing the error on the respective task validation sets $\mathcal{D}_i^{val}$:

$$\theta \leftarrow \theta - \beta \sum_{T_i \sim P(\mathcal{T})} \mathcal{L}_i(\theta_i', \mathcal{D}_i^{val}) \tag{2.5}$$

More sophisticated routines for optimization, such as Adam (Kingma and Ba, 2014), can also be used here instead of vanilla SGD. Note that even though MAML is typically trained to generate a good initialization point for fast adaptation in few-shot setting, since the inner-loop adaptation for each task also employs gradient-descent for learning, it's performance can approach regular supervised learning in the limit of large data with more number of steps.

Other Optimization-based Methods. Note that MAML differentiates through the inner loop updates which requires computing second-order derivatives. Since this is computationally expensive, often first-order approximation is employed which ignores the second order terms (Finn et al., 2017). Rajeswaran et al. (2019) introduced iMAML, which does not need to differentiate through the entire optimization trajectory and only requires the final point for each optimization. Rusu et al. (2018) introduced Latent Embedding Optimization which learns a lower-dimensional embedding of the parameters of the base learner, which are in high dimension, to enable better optimization of the parameters in the lower dimension. Lee et al. (2019) propose to learn linear predictors as base learners over shared representations, in a bi-level optimization framework, showing improvements on few-shot image classification. Nichol and Schulman (2018) introduced Reptile which is a method similar to MAML but only learns the initialization of the model parameters. Reptile samples tasks in the inner loop to perform gradient-based optimization on the task-specific loss and then, in the outer loop, directly moves the initial weights in the direction of the averaged final point of the optimized task-specific weights. Bayesian extensions of MAML have also been proposed (Finn et al., 2018; Grant et al., 2018; Yoon et al., 2018).

Summary. Optimization-based methods have the inductive bias of using gradients of the task loss to make updates to the base learner such that it enables learning new tasks rapidly. MAML does this by learning an initialization point for the base learner that can solve new tasks with few steps of gradient descent. This closely resembles typical supervised learning in the inner loop. They have been shown to have the same representative power as recurrent meta-learners and can perform well on a wider distribution of tasks than seen at training time (Finn and Levine, 2018) (see also sec 2.3.4). MAML is agnostic to the choice of the base learner model and has been successfully applied on many meta-learning benchmarks in computer vision and reinforcement learning (Finn, 2018).

2.3.3 $P(\mathcal{T})$: Task Distribution for Meta-Learning

Note that all of the different approaches to meta-learning, discussed so far, require a distribution over tasks for learning. Learning typically requires a large number of tasks during training to enable generalization to new tasks and to help avoid overfitting to the training tasks. Moreover, the diversity of tasks in training will affect the generalization ability of models. The most dominant approach in Computer Vision to meta-learning is to create tasks from a fixed dataset consisting of many object classes. This is has lead to the creation of benchmarks like miniImageNet (Vinyals et al., 2016) which are widely used in meta-learning (Hospedales et al., 2020).

We summarize how a supervised task data-set is typically leveraged to create meta-training tasks (Vinyals et al., 2016). Assuming access to a supervised task with L classes, an N-way k-shot task is created by first sampling N classes, assuming $N << L$. Then for each of the N sampled classes, $(k + q)$ examples of each class is randomly sampled from the dataset and assigned a unique label in $\{1, \ldots, N\}$. The k examples for each label serve as the support set, while the q examples constitute the validation set described above. Note, that each task consists of a small subset of classes and the class to label (1 to N) assignment is random. This is important to ensure that the meta-learner utilizes the task-specific data for learning the task and does not *memorize* the sample to label assignments in the parameters of the model. In case the labels are assigned unique identifiable labels across tasks, then generalization to new tasks suffers severely as the learning process falls back to standard supervised learning (Yin et al., 2020a). Related to miniImageNet in computer vision, the FewRel (Han et al., 2018) dataset and benchmark was proposed. Here tasks are created from a fixed dataset of sentences annotated with 100 relation labels. The 100 relations are split into training, validation and test splits used for meta-learning. Note that both training and evaluation are typically performed on synthetically sampled tasks.

While utilizing a fixed task type can help meta-learning to generalize to new classes of the task, this has severe limitations for the practical utility of meta-learning to many problems of interest. First and foremost, this limits learning to a very specific type of task. Such issues have been brought to front recently in the computer vision literature (Triantafillou et al., 2019; Huang et al., 2019) and better benchmarks that combine multiple datasets have been proposed (Triantafillou et al., 2019). Second, this imposes limits on the size and diversity of tasks available for meta-learning for most practical scenarios. Moreover, this approach is not feasible for tasks that have limited number of classes to enable such sub-sampling of class labels, for example sentiment classification. This has a stark contrast with the related field of multi-task learning (Caruana, 1997) where it is often feasible to utilize many diverse tasks for learning (McCann et al., 2018) which are also useful in the learning to learn setting (Thrun and Pratt, 2012; Maurer et al., 2016). Moreover, while self-supervised learning from unlabeled data has also been shown to be highly beneficial for representation learning, utilizing such unlabeled data for meta-learning is still challenging.

Unsupervised meta-learning has been explored in computer vision (Hsu et al., 2019; Khodadadeh et al., 2019) and reinforcement learning (Gupta et al., 2018; Jabri et al., 2019). Hsu et al. (2019) cluster embeddings of images and subsample from the clustering to create tasks for image classification. They show improved generalization from such unsupervised meta-learning over other unsupervised representation learning. Khodadadeh et al. (2019) propose to use image-specific augmentations for 1-shot learning. Gupta et al. (2018) propose tasks for meta-RL using mutual information. Other methods (Metz et al., 2019) propose supervised learning of an unsupervised update rule that can lead to better performance on a set of target tasks. These typically require supervised data during training to evaluate the update rule and face computational challenges in learning. Among these, the approach to clustering embeddings (Hsu et al., 2019) has direct application to NLP, although it hasn't been explored. Note that such an approach requires a pre-trained model whose representations are used to generate the clustering. Thus, the tasks are only meaningful from the and learning to predict the same clustering might not help in learning useful representations beyond the pre-trained model. Indeed, we show in chapter 5 that such an approach is sub-optimal for NLP tasks.

2.3.4 Theory on Meta-Learning

Many recent works have studied theoretical properties of many meta-learning methods, most prominently the optimization-based MAML algorithm. Finn and Levine (2018) proved that deep representations combined with gradient descent can approximate any learning algorithm, showing that there is no particular representational advantage to using a black-box meta-learner, like recurrent neural networks, over such methods. Given similar representational capacity, they also empirically studied the benefit of gradient-based methods over black-box methods. They show that initializations learned by MAML are extremely resilient to over-fitting to tiny datasets, even when taking many more gradient steps than were used during meta-training, and are better suited for extrapolation beyond the distribution of tasks seen at meta-training time.

Convergence properties of MAML have also been studied (Wang et al., 2020a; Fallah et al., 2020; Wang et al., 2020b), characterizing the optimality gap in the convergence point attained by MAML. Other theoretical work (Khodak et al., 2019; Denevi et al., 2019) studied gradient-based meta-learning through the lens of online convex optimization, for an online meta-learning setup. Collins et al. (2020) studied why MAML outperforms empirical risk minimization, finding that MAML induces a global optimization landscape with optimal solution closer to the optimal solutions of the hard tasks, i.e. the tasks with loss functions that SGD traverses slowly, thereby outperforming ERM on the hard tasks without sacrificing easy task performance, since they can still be solved in a small number of SGD steps. Finally, there is work on how representation learning across many tasks helps reduce sample complexity of new tasks. Maurer et al. (2016) studied the benefit of multi-task representation learning in the learning to learn setting, that is for solving new tasks. Du et al. (2020b) extend their result and studied sample complexity reduction on learning new tasks by learning on diverse source tasks with a shared representation.

2.3.5 Relation to Other Fields

A closely related field to meta-learning is multi-task learning (Caruana, 1997), where a model is trained to jointly perform well on a fixed set of tasks. The idea here is to share structure and statistical strength across related tasks to benefit learning of each task. This approach can also be used for better representation learning of shared representations across tasks that have implications for better generalization to new tasks (Caruana, 1996), i.e. in the learning to learn setting (Thrun and Pratt, 2012). The benefit of such representation learning has also been explored from a theoretical perspective (Maurer et al., 2016). A key difference between multi-task learning and meta-learning is that meta-learning methods directly optimize for learning of new tasks, and thus also learn the learning process for each task in order to enable better generalization to new tasks.

Transfer learning methods aim at transferring knowledge learned from previous tasks in order to enable better learning of new tasks. As such it subsumes both multi-task and meta-learning methods. Unsupervised learning methods, such as pre-training followed by fine-tuning, are another form of transfer learning that has received a lot of recent interest (Liu et al., 2019a). An important consideration in transfer learning is how tasks relate to each other, which has explored in computer vision Achille et al. (2019) as well as NLP Vu et al. (2020). Refer to Pan and Yang (2009) for a survey of transfer learning methods and Ruder (2019) for an overview of transfer learning methods in NLP.

2.3.6 Meta-Learning for NLP

Meta-learning methods have been applied for specific applications in NLP, to improve few-shot or low-resource performance on that applications. Note that prior work mostly uses human labelled data to create tasks for meta-learning methods. Moreover, many works, specially on text classification, rely on synthetically generated tasks for *both training and evaluation* following the popular miniImageNet approach in computer vision. We categorize the work in this area based on the type of NLP tasks below.

Seq2Seq Tasks. This set of NLP applications consider seq2seq (Sutskever et al., 2014) models for tasks that involve natural language generation, such as machine translation (Bahdanau et al., 2014). Gu et al. (2018) used MAML for low resource machine translation, by meta-training on high resource language pairs, treating each language pair as a new task. They showed improvements over multi-lingual and transfer learning methods. Sharaf et al. (2020) apply a similar approach for domain adaptation across domains, where meta-training consists of some domains of labelled parallel data and meta-test contains new domains with small amounts of parallel data. Huang et al. (2018) use MAML for generating SQL queries from natural language, where a task is defined as a set of related examples with similar SQL type. Mi et al. (2019) learn a MAML-based model for task-oriented dialogue systems in low-resource settings. Madotto et al. (2019) apply MAML for personalizing dialogue agents. Kann et al. (2020) extend the approach of Gu et al. (2018) to learn morphological inflection for low-resource languages.

Classification Tasks. Chen et al. (2018) learn HyperLSTM (Ha et al., 2016) model in a multi-task setting across various sentiment classification domains and show improvements when transferring to a new domain. Yu et al. (2018) learn multiple metrics to handle few-shot classification. Their meta-training procedure involves first clustering source tasks followed by learning a distance metric per cluster. Geng et al. (2019) employ capsules and dynamic routing algorithm (Sabour et al., 2017) for generating class prototypes in a framework similar to prototypical networks. Han et al. (2018) introduced the few-shot relation classification benchmark which was constructed similar to the miniImageNet (Vinyals et al., 2016) benchmark in computer vision. The dataset consists of a total of 100 relations which are split into train and test relations for training meta-learning methods. They evaluated various meta-learning methods on this benchmark. Gao et al. (2019a); Sun et al. (2019) developed prototypical networks with attention for few-shot relation classification and showed improvements on the FewRel task. Gao et al. (2019b) introduced the FewRel 2.0 benchmark by adding the aspect of domain transfer, where the meta-test set is of a different domain, to enable better evaluation of generalization. Obamuyide and Vlachos (2019) use MAML for relation classification by learning across relational labels with large number of examples to generalize to labels with fewer examples. Dou et al. (2019) explored Reptile algorithm Nichol and Schulman (2018) for generalizing to low-resource tasks. They train their method on a subset of high resource GLUE tasks and evaluate on related low-resource GLUE tasks. Holla et al. (2020) study meta-learning methods for word-sense disambiguation.

Note that all of these methods require some supervised data to enable meta-learning. In contrast, in this thesis, we will develop methods that can meta-learn from completely unlabeled data for classification tasks.

CHAPTER 3

LEARNING TO FEW-SHOT LEARN ACROSS DIVERSE NATURAL LANGUAGE PROCESSING TASKS

Pre-trained transformer models have shown enormous success in improving performance on several downstream tasks. However, fine-tuning on a new task still requires large amounts of task-specific labeled data to achieve good performance. We consider this problem of learning to generalize to new tasks with a few examples as a meta-learning problem. While meta-learning has shown tremendous progress in recent years, its application is still limited to simulated problems or problems with limited diversity across tasks. In this chapter, we develop a novel method, LEOPARD, which enables optimization-based meta-learning across tasks with different number of classes, and evaluate different methods on generalization to diverse NLP classification tasks. LEOPARD is trained with the state-of-the-art transformer architecture and shows better generalization to tasks not seen at all during training, with as few as 4 examples per label. Across 17 NLP tasks, including diverse domains of entity typing, natural language inference, sentiment analysis, and several other text classification tasks, we show that LEOPARD learns better initial parameters for few-shot learning than self-supervised pre-training or multi-task training, outperforming many strong baselines, for example, yielding 14.6% average relative gain in accuracy on unseen tasks with only 4 examples per label.

3.1 Introduction

Learning to learn (Schmidhuber, 1987; Bengio et al., 1992; Thrun and Pratt, 2012) from limited supervision is an important problem with widespread application in areas where obtaining labeled data for training large models can be difficult or expensive. We consider this problem of *learning in k-shots* for natural language processing (NLP) tasks, that is, given k labeled examples of a new NLP task learn to efficiently solve the new task. Recently, self-supervised pre-training of transformer models using language modeling objectives (Devlin et al., 2019; Radford et al., 2019; Yang et al., 2019) has achieved tremendous success in learning general-purpose parameters which are useful for a variety of downstream NLP tasks. While pre-training is beneficial, it is not optimized for fine-tuning with limited supervision and such models still require large amounts of task-specific data for fine-tuning, in order to achieve good performance (Yogatama et al., 2019).

On the other hand, meta-learning methods have been proposed as effective solutions for few-shot learning. Existing applications of such meta-learning methods have shown improved performance in few-shot learning for vision tasks such as learning to classify new image classes within a similar dataset. However, these applications are often limited to simulated datasets where each classification label is considered a task. Moreover, their application in NLP has followed a similar trend (Han et al., 2018; Yu et al., 2018; Guo et al., 2018; Mi et al., 2019; Geng et al., 2019). Since the input space of natural language is shared across all NLP tasks, it is possible that a meta-learning approach generalizes to unseen tasks. We thus move beyond simulated tasks to investigate meta-learning performance on generalization outside the training tasks, and focus on a diverse task-set with different number of labels across tasks.

Model agnostic meta-learning (MAML) (Finn et al., 2017) is an optimization-based approach to meta-learning which is agnostic to the model architecture and task specification. Hence, it is an ideal candidate for learning to learn from diverse tasks. However, it requires sharing model parameters, including softmax classification layers across tasks and learns a single initialization point across tasks. This poses a barrier for learning across diverse tasks, where different tasks can have potentially disjoint label spaces. Contrary to this, multi-task learning (Caruana, 1997) naturally handles disjoint label sets, while still benefiting from sharing statistical strength across tasks. However, to solve a new task, multi-task learning would require training a new classification layer for the task. On the other hand, metric-based approaches, such as prototypical networks (Vinyals et al., 2016; Snell et al., 2017), being non-parametric in nature can handle varied number of classes. However, as the number of labeled examples increase, these methods do not adapt to leverage larger data and their performance can lag behind optimization-based methods.

We address these concerns and make the following contributions: (1) we introduce a MAML-based meta-learning method, **LEOPARD**[1], which is coupled with a parameter generator that learns to generate *task-dependent* initial softmax classification parameters for any given task and enables meta-learning across tasks with disjoint label spaces; (2) we train LEOPARD with a transformer model, BERT (Devlin et al., 2019), as the underlying neural architecture, and show that it is possible to learn better initialization parameters for few-shot learning than that obtained from just self-supervised pre-training or pre-training followed by multi-task learning; (3) we evaluate on generalization, with a few-examples, to NLP tasks not seen during training or to new domains of seen tasks, including *entity typing, natural language inference, sentiment classification, and various other text classification tasks*; (4) we study how meta-learning, multi-task learning and fine-tuning perform for few-shot learning of completely new tasks, analyze merits/demerits of parameter efficient meta-training, and study how various train tasks affect performance on target tasks. To the best of our knowledge, this is the first application of meta-learning in NLP which evaluates on test tasks which are significantly different than training tasks and goes beyond

[1]**L**earning to g**e**nerate s**o**ftmax **pa**rameters fo**r** **d**iverse classification

simulated classification tasks or domain-adaptation tasks (where train and test tasks are similar but from different domains).

3.2 Model

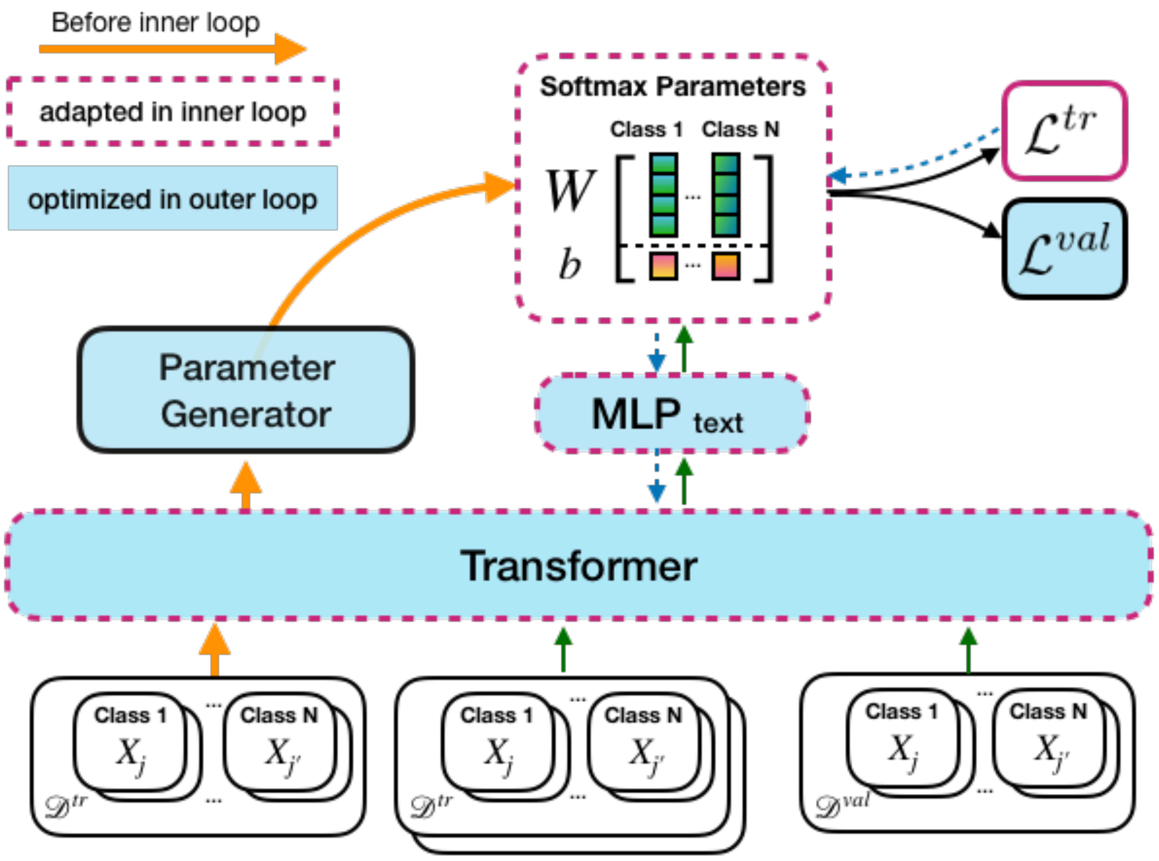


Figure 3.1. The proposed LEOPARD model. Input is first encoded using the Transformer. The first batch from the support set is passed through the parameter generator which learns a per-class set representation that is used to generate the initial softmax parameters. Subsequently, the support batches are used for adaptation of the generated parameters as well as the encoder parameters. Pink box (dashed) outline shows modules that are adapted in the inner loop, whereas blue boxes are optimized in the outer loop.

In this section, we describe our proposed method, LEOPARD, for learning new NLP classification tasks with k-examples. Fig. 3.1 shows a high-level description of the model. Our approach builds on the MAML framework and addresses some of its limitations when applied to a diverse set of tasks with different number of classes across tasks. Our model consists of three main components: (1) a shared neural input encoder which generates feature representations useful across tasks; (2) a softmax parameter generator *conditioned on the training dataset* for an N-way task, which generates the initial softmax parameters for the task; (3) a MAML-based adaptation method with a distinction between *task-specific parameters*, which are adapted per task, and *task-agnostic* parameters, which are shared across tasks, that can lead to parameter-efficient fine-tuning of large models. Full training algorithm is shown in Alg. 1.

3.2.1 Text Encoder

The input consists of natural language sentences, thus our models take sequences of words as input. Note that some tasks require classifying pairs of sentences (such as natural language inference) and phrases in a sentence (such as entity typing), and we discuss how these can also be encoded as a sequence in Section 3.3.1. We use a Transformer model (Vaswani et al., 2017) as our text encoder which has shown success for many NLP tasks. Concretely, we follow (Devlin et al., 2019) and use their BERT-base model architecture. We denote the Transformer model by f_θ, with parameters $\theta = \{\theta_1, \dots, \theta_{12}\}$ where θ_v are the parameters of layer v. Transformer takes a sequence of words $\mathbf{x}_j = [x_{j1}, \dots, x_{js}]$ as input (s being the sequence length), and outputs d-dimensional contextualized representations at the final layer of multi-head self-attention. BERT adds a special CLS token (Devlin et al., 2019) to the start of every input, which can be used as a sentence representation. We thus use this as the fixed-dimensional input feature representation of the sentence: $\tilde{x}_j = f_\theta([x_{j1}, \dots, x_{js}])$.

3.2.2 Generating Softmax Parameters for Task-specific Classification

Existing applications of MAML consider few-shot learning with a fixed N, i.e. the number of classes. This limits applicability to multiple types of tasks, each of which

Algorithm 1 LEOPARD

```
Require: set of $M$ training tasks and losses $\{(T_1, L_1), \ldots, (T_M, L_M)\}$, model parameters $\Theta = \{\theta, \psi, \alpha\}$, hyper-parameters $\nu, G, \beta$
    Initialize $\theta$ with pre-trained BERT-base;
 1: while not converged do
 2:     # sample batch of tasks
 3:     for all $T_i \in T$ do
 4:         $\mathcal{D}_i^{tr} \sim T_i$        # sample a batch of train data
 5:         $C_i^n \leftarrow \{x_j | y_j = n\}$        # partition data according to class labels
 6:         $w_i^n, b_i^n \leftarrow \frac{1}{|C_i^n|} \sum_{x_j \in C_i^n} g_\psi(f_\theta(x_j))$        # generate softmax parameters
 7:         $\mathbf{W}_i \leftarrow [w_i^1; \ldots; w_i^{N_i}]$;  $\mathbf{b}_i \leftarrow [b_i^1; \ldots; b_i^{N_i}]$
 8:         $\Phi_i^{(0)} \leftarrow \theta_{>\nu} \cup \{\phi, \mathbf{W}_i, \mathbf{b}_i\}$        # task-specific parameters
 9:         for $s := 0 \ldots G-1$ do
10:             $\mathcal{D}_i^{tr} \sim T_i$        # sample a batch of train data
11:             $\Phi_i^{(s+1)} \leftarrow \Phi_i^{(s)} - \alpha_s \nabla_\Phi \mathcal{L}_i(\{\Theta, \Phi_i\}, \mathcal{D}_i^{tr})$        # adapt task-specific parameters
12:         end for
13:         $\mathcal{D}_i^{val} \sim T_i$        # sample a batch of validation data
14:         $g_i \leftarrow \nabla_\Theta \mathcal{L}_i(\{\Theta, \Phi_i^{(G)}\}, \mathcal{D}_i^{val})$        # gradient of task-agnostic parameters on validation
15:     end for
16:     $\Theta \leftarrow \Theta - \beta \cdot \sum_i g_i$        # optimize task-agnostic parameters
17: end while
```

would require a different number of classes for classification. To remedy this, we introduce a method to generate *task-dependent* softmax parameters (both linear weights and bias). Given the training data, $\mathcal{D}_i^{tr} = \{(x_j, y_j)\}$, for a task T_i in an episode, we first partition the input into the N_i number of classes for the task (available in $\mathcal{D}_i^{tr}$): $C_i^n = \{x_j | y_j = n\}$, where $n \in [N_i]$. Now, we perform a non-linear projection on the representations of the x_j in each class partition obtained from the text-encoder, and obtain a set representation for class n:

$$w_i^n, b_i^n = \frac{1}{|C_i^n|} \sum_{x_j \in C_i^n} g_\psi(f_\theta(\mathbf{x}_j)) \tag{3.1}$$

where g_ψ is multi-layer perceptron (MLP) with two layers and *tanh* non-linearities, w_i^n is a l-dimensional vector and b_i^n is a scalar. w_i^n and b_i^n are the softmax linear weight and bias, respectively, for the class n:

$$\mathbf{W}_i = [w_i^1; \ldots ; w_i^{N_i}] \quad \mathbf{b}_i = [b_i^1; \ldots ; b_i^{N_i}] \tag{3.2}$$

Thus, the softmax classification weights $\mathbf{W}_i \in \mathcal{R}^{N_i \times l}$ and bias $\mathbf{b}_i \in \mathcal{R}^{N_i}$ for task T_i are obtained by row-wise concatenation of the per-class weights in equation 3.1. Note that encoder $g_\psi(\cdot)$ would be shared across tasks in different episodes.

Now, given the softmax parameters, the prediction for a new data-point $\mathbf{x}^*$ is given as:

$$p(y|\mathbf{x}^*) = \text{softmax}\{\mathbf{W}_i h_\phi(f_\theta(\mathbf{x}^*)) + \mathbf{b}_i\} \tag{3.3}$$

where $h_\phi(\cdot)$ is another MLP with parameters ϕ and output dimension l, and the softmax is over the set of classes N_i for the task.

Note that if we use $x^* \in \mathcal{D}_i^{val}$, then the model is a form of a prototypical network (Snell et al., 2017) which uses a learned distance function. However, this would limit the model to not adapt its parameters with increasing data. We next discuss how we learn to adapt using the generated softmax. It is important to note that *we do not introduce any task-specific parameters*, unlike multi-task learning (Caruana, 1997) which will require new softmax layers for each task, and the existing parameters are used to generate a good starting point for softmax parameters across tasks which can then be *adapted* using stochastic gradient (SGD) based learning.

3.2.3 Learning to Adapt Efficiently

Given the task-specific classification loss computed at an episode, MAML takes multiple steps of SGD on the same training set $\mathcal{D}_i^{tr}$, as in equation 2.4. We apply MAML on the model parameters, including the generated softmax parameters. However, the number of parameters in BERT is substantially high ($\sim$ 110 million) and it can be beneficial to adapt a smaller number of parameters (Houlsby et al., 2019; Zintgraf et al., 2019). We thus separate the set of parameters into *task-specific* and *task-agnostic.* For the transformer parameters for each layer $\{\theta_v\}$, we consider a threshold ν over layers, and consider $\theta_{\leq\nu} = \{\theta_v | v \leq \nu\}$ to be the parameters for first ν layers (closest to the input) and the rest of the parameters as $\theta_{>\nu}$. Then we consider $\theta_{\leq\nu}$ and the parameters ψ of the softmax generating function (equation 3.1) as the set of task-agnostic parameters $\Theta = \theta_{\leq\nu} \cup \{\psi\}$. These task-agnostic parameters Θ need to generalize to produce good feature representations and good initial point for classification layer *across tasks.* The remaining set of parameters for the higher layers of transformer, the input projection function in 3.3, and the softmax weights and bias *generated* in equation 3.2 are considered as the set of task-specific parameters $\Phi_i = \theta_{>\nu} \cup \{\phi, \mathbf{W}_i, \mathbf{b}_i\}$.

The task-specific parameters will be adapted for each task using SGD, as in equation 2.4. Note that MAML usually does gradient descent steps on the same meta-train batch $\mathcal{D}_i^{tr}$ for a task in an episode. However, since we use $\mathcal{D}_i^{tr}$ to generate the softmax parameters in equation 3.1, using the same data to also take multiple gradient steps can lead to over-fitting. Thus, we instead sample $G > 1$ meta-train batches in each episode of training, and use the subesequent batches (after the first batch) for adaptation. Task-specific adaptation in the inner loop does G steps of the following update, starting with $\Phi_i^{(0)} \leftarrow \Phi_i$, for $s := 0, \ldots, G-1$:

$$\Phi_i^{(s+1)} = \Phi_i^{(s)} - \alpha_s \, \mathbb{E}_{\mathcal{D}_i^{tr} \sim T_i} \left[\nabla_\Phi \mathcal{L}_i(\{\Theta, \Phi_i\}, \mathcal{D}_i^{tr}) \right] \tag{3.4}$$

Note that we only take gradient with respect to the task-specific parameters Φ_i, however the updated parameter is also a function of Θ. After the G steps of adaptation, the final point (which consists of parameters Θ and Φ^G) is evaluated on the validation set for the task, $\mathcal{D}_i^{val}$, and the task-agnostic parameters Θ are updated (as in equation 2.5) to adjust the initial point across tasks. Note that optimization of the task-agnostic parameters requires back-propagating through the inner-loop gradient steps and requires computing higher-order gradients. (Finn et al., 2017) proposed using a first-order approximation for computational efficiency. We use this approximation in this work, however we note that the distinction between task-specific and task-agnostic parameters can allow for higher order gradients when there are few task-specific parameters (for example, only the last layer).

Other Technical Details: For few-shot learning, learning rate can often be an important hyper-parameter and the above approach can benefit from also learning the learning-rate for adaptation (Li et al., 2017). Instead of scalar inner loop learning rates, it has been shown beneficial to have per-parameter learning rates that are also learned (Li et al., 2017; Antoniou et al., 2018). However, this doubles the number of parameters and can be inefficient. Instead, we learn a per-layer learning rate for the inner loop to allow different transformer layers to adapt at different rates. We apply layer normalization across layers of transformers (Vaswani et al., 2017; Ba et al., 2016) and also adapt their parameters in the inner loop. The number of layers to consider as task-specific, ν, is a hyper-parameter. We initialize the meta-training of LEOPARD from pre-trained BERT model which stabilizes training.

3.3 Experiments

Our experiments evaluate how different methods generalize to new NLP tasks with limited supervision. We focus on sentence-level classification tasks, including natural language inference (NLI) tasks which require classifying pairs of sentences as well as tasks like entity typing which require classifying a phrase in a sentence. We consider 17 target tasks[2]. Main results are in Sec. 3.3.3.

3.3.1 Training Tasks

We use the GLUE benchmark tasks Wang et al. (2018b) for training all the models. Such tasks are considered important for general linguistic intelligence, have lots of

[2]Code, trained model parameters, and datasets: `https://github.com/iesl/leopard`

supervised data for many tasks and have been useful for transfer learning Phang et al. (2018); Wang et al. (2018a). We consider the following tasks for training[3]: MNLI (m/mm), SST-2, QNLI, QQP, MRPC, RTE, and the SNLI dataset Bowman et al. (2015). We use the corresponding validation sets for hyper-parameter tuning and early stopping. For meta-learning methods, we classify between every pair of labels (for tasks with more than 2 labels) which increases the number of tasks and allows for more per-label examples in a batch during training. Moreover, to learn to do phrase-level classification, we modify SST (for all models) which is a phrase-level sentiment classification task by providing a sentence in which the phrase occurs as part of the input. That is, the input is the sentence followed by a separator token Devlin et al. (2019) followed by the phrase to classify. See Appendix A.1 for more details.

3.3.2 Evaluation and Baselines

Unlike existing methods which evaluate meta-learning models on sampled tasks from a fixed dataset Vinyals et al. (2016); Finn et al. (2017), we evaluate methods on real NLP datasets by using the entire test sets for the target task after using a sampled k-shot training data for fine-tuning. The models parameters are trained on the set of training tasks and are then fine-tuned with k training examples per label for a target test task. The fine-tuned models are then evaluated on the *entire test-set* for the task. We evaluate on $k \in \{4, 8, 16\}$. For each task, for every k, we sample 10 training datasets and report the mean and standard deviation, since model performance can be sensitive to the k examples chosen for training. In the few-shot setting it can be unreasonable to assume access to a large validation set Yu et al. (2018); Kann et al. (2019), thus for the fine-tuning step we tuned the hyper-parameters for all baselines on a held out validation task. We used SciTail, a scientific NLI task, and electronics

[3]We exclude WNLI since its training data is small and STS-B task since it is a regression task

domain of Amazon sentiment classification task as the validation tasks. We took the hyper-parameters that gave best average performance on validation data of these tasks, for each value of k. For LEOPARD, we only tune the number of epochs for fine-tuning, use the learned per-layer learning rates and reuse remaining hyper-parameters (see Appendix A.3).

We evaluate multiple transfer learning baselines as well as a meta-learning baseline. Note that most existing applications of few-shot learning are tailored towards specific tasks and don't trivially apply to diverse tasks considered here. We evaluate the following methods:

BERT$_{\textbf{base}}$: We use the cased BERT-base model Devlin et al. (2019) which is a state-of-the-art transformer Vaswani et al. (2017) model for NLP. BERT uses language model pre-training followed by supervised fine-tuning on a downstream task. For fine-tuning, we tune all parameters as it performed better on the validation task.

Multi-task BERT (MT-BERT): This is the BERT-base model trained in a multi-task learning setting on the set of training tasks. Our MT-BERT is comparable to the MT-DNN model of Liu et al. (2019a) that is trained on the tasks considered here and uses the cased BERT-base as the initialization. We did not use the specialized stochastic answer network for NLI used by MT-DNN. For this model, we tune all the parameters during fine-tuning.

MT-BERT$_{\textbf{softmax}}$: This is the multi-task BERT model above, where we only tune the softmax layer during fine-tuning.

Prototypical BERT (Proto-BERT): This is the prototypical network method Snell et al. (2017) that uses BERT-base as the underlying neural model. Following Snell et al. (2017), we used euclidean distance as the distance metric.

All methods are initialized with pre-trained BERT. All parameters of MT-BERT and Proto-BERT are also tuned during training. We don't compare with MAML Finn et al. (2017) as it does not trivially support varying number of classes, and show in ablations (3.3.4) that solutions like using zero-initialized initial softmax perform worse.

Implementation Details: Since dataset sizes can be imbalanced, it can affect multi-task and meta-learning performance. Wang et al. (2018a) analyze this in detail for multi-task learning. We explored sampling tasks with uniform probability, proportional to size and proportional to the square-root of the size of the task. For all models, we found the latter to be beneficial. All methods are trained on 4 GPUs to benefit from large batches. Best hyper-parameters, search ranges and data statistics are in Appendix.

3.3.3 Results

We evaluate all the models on 17 target NLP tasks. None of the task data is observed during the training of the models, and the models are fine-tuned on few examples for the target task and then evaluated on the entire test set for the task. For k-shot learning of tasks not seen at all during training, we observe, on average, relative gain in accuracy of 14.60%, 10.83%, and 11.16%, for $k = 4, 8, 16$ respectively.

3.3.3.1 Generalization Beyond Training Tasks

We use the following datasets (more details in Appendix): (1) entity typing: CoNLL-2003 Sang and De Meulder (2003), MIT-Restaurant Liu et al. (2013); (2) rating classification: we use the review ratings for each domain from the Amazon Reviews dataset Blitzer et al. (2007) and consider a 3-way classification based on the ratings; (3) text classification: social-media datasets from crowdflower[4].

Table 3.1 shows the performance. We can see that, on average, LEOPARD outperforms all the baselines, yielding significant improvements in accuracy. This shows LEOPARD's robustness to varying number of labels across tasks and across differ-

[4]https://www.figure-eight.com/data-for-everyone/

			Entity Typing				
	N	k	BERT$_{\text{base}}$	MT-BERT$_{\text{softmax}}$	MT-BERT	Proto-BERT	LEOPARD
CoNLL	4	4	50.44 ± 08.57	52.28 ± 4.06	**55.63** ± 4.99	32.23 ± 5.10	54.16 ± 6.32
		8	50.06 ± 11.30	65.34 ± 7.12	58.32 ± 3.77	34.49 ± 5.15	**67.38** ± 4.33
		16	74.47 ± 03.10	71.67 ± 3.03	71.29 ± 3.30	33.75 ± 6.05	**76.37** ± 3.08
MITR	8	4	49.37 ± 4.28	45.52 ± 5.90	**50.49** ± 4.40	17.36 ± 2.75	49.84 ± 3.31
		8	49.38 ± 7.76	58.19 ± 2.65	58.01 ± 3.54	18.70 ± 2.38	**62.99** ± 3.28
		16	69.24 ± 3.68	66.09 ± 2.24	66.16 ± 3.46	16.41 ± 1.87	**70.44** ± 2.89
			Text Classification				
Airline	3	4	42.76 ± 13.50	43.73 ± 7.86	46.29 ± 12.26	40.27 ± 8.19	**54.95** ± 11.81
		8	38.00 ± 17.06	52.39 ± 3.97	49.81 ± 10.86	51.16 ± 7.60	**61.44** ± 03.90
		16	58.01 ± 08.23	58.79 ± 2.97	57.25 ± 09.90	48.73 ± 6.79	**62.15** ± 05.56
Disaster	2	4	**55.73** ± 10.29	52.87 ± 6.16	50.61 ± 8.33	50.87 ± 1.12	51.45 ± 4.25
		8	**56.31** ± 09.57	56.08 ± 7.48	54.93 ± 7.88	51.30 ± 2.30	55.96 ± 3.58
		16	64.52 ± 08.93	**65.83** ± 4.19	60.70 ± 6.05	52.76 ± 2.92	61.32 ± 2.83
Emotion	13	4	09.20 ± 3.22	09.41 ± 2.10	09.84 ± 2.14	09.18 ± 3.14	**11.71** ± 2.16
		8	08.21 ± 2.12	11.61 ± 2.34	11.21 ± 2.11	11.18 ± 2.95	**12.90** ± 1.63
		16	13.43 ± 2.51	**13.82** ± 2.02	12.75 ± 2.04	12.32 ± 3.73	13.38 ± 2.20
Political Bias	2	4	54.57 ± 5.02	54.32 ± 3.90	54.66 ± 3.74	56.33 ± 4.37	**60.49** ± 6.66
		8	56.15 ± 3.75	57.36 ± 4.32	54.79 ± 4.19	58.87 ± 3.79	**61.74** ± 6.73
		16	60.96 ± 4.25	59.24 ± 4.25	60.30 ± 3.26	57.01 ± 4.44	**65.08** ± 2.14
Political Audience	2	4	51.89 ± 1.72	51.50 ± 2.72	51.53 ± 1.80	51.47 ± 3.68	**52.60** ± 3.51
		8	52.80 ± 2.72	53.53 ± 2.26	**54.34** ± 2.88	51.83 ± 3.77	54.31 ± 3.95
		16	**58.45** ± 4.98	56.37 ± 2.19	55.14 ± 4.57	53.53 ± 3.25	57.71 ± 3.52
Political Message	9	4	15.64 ± 2.73	13.71 ± 1.10	14.49 ± 1.75	14.22 ± 1.25	**15.69** ± 1.57
		8	13.38 ± 1.74	14.33 ± 1.32	15.24 ± 2.81	15.67 ± 1.96	**18.02** ± 2.32
		16	**20.67** ± 3.89	18.11 ± 1.48	19.20 ± 2.20	16.49 ± 1.96	18.07 ± 2.41
Rating Books	3	4	39.42 ± 07.22	44.82 ± 9.00	38.97 ± 13.27	48.44 ± 7.43	**54.92** ± 6.18
		8	39.55 ± 10.01	51.14 ± 6.78	46.77 ± 14.12	52.13 ± 4.79	**59.16** ± 4.13
		16	43.08 ± 11.78	54.61 ± 6.79	51.68 ± 11.27	57.28 ± 4.57	**61.02** ± 4.19
Rating DVD	3	4	32.22 ± 08.72	45.94 ± 7.48	41.23 ± 10.98	47.73 ± 6.20	**49.76** ± 9.80
		8	36.35 ± 12.50	46.23 ± 6.03	45.24 ± 9.76	47.11 ± 4.00	**53.28** ± 4.66
		16	42.79 ± 10.18	49.23 ± 6.68	45.19 ± 11.56	48.39 ± 3.74	**53.52** ± 4.77
Rating Electronics	3	4	39.27 ± 10.15	39.89 ± 5.83	41.20 ± 10.69	37.40 ± 3.72	**51.71** ± 7.20
		8	28.74 ± 08.22	46.53 ± 5.44	45.41 ± 09.49	43.64 ± 7.31	**54.78** ± 6.48
		16	45.48 ± 06.13	48.71 ± 6.16	47.29 ± 10.55	44.83 ± 5.96	**58.69** ± 2.41
Rating Kitchen	3	4	34.76 ± 11.20	40.41 ± 5.33	36.77 ± 10.62	44.72 ± 9.13	**50.21** ± 09.63
		8	34.49 ± 08.72	48.35 ± 7.87	47.98 ± 09.73	46.03 ± 8.57	**53.72** ± 10.31
		16	47.94 ± 08.28	52.94 ± 7.14	53.79 ± 09.47	49.85 ± 9.31	**57.00** ± 08.69
Overall Average		4	38.13	40.13	40.10	36.29	**45.99**
		8	36.99	45.89	44.25	39.15	**50.86**
		16	48.55	49.93	49.07	39.85	**55.50**

Table 3.1. Few-shot generalization performance across tasks not seen during training. k is the number of examples per label for fine-tuning and N is the number of classes for the task. On average, LEOPARD is significantly better than other models for few-shot transfer to new tasks.

ent text domains. Note that LEOPARD uses the same training tasks as MT-BERT but can adapt to new tasks with fewer examples, and improvements are highest with

	k	BERT$_{\text{base}}$	MT-BERT$_{\text{softmax}}$	MT-BERT	MT-BERT$_{\text{reuse}}$	Proto-BERT	LEOPARD
Natural Language Inference							
Scitail	4	58.53 ± 09.74	74.35 ± 5.86	63.97 ± 14.36	**76.65** ± 2.45	76.27 ± 4.26	69.50 ± 9.56
	8	57.93 ± 10.70	**79.11** ± 3.11	68.24 ± 10.33	76.86 ± 2.09	78.27 ± 0.98	75.00 ± 2.42
	16	65.66 ± 06.82	**79.60** ± 2.31	75.35 ± 04.80	79.53 ± 2.17	78.59 ± 0.48	77.03 ± 1.82
Amazon Review Sentiment Classification							
Books	4	54.81 ± 3.75	68.69 ± 5.21	64.93 ± 8.65	74.79 ± 6.91	73.15 ± 5.85	**82.54** ± 1.33
	8	53.54 ± 5.17	74.86 ± 2.17	67.38 ± 9.78	78.21 ± 3.49	75.46 ± 6.87	**83.03** ± 1.28
	16	65.56 ± 4.12	74.88 ± 4.34	69.65 ± 8.94	78.87 ± 3.32	77.26 ± 3.27	**83.33** ± 0.79
Kitchen	4	56.93 ± 7.10	63.07 ± 7.80	60.53 ± 9.25	75.40 ± 6.27	62.71 ± 9.53	**78.35** ± 18.36
	8	57.13 ± 6.60	68.38 ± 4.47	69.66 ± 8.05	75.13 ± 7.22	70.19 ± 6.42	**84.88** ± 01.12
	16	68.88 ± 3.39	75.17 ± 4.57	77.37 ± 6.74	80.88 ± 1.60	71.83 ± 5.94	**85.27** ± 01.31
DVD	4	54.98 ± 3.96	63.68 ± 5.03	66.36 ± 7.46	71.74 ± 8.54	74.38 ± 2.44	**80.32** ± 1.02
	8	55.63 ± 4.34	67.54 ± 4.06	68.37 ± 6.51	75.36 ± 4.86	75.19 ± 2.56	**80.85** ± 1.23
	16	58.69 ± 6.08	70.21 ± 1.94	70.29 ± 7.40	76.20 ± 2.90	75.26 ± 1.07	**81.25** ± 1.41
Electronics	4	58.77 ± 6.10	61.63 ± 7.30	64.13 ± 10.34	72.82 ± 6.34	65.68 ± 6.80	**74.88** ± 16.59
	8	59.00 ± 5.78	66.29 ± 5.36	64.21 ± 10.49	75.07 ± 3.40	68.54 ± 5.61	**81.29** ± 1.65
	16	67.32 ± 4.18	69.61 ± 3.54	71.12 ± 7.29	75.40 ± 2.43	67.84 ± 7.23	**81.86** ± 1.56

Table 3.2. Domain transfer evaluation (accuracy) on NLI and Sentiment classification datasets.

only 4 examples. Performance of prototypical networks is worse than most other fine-tuning methods on new training tasks. We hypothesize that this is because prototypical networks do not generate good class prototypes for new tasks and adaptation of class prototypes is important for improving performance. We also see that improved feature learning in MT-BERT with additional training tasks serves as a better initialization point for held-out tasks than BERT, and only tuning the softmax layer of this model is slightly better than tuning all parameters. Interestingly, on some tasks like Disaster classification, we observe BERT to perform better than other models, indicating negative transfer from the training tasks.

3.3.3.2 Few-Shot Domain Transfer

We now evaluate performance on new domains of tasks seen at training time. For this, we consider two tasks of Sentiment Classification and NLI. For sentiment classification we use 4 domains of Amazon reviews Blitzer et al. (2007) and for NLI we use a scientific entailment dataset (SciTail) Khot et al. (2018). We introduce another relevant baseline here, MT-BERT$_{\text{reuse}}$, which reuses the trained softmax parameters of a related train task. Results are summarized in Table 3.2, we show two domains of sentiment classification and more results are in Appendix A.2. Note that the related train task, SST, only contains phrase-level sentiments and the models weren't trained to predict sentence-level sentiment, while the target tasks require sentence-level sentiment. We observe that LEOPARD performs better than the baselines on all domains of sentiment classification, while on Scitail MT-BERT models perform better, potentially because training consisted of many related NLI datasets. Note that prototypical networks is a competitive baseline here and its performance is better for these tasks in comparison to those in Table 3.1 as it has learned to generate prototypes for a similar task during training.

3.3.4 Ablation Study

For ablations we use the dev-set of 3 tasks: CoNLL-2003 entity typing, Amazon reviews DVD domain sentiment classification and SciTail NLI.

Importance of softmax parameters: Since the softmax generation is an important component of LEOPARD, we study how it affects performance. We remove the softmax generator and instead add a softmax weight and bias with zero initialization for each task. The model is trained in a similar way as LEOPARD. This method, termed LEOPARD-ZERO, is a naive application of MAML to this problem. Table 3.3 shows that this performs worse on new tasks, highlighting the importance of softmax generator.

Parameter efficiency: We consider three variants of LEOPARD with parameter efficient training discussed in Sec 3.2.3. Denote LEOPARD$_\nu$ as the model which does not adapt layers 0 to ν (including word embeddings) in the inner loop of meta-

k	Model	Entity Typing	Sentiment Classification	NLI
16	LEOPARD $_{10}$	37.62 $\pm$ 7.37	58.10 $\pm$ 5.40	78.53 $\pm$ 1.55
	LEOPARD $_{5}$	62.49 $\pm$ 4.23	71.50 $\pm$ 5.93	73.27 $\pm$ 2.63
	LEOPARD	69.00 $\pm$ 4.76	76.65 $\pm$ 2.47	76.10 $\pm$ 2.21
	LEOPARD-ZERO	44.79 $\pm$ 9.34	74.45 $\pm$ 3.34	74.36 $\pm$ 6.67

Table 3.3. Ablations: LEOPARD$_\nu$ does not adapt layers $0-\nu$ (inclusive) in the inner loop (and fine-tuning), while LEOPARD adapts all parameters. Note that the outer loop still optimizes all parameters. For new tasks (like entity typing) adapting all parameters is better while for tasks seen at training time (like NLI) adapting fewer parameters is better. LEOPARD-ZERO is a model trained without the softmax-generator and a zero initialized softmax classifier, which shows the importance of softmax generator in LEOPARD.

training. Note that even for $\nu \neq 0$, the parameters are still optimized in the outer loop. Table 3.3 shows the results. Interestingly, for all tasks (except NLI) we find that adapting all parameters is better. This is potentially because the per-layer learning rate in LEOPARD also adjust the adaptation rates for each layer. On SciTail (NLI) we observe the opposite behaviour, suggesting that adapting fewer parameters is better for small k, potentially because training consisted of multiple NLI datasets.

Importance of training tasks: We study how target-task performance of MT-BERT and LEOPARD is dependent on tasks used for training. For this experiment, we held out each training task one by one and trained both models. The trained models are then evaluated for their performance on the target tasks (using the development set), following the same protocol as before. Fig. 3.2 shows a visualization of the relative change in performance when each training task is held out. We see that LEOPARD's performance is more consistent with respect to variation in training tasks, owing to the meta-training procedure that finds an initial point that performs equally well across tasks. Removing a task often leads to decrease in performance for LEOPARD as it decreases the number of meta-training tasks and leads to over-fitting to the training task-distribution. In contrast, MT-BERT's performance on target tasks varies greatly depending on the held-in training tasks.

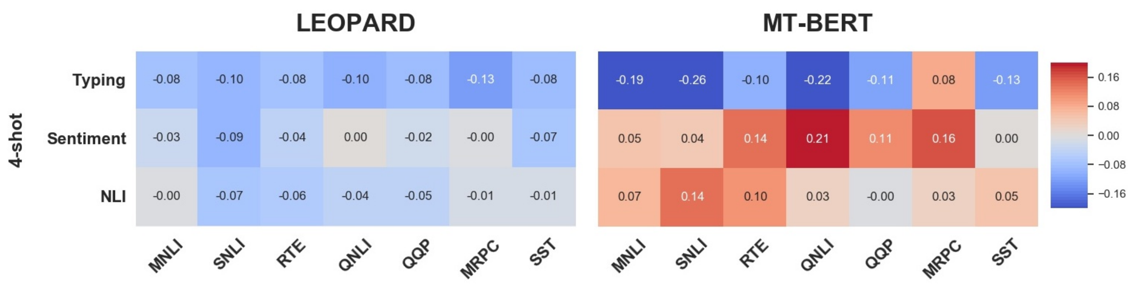


Figure 3.2. Analyzing target task performance as a function of training tasks (best viewed in color). Each column represents one held-out training task (name on x-axis) and each row corresponds to one target task (name on y-axis). Each cell is the relative change in performance on the target task when the corresponding training task is held-out, compared to training on all the train tasks. Dark blue indicates large drop, dark red indicates large increase and grey indicates close to no change in performance. In general, LEOPARD's performance is more consistent compared to MT-BERT indicating that meta-training learns more generalized initial parameters compared to multi-task training.

3.4 Related Work

Meta-Learning approaches can be broadly classified as: optimization-based (Finn et al., 2017; Al-Shedivat et al., 2018b; Nichol and Schulman, 2018; Rusu et al., 2019), model-based (Santoro et al., 2016; Ravi and Larochelle, 2017; Munkhdalai and Yu, 2017), and metric-learning based (Vinyals et al., 2016; Snell et al., 2017; Sung et al., 2018). Recently, related to this work, it has been shown that learning task-dependent model parameters improves few-shot learning (Rusu et al., 2019; Zintgraf et al., 2019). While many existing methods train and evaluate on simulated datasets with limited diversity, there is recent interest for more realistic meta-learning applications (Triantafillou et al., 2019) and our work advances this by training and evaluating on diverse and real NLP classification tasks.

Meta-learning applications in NLP have yielded improvements on specific tasks. Gu et al. (2018) used MAML to simulate low resource machine translation, Chen et al. (2018) learn HyperLSTM (Ha et al., 2016) model in a multi-task setting across various sentiment classification domains, and other recent approaches (Guo et al., 2018; Yu et al., 2018; Han et al., 2018; Obamuyide and Vlachos, 2019; Geng et al., 2019; Mi et al., 2019; Bao et al., 2020) meta-train for a *specific* classification task, such as relation classification, and do not generalize beyond the training task. Dou et al. (2019) train on a subset of GLUE tasks to generalize to other GLUE tasks and their approach does not consider unseen tasks. Transfer learning is a closely related research area. Self-supervised pre-training has been shown to learn general-purpose model parameters that improve downstream performance with fine-tuning (Peters et al., 2018; Howard and Ruder, 2018; Devlin et al., 2019; Radford et al., 2019; Yang et al., 2019; Raffel et al., 2019). Fine-tuning, however, typically requires large training data (Yogatama et al., 2019). Multi-task learning with BERT has been shown to improve performance for many related tasks (Phang et al., 2018; Wang et al., 2018a; Liu et al., 2019a). We refer the reader to Ruder (2019) for a more thorough discussion of transfer learning and multi-task learning.

3.5 Conclusions

Learning general linguistic intelligence has been a long-term goal of NLP. While humans, with all their prior knowledge, can quickly learn to solve new tasks with very few examples, machine-learned models still struggle to demonstrate such intelligence. To this end, we proposed LEOPARD, a meta-learning approach, and found that it learns more general-purpose parameters that better prime the model to solve completely new tasks with few examples. While we see improvements using meta-learning, performance with few examples still lags behind human-level performance. We consider bridging this gap as a lucrative goal to demonstrate general linguistic intelligence, and meta-learning as a strong contender to achieve this goal.

CHAPTER 4

SELF-SUPERVISED META-LEARNING FOR FEW-SHOT NATURAL LANGUAGE CLASSIFICATION TASKS

Self-supervised pre-training methods provide a useful initial point for parameters that generalize well to new tasks with fine-tuning. However, as seen in the previous chapter, fine-tuning is still data inefficient — when there are few labeled examples, accuracy can be low. Data efficiency can be improved by optimizing pre-training directly for future fine-tuning with few examples; this can be treated as a meta-learning problem. However, standard meta-learning techniques require many training tasks in order to generalize; unfortunately, finding a diverse set of such supervised tasks is usually difficult. This chapter proposes a self-supervised approach to generate a large, rich, meta-learning task distribution from unlabeled text. This is achieved using a cloze-style objective, but creating separate multi-class classification tasks by gathering tokens-to-be blanked from among only a handful of vocabulary terms. This yields as many unique meta-training tasks as the number of subsets of vocabulary terms. We meta-train a transformer model on this distribution of tasks using a recent meta-learning framework. On 17 NLP tasks, we show that this meta-training leads to better few-shot generalization than language-model pre-training followed by finetuning. Furthermore, we show how the self-supervised tasks can be combined with supervised tasks for meta-learning, providing substantial accuracy gains over previous supervised meta-learning.

4.1 Introduction

Self-supervised learning has emerged as an important training paradigm for learning model parameters which are more generalizable and yield better representations for many down-stream tasks. This typically involves learning through labels that come naturally with data, for example words in natural language. Self-supervised tasks typically pose a supervised learning problem that can benefit from lots of naturally available data and enable pre-training of model parameters that act as a useful prior for supervised fine-tuning (Erhan et al., 2010). Masked language modeling (Devlin et al., 2019), and other related approaches (Peters et al., 2018; Howard and Ruder, 2018; Radford et al., 2019), is an example of such a self-supervised task that is behind the success of transformer models like BERT.

While self-supervised pre-training is beneficial, it has been recently noted that it is not data-efficient and typically requires large amounts of fine-tuning data for good performance on a target task (Yogatama et al., 2019; Bansal et al., 2020a). This can be evaluated as a few-shot learning problem, where a model is given only few examples of a new task and is expected to perform well on that task. This chapter focuses on this problem of few-shot learning and develops models which demonstrate better few-shot generalization to new tasks.

Large scale pre-training suffers from a train-test mismatch as the model is not optimized to learn an initial point that yields good performance when fine-tuned with few examples. Moreover, fine-tuning of a pre-trained model typically introduces new random parameters, such as softmax layers, and important hyper-parameters such as learning rate, which are hard to estimate robustly from the few examples. Thus, we propose to remove this train-test mismatch, and treat learning an initial point and hyper-parameters jointly from unlabelled data, which allows data-efficient fine-tuning, as a meta-learning problem.

Meta-learning, or learning to learn (Thrun and Pratt, 2012; Schmidhuber, 1987), treats learning a parameterized algorithm, such as a neural net optimized with SGD, that generalizes to new tasks as a learning problem. This typically assumes access to a distribution over tasks in order to enable learning. Creating tasks which enable meta-learning is one of the main challenges for meta-learning (Bengio et al., 1992; Santoro et al., 2016; Vinyals et al., 2016), and typical supervised meta-learning approaches create task distributions from a fixed task dataset with large number of labels by sub-sampling from the set of labels (Vinyals et al., 2016; Ravi and Larochelle, 2017). While this enables generalization to new labels, it limits generalization to unseen tasks due to over-fitting to the training task distribution (Yin et al., 2020a). Moreover, large supervised datasets with a large label set are not always available for meta-learning, as is often the case in many NLP applications.

To overcome these challenges of supervised meta-learning, we propose a self-supervised approach and create the task-distribution from unlabelled sentences. Taking inspiration from the cloze task (Taylor, 1953), we create separate multi-class classification tasks by gathering tokens-to-be blanked from a subset of vocabulary words, allowing for as many unique meta-training tasks as the number of subsets of words in the language. The proposed approach, which we call Subset Masked Language Modeling Tasks (SMLMT), enables training of meta-learning methods for NLP at a much larger scale than was previously feasible while also ameliorating the risk of over-fitting to the training task distribution. This opens up new possibilities for applications of meta-learning in NLP, such as few-shot learning, continual learning, architecture search and more.

This work focuses on few-shot learning and makes the following contributions: (1) we introduce a self-supervised approach to create tasks for meta-learning in NLP, Subset Masked Language Modeling Tasks (SMLMT), which enables application of meta-learning algorithms for goals like few-shot learning; (2) utilizing SMLMT as the training task distribution, we train a state-of-the-art transformer architecture, BERT (Devlin et al., 2019), using a recent optimization-based meta-learning method which was developed for diverse classification tasks (Bansal et al., 2020a); (3) we show that the self-supervised SMLMT can also be combined with supervised task data to enable better feature learning, while still allowing for better generalization by avoiding meta-overfitting to the supervised tasks through the use of SMLMT; (4) we rigorously evaluate the proposed approach on few-shot generalization to unseen tasks as well as new domains of tasks seen during training and show that the proposed approach demonstrates better generalization than self-supervised pre-training or self-supervised pre-training followed by multi-task training; (5) we also study the effect of number of parameters for few-shot learning and find that while bigger pre-trained or meta-trained models generalize better than smaller models, meta-learning leads to substantial gains even for the smaller models.

4.2 Preliminaries

In supervised meta-learning, we typically assume access to a task distribution $\mathcal{P}(\mathcal{T})$. Practically, this translates to a fixed set of training tasks $\{T_1, \ldots, T_M\}$, which are referred to as meta-training tasks. For supervised classification, each task T_i is an N_i-way classification task. While many meta-learning algorithms assume a fixed N-way classification, we follow the more practical approach presented in the previous chapter and allow for a diverse set of classification tasks with potentially different number of classes.

The goal of a meta-learning algorithm is to utilize the meta-training tasks to learn a learning procedure that generalizes to held-out tasks $T' \sim \mathcal{P}(T)$. Model-agnostic meta-learning (MAML) (Finn et al., 2017) is an example of such a meta-learning algorithm. MAML learns an initial point θ for a classifier $f_\theta : x \to \hat{y}$, that can be optimized via gradient descent on the supervised loss $\mathcal{L}_i$ defined for the task T_i, using its support set $\mathcal{D}^{tr} \sim T_i$:

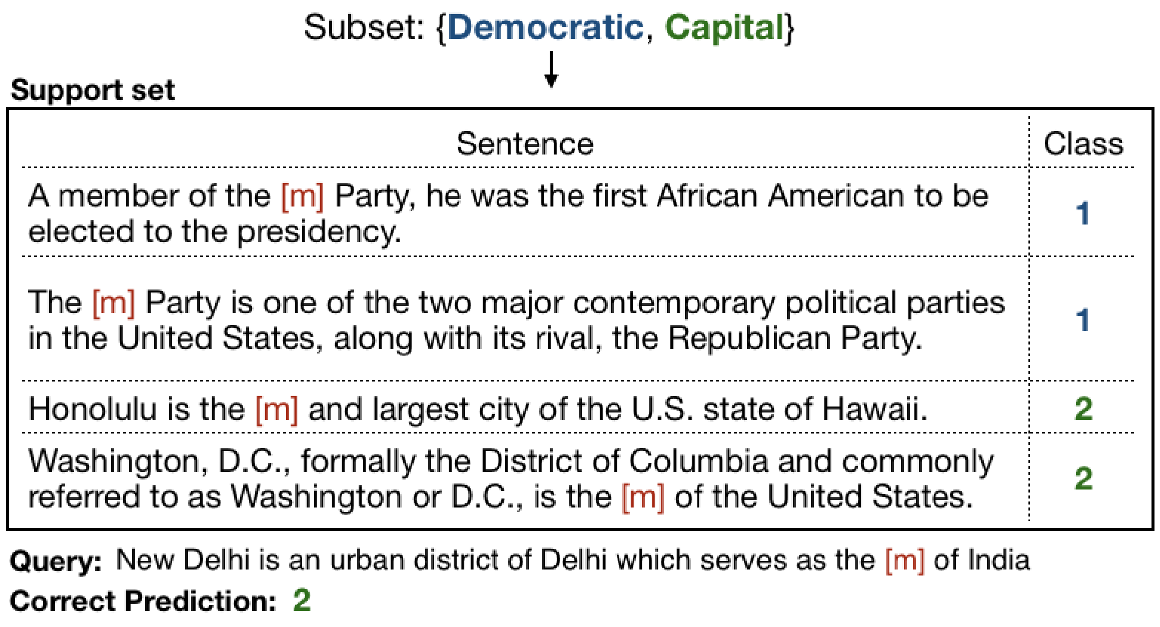
$$\theta'_i \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}_i(\mathcal{D}^{tr}, \theta) \tag{4.1}$$

where α is the learning rate. The optimized point θ' is then evaluated on another validation set for the task, $D^{val} \sim T_i$, using the loss function $\mathcal{L}_i$. This loss across meta-training tasks serves as the training error to optimize the initial point and parameters like learning-rate ($\Theta := \{\theta, \alpha\}$):

$$\Theta \leftarrow \Theta - \beta \; \nabla_\Theta \mathbb{E}_{T_i \sim \mathcal{P}(\mathcal{T})} \left[L_i(\mathcal{D}^{val}, \theta'_i) \right] \tag{4.2}$$

where β is the learning rate for the meta-training process. Training proceeds in an episodic framework (Vinyals et al., 2016), where in each episode a mini-batch of tasks are sampled along with their support and validation sets, and the model parameters are optimized using equation 4.1 and equation 4.2, which are also referred to as inner and outer loop, respectively.

Meta-training Tasks: We summarize how supervised task data-sets are typically leveraged to create meta-training tasks (Vinyals et al., 2016). Assuming access to a supervised task with L classes, an N-way k-shot task is created by first sampling N classes, assuming $N << L$. Then for each of the N sampled classes, $(k+q)$ examples of each class is randomly sampled from the dataset and assigned a unique label in $\{1, \ldots, N\}$. The k examples for each label serve as the support set, while the q examples constitute the validation set described above. Note, that each task consists of a small subset of classes and the class to label (1 to N) assignment is random. This is crucial to avoid learning the sample to label bindings in the parameters of the model, which will make the task-specific training (in equation 4.1) irrelevant and the model will not generalize to new tasks. An example of this approach is Mini-ImageNet (Ravi and Larochelle, 2017), which is a benchmark dataset for learning few-shot image classification.



Subset: {Democratic, Capital}

Support set

Sentence	Class
A member of the [m] Party, he was the first African American to be elected to the presidency.	1
The [m] Party is one of the two major contemporary political parties in the United States, along with its rival, the Republican Party.	1
Honolulu is the [m] and largest city of the U.S. state of Hawaii.	2
Washington, D.C., formally the District of Columbia and commonly referred to as Washington or D.C., is the [m] of the United States.	2

Query: New Delhi is an urban district of Delhi which serves as the [m] of India

Correct Prediction: 2

Figure 4.1. An example of a 2-way 2-shot task in SMLMT. The support set and one query is shown. Any N-way k-shot task can be constructed similarly.

4.3 Self-supervised Tasks for Meta-learning

The existing approach to using a supervised dataset to create tasks, as described above, is fraught with issues, specially for NLP applications. First, note that large classification datasets with large label spaces are not readily available for all NLP tasks, for example sentiment classification which has only few discrete labels. Second, limiting to a fixed supervised dataset to create tasks limits generalization ability and the meta-learned models might generalize to new labels for the task but fail to generalize to new novel tasks (Metz et al., 2019). Lastly, such an approach is also not feasible in all problems where we will like to apply meta-learning (Yin et al., 2020a). For example, consider meta-learning a natural language inference (NLI) model across multiple domains which can generalize to new domains. A powerful model can ignore the training data for each task and directly learn to predict the NLI tag for the examples in each training domain, which will lead to low training error but the model will not generalize to new domains. We overcome these issues by utilizing unlabelled data to create meta-learning tasks. See Fig. 4.1 for an example of generated task.

4.3.1 Subset Masked Language Modeling Tasks (SMLMT)

We are given a text corpus split into sentences X_i and each sentence is a sequence of words from a vocabulary of size V. Now, in Subset Masked Language Modeling Tasks, each task is defined from a *subset* of vocabulary words. To create an N-way classification task, we randomly select N unique vocabulary words: $\{v_1, \ldots, v_N\}$. Then we consider all sentences containing these N words, and for each word randomly sample $r = k + q$ sentences: $\mathbf{x}_{v_i} = \{X_1, \ldots, X_r | v_i \in X_i\}$. Now, we mask the corresponding chosen word from the sentences in each of these N sets, so $\mathbf{x}'_{v_i} = \{\text{Mask}(X_1, v_i), \ldots, \text{Mask}(X_r, v_i)\}$ where $\text{Mask}(X, v)$ replaces all occurrences of v in X with the mask token $[m]$. The set $\{\mathbf{x}'_{v_1}, \ldots, \mathbf{x}'_{v_N}\}$ is then a well-defined N-partition of $N \times r$ sentences, that serves as input examples for the N-way classification task. We forget the original word corresponding to the masked tokens in these sets and assign labels in $\{1, \ldots, N\}$ to the N sets. This gives an instance of an SMLMT classification task: $T = \{(x_{ij}, i) | i \in \{1, .., N\}, x_{ij} \in \mathbf{x}'_{v_i}\}$. This can be split into support and validation for meta-training.

In an SMLMT instance, each input sentence consists of exactly one word that is masked throughout the sentence and its label corresponds to that word. This requires a similar reasoning ability as cloze tasks (Taylor, 1953). Moreover, crucially, the SMLMT task creation ensures that a model cannot memorize the input-label mapping as the target masked word is hidden and the label assignment is randomized, requiring the model to infer the labels from the support set. Note that the SMLMT tasks are also closely related to masked language modeling (MLM) (Devlin et al., 2019). While MLM is a word-level classification task, SMLMT is a sentence-level classification task. Each unique subset of words from the vocabulary defines a unique task in SMLMT. This allows for as many unique tasks as the *number of subsets of words in the vocabulary*, enabling large-scale meta-learning from unsupervised data.

Hybrid SMLMT. Tasks from SMLMT can also be combined with supervised tasks to encourage better feature learning (Caruana, 1997) and increase diversity in tasks for meta-learning. We use a sampling ratio $\lambda \in (0, 1)$ and in each episode select an SMLMT task with probability λ or a supervised task with probability $(1 - \lambda)$. The use of SMLMT jointly with supervised tasks ameliorates meta-overfitting, as tasks in SMLMT cannot be solved without using the task support data. λ is a hyper-parameter. In our experiments, we found $\lambda = 0.5$ to work well.

4.4 Meta-learning Model

We now discuss the meta-learning model for learning new NLP tasks. Full meta-training algorithm can be found in the Appendix.

Text encoder. The input to the model is natural language sentences. This is encoded using a transformer (Vaswani et al., 2017) text encoder. We follow the BERT (Devlin et al., 2019) model and use the same underlying neural architecture for the transformer as their base model. Given an input sentence, the transformer model yields contextualized token representations for each token in the input after multiple layers of self-attention. Following BERT, we add a special CLS token to the start of the input that is used as a sentence representation for classification tasks. Given an input sentence X, let $f_\pi(X)$ be the CLS representation of the final layer of the transformer with parameters π.

Meta-learning across diverse classes. Our motivation is to meta-learn an initial point that can generalize to novel NLP tasks, thus we consider methods that apply to diverse number of classes. Note that many meta-learning models only apply to a fixed number of classes (Finn et al., 2017) and require training different models for different number of classes. We follow the approach of Bansal et al. (2020a) that learns to generate softmax classification parameters conditioned on a task support set to enable training meta-learning models that can adapt to tasks with diverse classes. This combines benefits of metric-based methods (Vinyals et al., 2016; Snell et al., 2017) and optimization-based methods for meta-learning. The key idea is to train a deep set encoder $g_\psi(\cdot)$, with parameters ψ, which takes as input the set of examples of a class n and generates a $(d+1)$ dimensional embedding that serves as the linear weight and bias for class n in the softmax classification layer. Let $\{X_{1n}, \dots, X_{kn}\}$ be the k examples for class n in the support set of a task t:

$$w_t^n, b_t^n = g_\psi(\{f_\pi(X_{1n}), \dots, f_\pi(X_{kn})\}) \tag{4.3}$$

$$p(y|X) = softmax\,\{\mathbf{W}_t\; h_\phi(f_\pi(X)) + \mathbf{b}_t\} \tag{4.4}$$

where $\mathbf{W}_t = [w_t^1; \dots; w_t^N] \in \mathcal{R}^{N \times d}$, $\mathbf{b}_t = [b_t^1; \dots; b_t^N] \in \mathcal{R}^d$ are the concatenation of the per-class vectors in equation 4.3, and h_ϕ is a MLP with parameters ϕ and output dimension d.

Using the above model to generate predictions, the parameters are meta-trained using the MAML algorithm (Finn et al., 2017). Concretely, set $\theta := \{\pi, \phi, \mathbf{W}_t, \mathbf{b}_t\}$ for the task-specific inner loop gradient updates in equation 4.1 and set $\Theta := \{\pi, \psi, \alpha\}$ for the outer-loop updates in equation 4.2. Note that we do multiple steps of gradient descent in the inner loop. Bansal et al. (2020a) performed extensive ablations over parameter-efficient versions of the model and found that adapting all parameters with learned per-layer learning rates performs best for new tasks. We follow this approach.

Fast adaptation. Flennerhag et al. (2019) proposed an approach which mitigates slow adaption often observed in MAML by learning to warp the task loss surface to enable rapid descent to the loss minima. This is done by interleaving a neural network's layers with non-linear layers, called warp layers, which are not adapted for each task but are still optimized across tasks in the outer-loop updates in equation 4.2. Since introducing additional layers will make computation more expensive, we use existing transformer layers as warp layers. We designate the feed-forward layers in between self-attention layers of BERT, which project from dimension 768 to 3072 to 768, as warp-layers. Note that these parameters also constitute a large fraction of total parameters ($\sim$ 51%). Thus in addition to the benefit from warping, not adapting these layers per task means significantly faster training and smaller number of per-task parameters during fine-tuning. The warp layers are still updated in the outer loop during meta-training.

4.5 Related Work

Language model pre-training has recently emerged as a prominent approach to learning general purpose representations (Howard and Ruder, 2018; Peters et al., 2018; Devlin et al., 2019; Radford et al., 2019; Yang et al., 2019; Raffel et al., 2019). Refer to (Weng, 2019) for a review of self-supervised learning. Pre-training is usually a two step process and fine-tuning introduces random parameters making it inefficient when target tasks have few examples (Bansal et al., 2020a). Multi-task learning of pre-trained models has shown improved results on many tasks (Phang et al., 2018; Liu et al., 2019a). More recently, and parallel to this work, (Brown et al., 2020) show that extremely large language models can act as few-shot learners. They propose a query-based approach where few-shot task data is used as context for the language model. In contrast, we employ a fine-tuning based meta-learning approach that enjoys nice properties like consistency which are important for good out-of-distribution generalization (Finn, 2018). Moreover, we show that self-supervised meta-learning can also improve few-shot performance for smaller models.

Meta-Learning methods can be categorized as: optimization-based (Finn et al., 2017; Li et al., 2017; Nichol and Schulman, 2018; Rusu et al., 2018), model-based (Santoro et al., 2016; Ravi and Larochelle, 2017; Munkhdalai and Yu, 2017), and metric-based (Vinyals et al., 2016; Snell et al., 2017). Refer to Finn (2018) for an exhaustive review. Unsupervised meta-learning has been explored in vision. Hsu et al. (2019) cluster images using pre-trained embeddings to create tasks for meta-learning. Metz et al. (2019) meta-learn a biologically-motivated update rule from unsupervised data in a semi-supervised framework. Compared to these, we directly utilize text data to automatically create unsupervised tasks without relying on pre-trained embeddings or access to target tasks.

In NLP, meta-learning approaches have followed the recipe of using supervised task data and learning models for specific tasks. Such approaches (Yu et al., 2018; Gu et al., 2018; Guo et al., 2018; Han et al., 2018; Mi et al., 2019) train to generalize to new labels of a specific task like relation classification and don't generalize to novel tasks. Bansal et al. (2020a) proposed an approach that applies to diverse tasks to enable practical meta-learning models and evaluate on generalization to new tasks. However, they rely on supervised task data from multiple tasks and suffer from meta-overfitting as we show in our empirical results. Holla et al. (2020) studied related approaches for the task of word-sense disambiguation. To the best of our knowledge, the method proposed here is the first self-supervised approach to meta-learning in NLP.

4.6 Experiments

We evaluate the models on few-shot generalization to new tasks and new domains of train tasks. Evaluation consist of a diverse set of NLP classification tasks from multiple domains: entity typing, sentiment classification, natural language inference and other text classification tasks. Our results[1] show that self-supervised meta-learning using SMLMT improves performance over self-supervised pre-training. Moreover, combining SMLMT with supervised tasks achieves the best generalization, improving over multi-task learning by up to 21%.

4.6.1 Implementation Details

SMLMT: We use the English Wikipedia dump, as of March 2019, to create SMLMT. This is similar to the dataset for pre-training of BERT (Devlin et al., 2019), which ensures that gains are not due to using more or diverse pre-training corpora (Liu et al., 2019b). The corpus is split into sentences and word-tokenized to create SMLMT. We run task creation offline and create about 2 Million SMLMT for meta-training, including a combination of 2, 3 and 4-way tasks. After task creation, the data is word-piece tokenized using the BERT-base cased model vocabulary for input to the models.

Supervised Tasks: Bansal et al. (2020a) demonstrated that better feature learning from supervised tasks helps few-shot learning. Thus, we also evaluate multi-task learning and supervised meta-learning for few-shot generalization. We also use GLUE tasks (Wang et al., 2018b) and SNLI (Bowman et al., 2015) as the supervised tasks. Supervised tasks can be combined with SMLMT for meta-training (see 4.3.1). Note that since these are only a few supervised tasks (8 in this case) with a small label space, it is easy for meta-learning models to overfit to the supervised tasks (Yin et al., 2020a) limiting generalization as we show in experiments.

Models: We evaluate the following models:

(1) BERT: This is transformer model trained with self-supervised learning using MLM as the pre-training task on Wikipedia and BookCorpus. We use the cased BERT base

[1]Code and trained models: `https://github.com/iesl/metanlp`

model (Devlin et al., 2019).

(2) MT-BERT: This is a multi-task learning model trained on the supervised tasks. We follow Bansal et al. (2020a) in training this model.

(3) MT-BERT$_{\text{softmax}}$: This is the same model above where only the softmax layer is fine-tuned on downstream tasks.

(4) LEOPARD: This is the meta-learning model proposed in Bansal et al. (2020a) which is trained on only the supervised tasks.

(5) SMLMT: This is the meta-learning model (in 4.4) which is trained on the self-supervised SMLMT.

(6) Hybrid-SMLMT: This is the meta-learning model (in 4.4) trained on a combination of SMLMT and supervised tasks.

Note that all models share the same transformer architecture making the contribution from each component discernible. Moreover, SMLMT and Hybrid-SMLMT models use similar meta-learning algorithm as LEOPARD, so any improvements are due to the self-supervised meta-training. All model are initialized with pre-trained BERT for training.

Evaluation Methodology: We evaluate on few-shot generalization to multiple NLP tasks using the same set of tasks[2] considered in Bansal et al. (2020a). Each target task consists of k examples per class, for $k \in \{4, 8, 16, 32\}$, and different tasks can have different number of classes. Since few-shot performance is sensitive to the few examples used in fine-tuning, each model is fine-tuned on 10 such k-shot support sets for a task, for each k, and the average performance with standard deviation is reported. Models are trained using their training procedures, without access to the target tasks, and are then fine-tuned for each of the k-shot task. Results for MT-

[2]Data: `https://github.com/iesl/leopard`

BERT and LEOPARD are taken from Bansal et al. (2020a).

Hyper-parameters: We follow the approach of Bansal et al. (2020a) and use validation tasks for estimating hyper-parameters during fine-tuning for all baseline models. Note the meta-learning approach learn the learning rates during training and only require the number of epochs of fine-tuning to be estimated from the validation tasks. Detailed hyper-parameters are in Appendix.

4.6.2 Results

4.6.2.1 Few-shot generalization to new tasks

We first evaluate performance on novel tasks not seen during training. The task datasets considered are: (1) entity typing: CoNLL-2003 (Sang and De Meulder, 2003), MIT-Restaurant (Liu et al., 2013); (2) rating classification (Bansal et al., 2020a): 4 domains of classification tasks based on ratings from the Amazon Reviews dataset (Blitzer et al., 2007); (3) text classification: multiple social-media datasets from figure-eight[3].

Results are presented in Table 4.1. Results on 2 domains of Rating are in Supplementary due to space limitation. First, comparing models which don't use any supervised data, we see that on average across the 12 tasks, the meta-trained SMLMT performs better than BERT specially for small $k \in \{4, 8, 16\}$. Interestingly, the SMLMT model which doesn't use any supervised data, also outperforms even MT-BERT models which use supervised data for multi-task training. Next, comparing among all the models, we see that the Hybrid-SMLMT model performs best on average across tasks. For instance, on average 4-shot performance across tasks, Hybrid-SMLMT provides a relative gain in accuracy of 21.4% over the best performing MT-BERT baseline. Compared to LEOPARD, the Hybrid-SMLMT yields consistent improvements for all

[3]https://www.figure-eight.com/data-for-everyone/

Task	N	k	BERT	**SMLMT**	MT-BERT$_{\text{softmax}}$	MT-BERT	LEOPARD	**Hybrid-SMLMT**
CoNLL	4	4	50.4 ± 8.6	46.8 ± 4.8	52.3 ± 4.1	55.6 ± 5.0	54.2 ± 6.3	**57.6** ± 7.1
		8	50.1 ± 11.3	61.7 ± 3.1	65.3 ± 7.1	58.3 ± 3.8	67.4 ± 4.3	**70.2** ± 3.0
		16	74.5 ± 3.1	75.8 ± 4.0	71.7 ± 3.0	71.3 ± 3.3	76.4 ± 3.1	**80.6** ± 2.8
		32	83.3 ± 2.1	84.0 ± 1.7	73.1 ± 2.4	79.9 ± 2.5	83.6 ± 2.4	**85.5** ± 1.7
MITR	8	4	49.4 ± 4.3	46.2 ± 3.90	45.5 ± 5.9	50.5 ± 4.4	49.8 ± 3.3	**52.3** ± 4.3
		8	49.4 ± 7.8	61.1 ± 1.9	58.2 ± 2.6	58.0 ± 3.5	63.0 ± 3.3	**65.2** ± 2.3
		16	69.2 ± 3.7	69.2 ± 2.8	66.1 ± 2.2	66.2 ± 3.5	70.4 ± 2.9	**73.4** ± 1.9
		32	78.8 ± 1.9	78.8 ± 1.3	69.3 ± 1.0	76.4 ± 1.2	78.4 ± 2.0	**80.0** ± 1.5
Airline	3	4	42.8 ± 13.5	42.8 ± 6.1	43.7 ± 7.9	46.3 ± 12.3	55.0 ± 11.8	**56.5** ± 10.7
		8	38.0 ± 17.1	51.5 ± 7.3	52.4 ± 4.0	49.8 ± 10.9	61.4 ± 3.9	**63.0** ± 8.2
		16	58.0 ± 8.2	58.4 ± 3.4	58.8 ± 3.0	57.2 ± 9.9	62.1 ± 5.6	**69.3** ± 2.2
		32	63.7 ± 4.4	65.3 ± 3.8	61.1 ± 3.9	62.5 ± 4.5	67.4 ± 1.2	**71.2** ± 3.3
Disaster	2	4	55.7 ± 10.3	**62.3** ± 9.2	52.9 ± 6.2	50.6 ± 8.3	51.5 ± 4.2	55.3 ± 8.3
		8	56.3 ± 9.6	**67.9** ± 6.8	56.1 ± 7.5	54.9 ± 7.9	56.0 ± 3.6	63.6 ± 6.8
		16	64.5 ± 8.9	**72.9** ± 1.7	65.8 ± 4.2	60.7 ± 6.0	61.3 ± 2.8	70.6 ± 2.2
		32	73.6 ± 1.8	**73.7** ± 2.3	67.1 ± 3.1	72.5 ± 2.3	63.8 ± 2.3	71.8 ± 1.9
Emotion	13	4	9.2 ± 3.2	9.8 ± 1.1	9.4 ± 2.1	9.8 ± 2.1	11.7 ± 2.2	**11.9** ± 1.7
		8	8.2 ± 2.1	11.0 ± 1.0	11.6 ± 2.3	11.2 ± 2.1	12.9 ± 1.6	**13.3** ± 1.0
		16	13.4 ± 2.5	12.1 ± 1.2	13.8 ± 2.0	12.8 ± 2.0	13.4 ± 2.2	**15.2** ± 0.9
		32	16.7 ± 1.2	14.3 ± 1.1	13.8 ± 1.6	**16.9** ± 1.8	14.8 ± 2.0	16.1 ± 1.2
Political Bias	2	4	54.6 ± 5.0	57.7 ± 5.7	54.3 ± 3.9	54.7 ± 3.7	60.5 ± 6.7	**61.2** ± 4.9
		8	56.1 ± 3.8	63.0 ± 4.6	57.4 ± 4.3	54.8 ± 4.2	61.7 ± 6.7	**64.1** ± 4.0
		16	61.0 ± 4.2	**66.3** ± 2.8	59.2 ± 4.2	60.3 ± 3.3	65.1 ± 2.1	66.1 ± 2.0
		32	65.0 ± 2.3	**67.7** ± 2.3	62.7 ± 3.2	65.0 ± 3.0	64.7 ± 3.4	67.3 ± 1.5
Political Audience	2	4	51.9 ± 1.7	**57.9** ± 4.3	51.5 ± 2.7	51.5 ± 3.7	52.6 ± 3.5	57.4 ± 7.2
		8	52.8 ± 2.7	**62.8** ± 4.5	53.5 ± 2.3	54.3 ± 2.9	54.3 ± 4.0	60.0 ± 4.5
		16	58.5 ± 5.0	**64.6** ± 5.2	56.4 ± 2.2	55.1 ± 4.6	57.7 ± 3.5	63.1 ± 4.1
		32	55.3 ± 1.5	**67.7** ± 3.1	53.1 ± 1.3	55.7 ± 1.9	52.5 ± 1.5	65.5 ± 3.8
Political Message	9	4	15.6 ± 2.7	16.2 ± 1.8	13.7 ± 1.1	14.5 ± 1.8	15.7 ± 1.6	**16.7** ± 2.5
		8	13.4 ± 1.7	19.2 ± 2.3	14.3 ± 1.3	15.2 ± 2.8	18.0 ± 2.3	**20.3** ± 1.2
		16	20.7 ± 3.9	21.9 ± 0.6	18.1 ± 1.5	19.2 ± 2.2	18.1 ± 2.4	**22.9** ± 1.8
		32	**24.6** ± 1.8	23.9 ± 1.7	18.7 ± 1.5	21.6 ± 1.8	19.9 ± 1.9	23.8 ± 0.5
Rating Books	3	4	39.4 ± 7.2	35.0 ± 3.9	44.8 ± 9.0	39.0 ± 13.3	54.9 ± 6.2	**57.8** ± 8.3
		8	39.5 ± 10.0	37.2 ± 4.2	51.1 ± 6.8	46.8 ± 14.1	**59.2** ± 4.1	56.9 ± 5.6
		16	43.1 ± 11.8	43.6 ± 4.6	54.6 ± 6.8	51.7 ± 11.3	61.0 ± 4.2	**63.3** ± 4.4
		32	52.2 ± 4.0	50.5 ± 3.3	55.0 ± 6.1	55.0 ± 4.8	64.1 ± 2.0	**64.5** ± 3.1
Rating DVD	3	4	32.2 ± 8.7	38.3 ± 3.6	45.9 ± 7.5	41.2 ± 11.0	49.8 ± 9.8	**52.1** ± 11.0
		8	36.4 ± 12.5	37.9 ± 3.6	46.2 ± 6.0	45.2 ± 9.8	**53.3** ± 4.7	53.0 ± 7.8
		16	42.8 ± 10.2	41.9 ± 4.3	49.2 ± 6.7	45.2 ± 11.6	53.5 ± 4.8	**56.7** ± 4.3
		32	48.6 ± 3.2	46.4 ± 4.9	51.2 ± 4.3	52.8 ± 3.4	55.5 ± 4.5	**57.9** ± 3.9
Rating Electronics	3	4	39.3 ± 10.2	37.7 ± 4.8	39.9 ± 5.8	41.2 ± 10.7	51.7 ± 7.2	**53.7** ± 10.2
		8	28.7 ± 8.2	40.0 ± 4.0	46.5 ± 5.4	45.4 ± 9.5	54.8 ± 6.5	**56.6** ± 3.0
		16	45.5 ± 6.1	45.9 ± 4.7	48.7 ± 6.2	47.3 ± 10.6	**58.7** ± 2.4	**58.7** ± 3.7
		32	51.0 ± 5.9	50.9 ± 3.4	52.6 ± 2.5	53.5 ± 3.9	58.5 ± 5.1	**61.4** ± 3.9
Rating Kitchen	3	4	34.8 ± 11.2	40.8 ± 7.3	40.4 ± 5.3	36.8 ± 10.6	50.2 ± 9.6	**52.1** ± 10.2
		8	34.5 ± 8.7	43.0 ± 5.2	48.4 ± 7.9	48.0 ± 9.7	53.7 ± 10.3	**58.1** ± 7.3
		16	47.9 ± 8.3	46.8 ± 3.9	52.9 ± 7.1	53.8 ± 9.5	57.0 ± 8.7	**61.0** ± 5.5
		32	50.8 ± 4.5	51.7 ± 4.6	54.3 ± 6.4	53.2 ± 5.1	61.1 ± 4.8	**64.7** ± 2.4
Overall Average		4	38.13	40.95	40.13	40.10	45.99	**48.71**
		8	36.99	46.37	45.89	44.25	50.86	**53.70**
		16	48.55	51.61	49.93	49.07	55.50	**58.41**
		32	55.30	56.23	52.65	55.42	57.02	**60.81**

Table 4.1. k-shot accuracy on novel tasks not seen in training. Models on left of separator don't use supervised data.

$k \in \{4, 8, 16, 32\}$ and demonstrates steady improvement in performance with increasing data (k). We note that on some tasks, such as Disaster, SMLMT is better than Hybrid-SMLMT. We suspect negative transfer from multi-task training on these tasks as also evidenced by the drop in performance of MT-BERT. These results show that SMLMT meta-training learns a better initial point that enables few-shot generalization.

4.6.2.2 Few-shot domain transfer

Task	k	BERT$_{\text{base}}$	**SMLMT**	MT-BERT$_{\text{softmax}}$	MT-BERT	MT-BERT$_{\text{reuse}}$	LEOPARD	**Hybrid-SMLMT**
Scitail	4	58.5 ± 9.7	50.7 ± 4.3	74.3 ± 5.9	64.0 ± 14.4	76.7 ± 2.5	69.5 ± 9.6	**76.8** ± 3.4
	8	57.9 ± 10.7	55.6 ± 2.4	**79.1** ± 3.1	68.2 ± 10.3	76.9 ± 2.1	75.0 ± 2.4	**79.1** ± 1.1
	16	65.7 ± 6.8	56.5 ± 3.8	79.6 ± 2.3	75.3 ± 4.8	79.5 ± 2.2	77.0 ± 1.8	**80.4** ± 1.4
	32	68.8 ± 6.3	62.4 ± 3.2	**82.2** ± 1.1	74.9 ± 3.6	81.8 ± 1.1	79.4 ± 2.0	**82.2** ± 1.3
Amazon Books	4	54.8 ± 3.8	55.7 ± 2.6	68.7 ± 5.2	64.9 ± 8.7	74.8 ± 6.9	82.5 ± 1.3	**84.7** ± 0.4
	8	53.5 ± 5.2	60.2 ± 5.3	74.9 ± 2.2	67.4 ± 9.8	78.2 ± 3.5	83.0 ± 1.3	**84.8** ± 0.5
	16	65.6 ± 4.1	62.9 ± 4.4	74.9 ± 4.3	69.7 ± 8.9	78.9 ± 3.3	83.3 ± 0.8	**85.1** ± 0.7
	32	73.5 ± 3.4	71.5 ± 4.7	77.5 ± 1.1	78.9 ± 1.7	82.2 ± 1.1	83.5 ± 0.7	**85.3** ± 0.4
Amazon Kitchen	4	56.9 ± 7.1	58.6 ± 4.7	63.1 ± 7.8	60.5 ± 9.2	75.4 ± 6.3	78.3 ± 18.4	**80.7** ± 7.1
	8	57.1 ± 6.6	59.8 ± 3.7	68.4 ± 4.5	69.7 ± 8.1	75.1 ± 7.2	**84.9** ± 1.1	84.7 ± 1.8
	16	68.9 ± 3.4	65.2 ± 5.8	75.2 ± 4.6	77.4 ± 6.7	80.9 ± 1.6	85.3 ± 1.3	**85.3** ± 1.1
	32	78.7 ± 3.6	71.7 ± 4.3	76.6 ± 2.0	79.7 ± 4.1	82.2 ± 0.7	85.8 ± 0.7	**86.3** ± 0.7
Amazon DVD	4	55.0 ± 4.0	53.0 ± 2.5	63.7 ± 5.0	66.4 ± 7.5	71.7 ± 8.5	80.3 ± 1.0	**83.3** ± 1.9
	8	55.6 ± 4.3	54.3 ± 4.2	67.5 ± 4.1	68.4 ± 6.5	75.4 ± 4.9	80.8 ± 1.2	**83.9** ± 1.1
	16	58.7 ± 6.1	57.9 ± 2.7	70.2 ± 1.9	70.3 ± 7.4	76.2 ± 2.9	81.2 ± 1.4	**83.7** ± 1.0
	32	66.2 ± 5.4	65.1 ± 4.4	70.2 ± 2.1	73.5 ± 4.4	79.2 ± 1.7	81.5 ± 1.3	**84.2** ± 0.9
Amazon Electronics	4	58.8 ± 6.1	56.4 ± 2.7	61.6 ± 7.3	64.1 ± 10.3	72.8 ± 6.3	74.9 ± 16.6	**81.0** ± 1.8
	8	59.0 ± 5.8	62.1 ± 3.9	66.3 ± 5.4	64.2 ± 10.5	75.1 ± 3.4	81.3 ± 1.6	**82.6** ± 0.8
	16	67.3 ± 4.2	64.6 ± 4.3	69.6 ± 3.5	71.1 ± 7.3	75.4 ± 2.4	**81.9** ± 1.6	81.2 ± 2.4
	32	72.8 ± 4.3	70.1 ± 3.8	73.2 ± 2.1	72.3 ± 3.9	80.0 ± 1.6	82.4 ± 0.8	**83.2** ± 1.1

Table 4.2. k-shot domain transfer accuracy.

The tasks considered here had another domain of a similar task in the GLUE training tasks. Datasets used are (1) 4 domains of Amazon review sentiments (Blitzer et al., 2007), (2) Scitail, a scientific NLI dataset (Khot et al., 2018). Results on 2 domains of Amazon are in Supplementary due to space limitation. A relevant baseline here is MT-BERT$_{\text{reuse}}$ which reuses the softmax layer from the related training task. This is a prominent approach to transfer learning with pre-trained models. Comparing Hybrid-SMLMT with variants of MT-BERT, we see that Hybrid-SMLMT performs comparable or better. Comparing with LEOPARD, we see that Hybrid-SMLMT generalizes better to new domains. LEOPARD performs worse than Hybrid-SMLMT on Scitail even though the supervised tasks are biased towards NLI, with 5 of the 8 tasks being variants of NLI tasks. This is due to meta-overfitting to the training domains in LEOPARD which is prevented through the regularization from SMLMT in Hybrid-SMLMT.

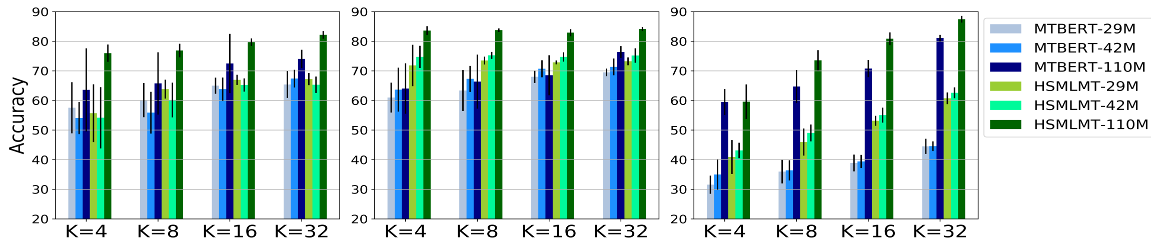


Figure 4.2. k-shot performance with number of parameters on Scitail (left), Amazon DVD (middle), and CoNLL (right). Larger models generalize better and Hybrid-SMLMT provides accuracy gains for all parameter sizes.

4.6.3 Analysis

Meta-overfitting: We study the extent of meta-overfitting in LEOPARD and Hybrid-SMLMT. Since these models learn the adaptation learning-rates, we can study the learning rates trajectory during meta-training. Fig. 4.3 shows the results. We expect the learning rates to converge towards zero if the task-adaptation become irrelevant due to meta-overfitting. LEOPARD shows clear signs of meta-overfitting with much smaller learning rates which converge towards zero for most of the layers. Note that due to this, held-out validation during training is essential to enable any generalization (Bansal et al., 2020a). Hybrid-SMLMT doesn't show this phenomenon for most layers and learning rates converge towards large non-zero values even when we continue training for much longer. This indicates that SMLMT help in ameliorating meta-overfitting.

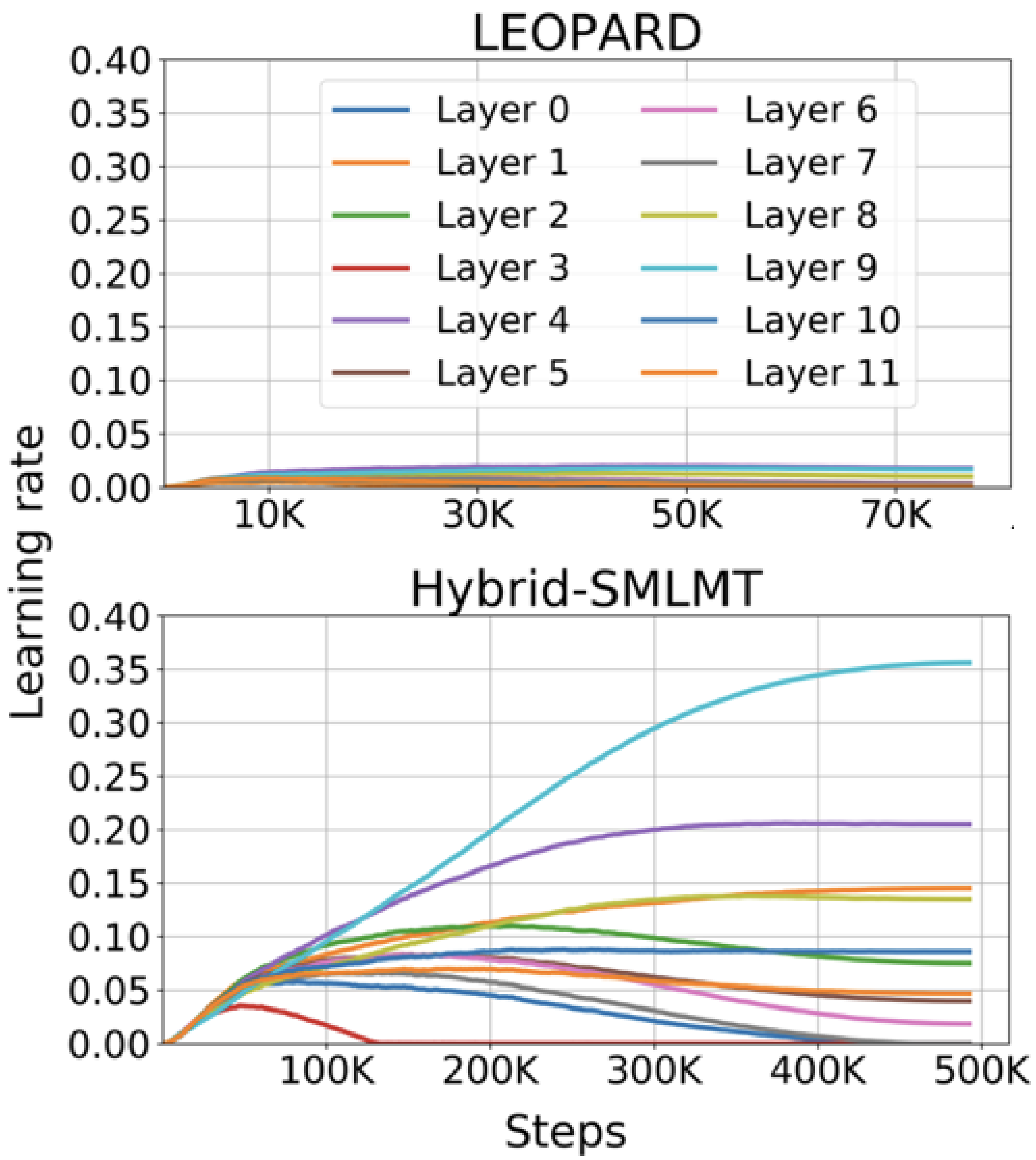
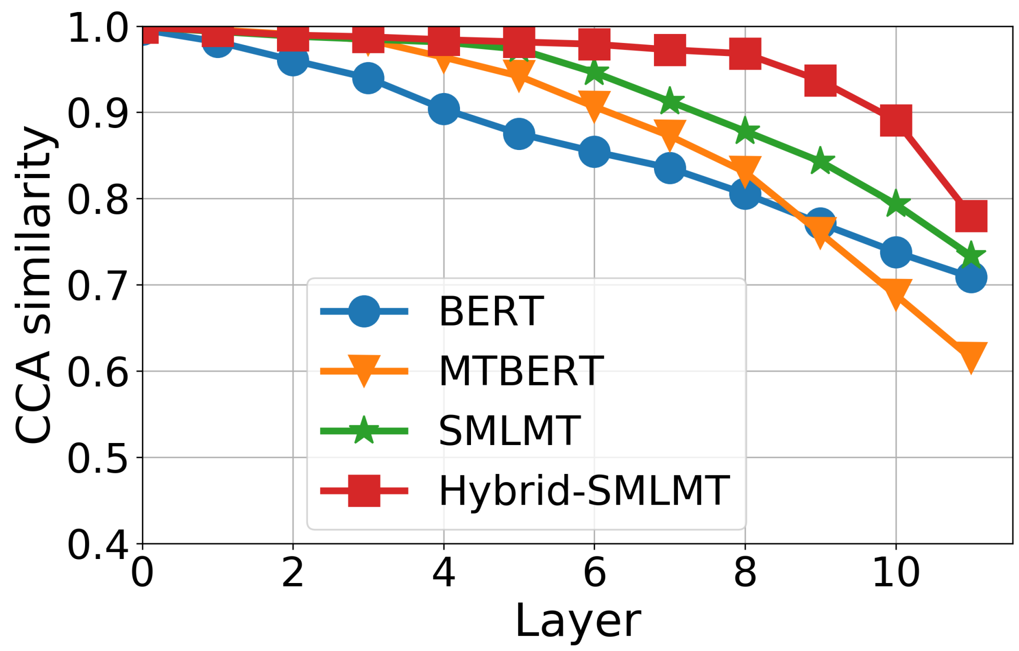


Figure 4.3. Learning rate trajectory during meta-training. LEOPARD learning-rates converge towards 0 for many layers, indicating meta-overfitting.

Effect of the number of parameters: We study how the size of the models affect few-shot performance. Recently, there has been increasing evidence that larger pre-trained models tend to generalize better (Devlin et al., 2019; Radford et al., 2019; Raffel et al., 2019). We explore whether this is true even in the few-shot regime.



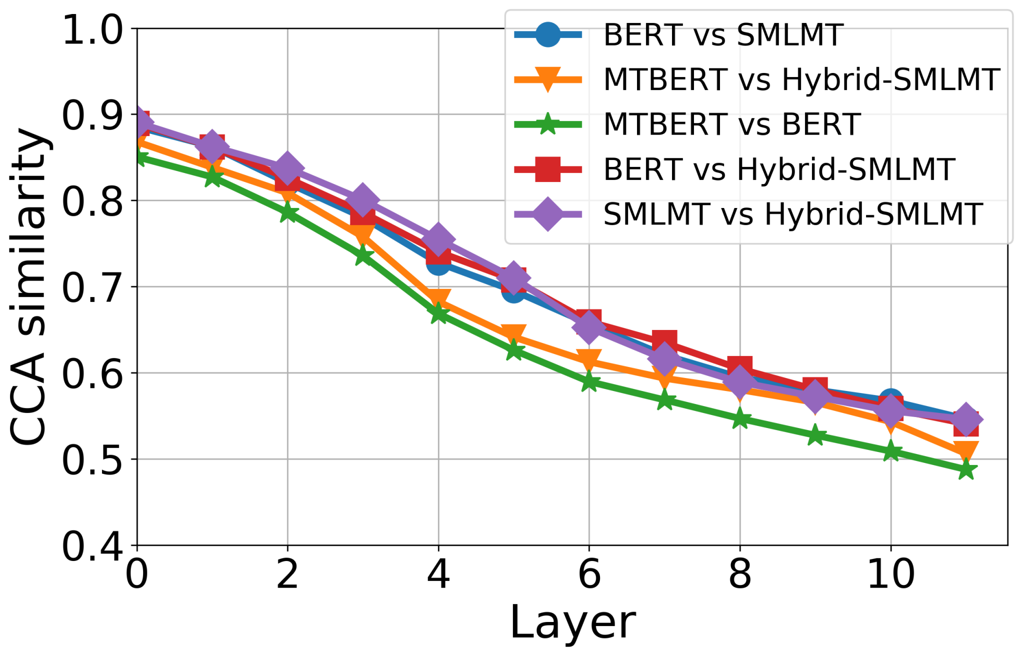


Figure 4.4. CCA similarity for each transformer layer. Top: similarity before and after fine-tuning for the same model. Bottom: similarity between different pairs of models post fine-tuning. More results in Appendix.

For this analysis we use the development data for 3 tasks: Scitail, Amazon DVD sentiment classification, and CoNLL entity typing. We consider the BERT base architecture with 110M parameters, and two smaller versions made available by (Turc et al., 2019) consisting of about 29M and 42M parameters. We train versions of Hybrid-SMLMT as well as MT-BERT corresponding to the smaller models. Results are presented in Fig. 4.2. Interestingly, we see that bigger models perform much better than the smaller models even when the target task had only 4 examples per class. Moreover, we see consistent and large performance gains from the meta-learned Hybrid-SMLMT, even for its smaller model variants. These results indicate that meta-training helps in data-efficient learning even with smaller models, and enables larger models to learn more generalizable representations.

Representation analysis: To probe how the representations in the proposed models are different from the representations in the self-supervised BERT model and multi-task BERT models, we performed CCA analysis on their representations (Raghu et al., 2017). We use the representations on the CoNLL and Scitail tasks for this analysis. Results on CoNLL task are in Fig. 4.4. First, we analyze the representation of the same model before and after fine-tuning on the target task. Interestingly, we see that the Hybrid-SMLMT model is closer to the initial point after task-specific fine-tuning than the BERT and MT-BERT models. Coupled with the better performance of Hybrid-SMLMT (in 4.6.2), this indicates a better initialization point for Hybrid-SMLMT. Note that the representations in lower layers are more similar before and after fine-tuning, and lesser in the top few layers. Next, we look at how representations differ across these models. We see that the models converge to different representations, where the lower layer representations are more similar and they diverge as we move towards the upper layers. In particular, note that this indicates that multi-task learning helps in learning different representations than self-supervised pre-training, and meta-learning model representations are different from the other models.

4.7 Conclusion

We introduced an approach to leverage unlabeled data to create meta-learning tasks for NLP. This enables better representation learning, learning key hyper-parameters like learning rates, demonstrates data-efficient fine-tuning, and ameliorates meta-overfitting when combined with supervised tasks. Through extensive experiments, we evaluated the proposed approach on few-shot generalization to novel tasks and domains and found that leveraging unlabelled data has significant benefits to enabling data-efficient generalization. This opens up the possibility of exploring large-scale meta-learning in NLP for various meta problems, including neural architecture search, continual learning, hyper-parameter learning, and more.

CHAPTER 5

EXPLORING SELF-SUPERVISED TASK DISTRIBUTIONS FOR META-LEARNING

In the previous chapter, we developed an unsupervised task distribution, called Subset Masked Language Modeling Tasks (SMLMT), which generates cloze-style tasks from unlabeled data for meta-learning and showed improvements on a diverse set of few-shot classification tasks. In this chapter, we design multiple distributions of self-supervised tasks by considering important aspects of task diversity, difficulty, type, domain, and curriculum, and investigate how they affect meta-learning performance. Our analysis shows that all these factors meaningfully alter the task distribution, some inducing significant improvements in downstream few-shot accuracy of the meta-learned models. Empirically, results on 20 downstream tasks show significant improvements in few-shot learning – adding up to +4.2% absolute accuracy (on average) to the previous unsupervised meta-learning method, and perform comparably to supervised methods on the FewRel 2.0 benchmark.

5.1 Introduction

In the supervised setting, meta-learning task distribution is often defined by subsampling from the classes in a classification problem over a fixed dataset (Vinyals et al., 2016). This not only limits the applicability of meta-learning to the underlying classification problem, but also requires a diverse set of supervised datasets with a large number of classes to enable learning. Self-supervised meta-learning methods, like those developed in the last chapter, seek to propose tasks from unlabelled data

(Hsu et al., 2019; Bansal et al., 2020b), and have great potential to enable numerous important applications (Hospedales et al., 2020) such as neural architecture search, continual learning, hyper-parameter optimization, learning in low-resource settings, etc. Existing work in meta-learning for NLP, however, defaults to task distributions that tend to be overly simplistic, e.g. using existing supervised datasets (Han et al., 2018; Dou et al., 2019; Bansal et al., 2020a). On the other hand, the SMLMT approach developed in the previous chapter used a uniform selection of words from the vocabulary (Bansal et al., 2020b) for creating tasks which can be sub-optimal. Given the lack of exploration on this critical component, we propose to devise and evaluate diverse task distributions in the context of unsupervised meta-learning for NLP.

Specifically, we explore a diverse set of approaches to create task distributions that are inductive to better meta-training efficacy. We provide empirical evidence that existing definitions of task distributions are prone to producing tasks that might not be challenging enough for the underlying model to learn useful representations, which in turn translates into poor downstream task performance. We therefore propose several new approaches that instead consider important features of the task distribution including task diversity, difficulty, resemblance to the downstream tasks, and the curriculum or the order in which tasks are presented during training. When evaluated on a suite of 20 NLP classification tasks, our best unsupervised meta-learning method leads to an absolute increase of up to +4.2% in average few-shot accuracy over *unsupervised* baseline results; and it even outperforms *supervised* meta-learning methods on FewRel 2.0 benchmark (Gao et al., 2019b) on 5-shot evaluation.

The chapter is organized as follows. We start by revisiting (5.2) the unsupervised task generation approach in SMLMT. Next, we introduce (5.3) new approaches to improve the task distribution. We then analyze (5.5.2) the different unsupervised distributions and how they relate to each other. Finally, we evaluate (5.5.3, 5.5.4) the different unsupervised methods on a wide range of NLP tasks including sentiment classification, entity typing, text classification, sentence-pair classification and relation classification.

5.2 Background

Training meta-learning methods requires a distribution over tasks. Supervised meta-learning often utilizes a fixed task dataset to create a distribution over tasks by sub-sampling from a pool of supervised labels (Vinyals et al., 2016; Finn et al., 2017). This limits learning to a specific type of task which has abundant supervised datasets and doesn't enable learning general purpose meta-learners for diverse tasks. Bansal et al. (2020b) sought to provide an unsupervised approach that proposes tasks from unlabelled data. They proposed Subset Masked Language Modeling Tasks (SMLMT) which propose self-supervised tasks to enable meta-learning and showed that it improves few-shot learning across a diverse set of classification task.

Sampling an N-way task from SMLMT requires first sampling a size-N subset of the vocabulary, which are subsequently mapped to consecutive integer ids and serve as labels for the task. Then to sample examples for each label, sentences containing that word are sampled and the occurrences of the word are masked out. Note that a task in SMLMT is a sentence classification task where each input sentence consists of exactly one word type that is masked throughout the sentence and the label for the sentence is the underlying word type that was masked. This enables sampling combinatorially many classification tasks for meta-learning.

5.3 Diverse Distributions of Self-Supervised Tasks

Sampling tasks in SMLMT depends on sampling of words, which serve as labels, and sampling of sentences containing that word. The original formulation used uniform sampling for both steps. This can lead to several limitations on the quality of the resulting task distribution including task diversity and difficulty. The single-sentence classification tasks also lack cross-sentence reasoning capacities, leading to a severe train-test mismatch for downstream tasks involving sentence pairs. To remedy these problems, we consider alternative distributions that are inductive to more diverse and challenging tasks for meta-training. We also describe an automatic curriculum over tasks that seeks to continuously find challenging tasks for the model during training.

5.3.1 Frequency-based sampling

Word distribution in natural language is characterized by an exponential distribution with a long tail of rare words (Baayen, 2002). Uniform sampling of words in SMLMT puts a disproportionately high weight on the long tail, leading to inefficient use of the training corpora since the low frequency words occur in only a small proportion of the sentences. On the other hand, simple frequency-based sampling can be highly skewed towards a handful of high frequency words. We thus propose to simply sample words in proportion to their log-frequency instead.

5.3.2 Cluster-based sampling

Given two words randomly sampled from a large vocabulary, it is likely to be rather trivial to distinguish their corresponding contexts. This can lead to overly simple tasks in the SMLMT task distribution. To avoid this problem, we consider clustering words based on pre-trained word embeddings and grouping words into semantically-related clusters. Diverse and difficult instances of tasks in SMLMT can then be sampled by selecting all words in a task from either (1) the same cluster (*intra-cluster* sampling), or (2) different clusters (*inter-cluster* sampling). Words co-occurring in the same cluster are semantically or topically related and hence occur in similar contexts, leading to harder to classify sentences as we see in our analysis (Sec 5.5.2). Moreover, choosing different clusters to sample words across tasks provides a natural diversity over topics in the training tasks. On the other hand, picking words from different clusters (*inter-cluster* sampling) can still lead to tasks where the sentences are easy to classify due to easily distinguishable contexts.

Specifically, clustering of pre-trained word embeddings using k-means has been proven effective in generating topical clusters rivaling topic models (Sia et al., 2020). We use the FastText (Joulin et al., 2017) embeddings as word representations. We choose FastText as it is fast, incorporates sub-word information, can generate embeddings for out-of-vocabulary words, and has been found to yield topical clusters (Sia et al., 2020).

Since cluster sizes can be imbalanced, we pick clusters proportional to the number of words in the cluster. Thus, assuming $\{C_1, \ldots, C_m\}$ to be the m clusters of the word vocabulary, we replace the uniform sampling over words in SMLMT as:

$$\begin{aligned} p_i &= \frac{|C_i|}{\sum_{t=1}^{m} |C_t|} \\ i &\sim \text{Cat}(p_1, \ldots, p_m) \\ w|i &\sim \text{Uniform}(\{w|w \in C_i\}) \end{aligned}$$

where $\text{Cat}(p_1, \ldots, p_m)$ is a categorical distribution over m categories with probabilities $\{p_1, \ldots, p_m\}$.

5.3.3 Dynamic curriculum over self-supervised tasks

The methods discussed so far use a static task distribution for learning with tasks sampled i.i.d from this distribution for training. Curriculum learning (Bengio et al., 2009; Graves et al., 2017) instead posits that choosing the order in which instances are presented, with gradually increasing complexity, can enable faster learning and better generalization. We explore whether a curriculum in task sampling is beneficial for meta-learning by proposing a method to sample increasingly difficult tasks during training. To enable this we need a method to propose difficult tasks based on the current state of the model during training.

Since words act as labels in SMLMT, words that are closer in the representational space of the neural model will be more difficult to distinguish, leading to more difficult tasks. On the other hand, nearest-neighbors can be too difficult to induce effective learning for a model. This is related to findings in negative sampling in metric learning literature (Schroff et al., 2015; Suh et al., 2019) where using “too hard” negatives typically hurts performance.

To alleviate this problem, we cluster representations computed from the model and uniformly sample words within the same cluster to create difficult but not impossible tasks (similar to the ”static” clustering approach). Secondly, we adopt an easy-to-hard curriculum by controlling the ratio between the harder tasks from the dynamic distribution $\mathcal{D}_t$ and the easier ones from the static distribution $\mathcal{S}$, consisting of tasks sampled i.i.d from uniform random word sampling or fixed word-clustering. At step t, let λ_t be the probability of sampling a task from $\mathcal{D}_t$ and $1 - \lambda_t$ from $\mathcal{S}$. Then the dynamic curriculum is defined by sampling tasks from the following mixture distribution with λ_t linearly annealed over the training epochs from 0 to 1:

$$T \sim \lambda_t \mathcal{D}_t + (1 - \lambda_t)\mathcal{S}$$

To construct $\mathcal{D}_t$, we consider the following word (i.e. label) representation for clustering, obtained by the average representation under the model of the masked sentences corresponding to a word:

$$\hat{w}_i^{(t)} = \mathbb{E}_{x \sim S(w_i)} \left[f_{\theta_t}(x) \right]$$

where $S(w_i)$ is the set of all sentences containing the word w_i with the word w_i masked out (as defined in SMLMT), $f_{\theta_t}(.)$ is the representation from the neural model for instance x that is fed into the softmax classification layer, and θ_t are the model parameters at step t.

To make the computation of $\hat{w}_i^{(t)}$ tractable, we first approximate the quantity by the expectation over a subset of $S(w_i)$. Moreover, since computing the representations $\{\hat{w}_i^{(t)}\}$ for all vocabulary words and clustering at every step t of the training will be computationally infeasible, we consider doing this after m steps of meta-training. This also allows the model to train on the current distribution for sometime before abruptly changing the distribution. Finally, while the model is being updated between time step t and $t + m$, we use the model snapshot at t to create the word clusters asynchronously for the model at $t + m$, which allows the task generation to run in parallel to the model training.

5.3.4 Task proposal using sentence clustering

SMLMT uses a data-augmentation strategy to automatically assign labels to unlabelled sentences by consistently masking out the *same* word type in a set of sentences. The masked out word then serves as the label for the sentences. This cloze-style approach to creating tasks was inspired by the success of masked language modeling (Devlin et al., 2019) in learning useful representations. While this leads to significant improvements in sentence-level classification on a range of real downstream tasks (Bansal et al., 2020b), it is unclear whether a word masking approach is the most efficient to learning useful sentence representations. To probe this question further, we explore an alternative to SMLMT that directly assigns semantic labels to sentences without any augmentation.

Specifically, we consider pre-trained sentence representations for proposing tasks, which have been proven useful for improving semi-supervised learning (Du et al., 2020a). We use a pre-trained sentence embedding model (Du et al., 2020a; Wenzek et al., 2020) to embed all sentences in a corpus and cluster them. To propose an N-way task, we first randomly sample N cluster-ids and remap them to random consecutive integers $\{1, \ldots, N\}$. Then examples for each label are sampled from the corresponding cluster, creating a classification task for classifying the sentences into their underlying cluster labels. Note that the step of remapping the cluster-ids ensures that the model cannot memorize the sentence to cluster mapping, which would lead to meta over-fitting (Hsu et al., 2019).

5.3.5 Contrastive learning over sentence pairs

SMLMT proposes sentence-level tasks and thus lacks cross-sentence reasoning. This is confirmed by the poor downstream few-shot performance of models trained on SMLMT (see Sec. 5.5.3). Since models trained on SMLMT have never seen pairs of sentences as input, it leads to a train-test mismatch for sentence-pair classification tasks. To remedy this, we introduce a simple but effective contrastive learning task over sentence-pairs that bridges this gap. Contrastive learning has been used to learn effective sentence representations (Logeswaran and Lee, 2018). Next sentence prediction, a sentence-pair task, was used in the training of BERT (Devlin et al., 2019) which was later found to be not effective (Liu et al., 2019b). BERT considered segments instead of full sentences, however the downstream tasks often require reasoning over complete sentences. Thus, we consider classifying whether two sentences come from the same document as opposed to different documents, as a sentence-pair task to enable cross-sentence reasoning. This simple objective was found to be quite effective in our experiments. Note that during meta-training, this can be treated as an additional task in the task distribution. Since the SMLMT task distribution consists of an exponential number of tasks, we sample the sentence-pair task in an episode with a fixed probability α, which is hyper-parameter.

5.4 Related Work

Meta-learning applications in NLP have yielded improvements on specific tasks (Gu et al., 2018; Chen et al., 2018; Guo et al., 2018; Yu et al., 2018; Han et al., 2018; Dou et al., 2019). Unsupervised meta-learning has been explored in computer vision (Hsu et al., 2019; Khodadadeh et al., 2019) and reinforcement learning (Gupta et al., 2018). Hsu et al. (2019) cluster images using pre-trained embeddings to create tasks. Metz et al. (2019) meta-learn an unsupervised update rule in a semi-supervised framework. Bansal et al. (2020b) developed the SMLMT approach to unsupervised meta-learning in NLP. Contemporary work (Murty et al., 2021) explored the use of clustering, though focused only on natural language inference tasks. Curriculum learning (Bengio et al., 2009) in the context of meta-learning has been unexplored in NLP, prior to this work. Jabri et al. (2019) found unsupervised curriculum to be beneficial for meta-reinforcement learning. We refer to Hospedales et al. (2020) for a comprehensive review of meta-learning.

Self-supervised learning has emerged as an efficient approach to representation learning in NLP (Howard and Ruder, 2018; Peters et al., 2018; Devlin et al., 2019; Radford et al., 2019; Yang et al., 2019). Multi-task learning of pre-trained models has shown improved results on many tasks (Phang et al., 2018; Liu et al., 2019a), including few-shot setting. Yin et al. (2020b) leveraged entailment tasks for few-shot learning. Du et al. (2020a) developed self-training methods for semi-supervised few-shot learning. Recently, extremely large language models have been shown to have few-shot capacities (Brown et al., 2020), while Schick and Schütze (2020) demonstrated few-shot capacities for small models in the semi-supervised setting. Meanwhile, Bansal et al. (2020a,b) showed meta-learning to be effective at improving few-shot performance in multi-task and unsupervised settings, as well as improving performance for small models.

5.5 Experiments

We evaluate various self-supervised task distributions for their utility in meta-learning for few-shot classification. We first describe the experimental setting, then we perform evaluations to understand how the different self-supervised tasks relate to each other, and finally show performance on a large set of 20 real classification datasets. These datasets cover a wide range of tasks: sentiment classification, entity typing, text classification, sentence pair classification and relation classification. Our proposed approach shows significant improvements over previous few-shot classification results (Bansal et al., 2020b; Gao et al., 2019b).

5.5.1 Experimental Setup

We consider the challenging few-shot setting where models are trained on unlabelled corpora and then evaluated on target tasks with only k examples per label ($k \leq 32$) to allow fine-tuning of the models on the target task. Since our focus is on unsupervised meta-learning, we closely follow the experimental setup of Bansal et al. (2020b).

Meta-learning Model: We use the same model as in Bansal et al. (2020b) for our results to be comparable[1]. The model is a BERT transformer encoder coupled with a parameter generator, a 2-layer MLP, that generates the initial point for classification layer for a task conditioned on the support examples. The model is meta-trained using the MAML algorithm (Finn et al., 2017), with learned per-layer learning rates, on the self-supervised task distributions. All model hyper-parameters are kept the same so that any change in performance can be attributed to differences in the task distribution. See Supplementary for all hyper-parameters.

Methods Evaluated: We consider all the different approaches to self-supervised task distributions described in Sec 5.3 and the baseline approach of SMLMT: (1) *Uniform*: this is the SMLMT approach of Bansal et al. (2020b) which use uniform

[1]Code and datasets available at: `https://github.com/thetb/meta_tasks`

random sampling over word-types; (2) *Frequency*: SMLMT with a sampling proportional to log-frequency (see 5.3.2); (3) *Cluster*: SMLMT where labels are picked from same word cluster (see 5.3.2); (4) *Dynamic*: curriculum-based task sampling with Cluster as the static distribution (see 5.3.3); (5) *Cluster-ccnet*: same as Cluster but using ccnet (Wenzek et al., 2020) as the corpora, which consists of web crawled data; (6) *SentCluster*: alternative to SMLMT which proposes tasks from subsets of sentence clustering (see 5.3.4); (7) *SentPair*: the sentence-pair tasks (see 5.3.5). All methods, except SentCluster and Cluster-ccnet, have Wikipedia as the text corpora. The sentence embeddings for SentCluster task distribution were obtained from Du et al. (2020a), and consist of embeddings of about 1 billion sentences from ccnet (Wenzek et al., 2020). For this reason, we also report Cluster-ccnet that uses this same set of sentences. We found it beneficial to include 25% *Frequency* tasks in the *Cluster* task distribution and *SentPair* tasks are included in all other task distributions unless otherwise noted. Note that we only consider completely unsupervised meta-learning methods for fair evaluation. However our results improve over Bansal et al. (2020b) which showed improvements over BERT and multi-task BERT baselines. As we utilize the same dataset splits released in their work, our results can be directly compared.

5.5.2 Analyzing task distributions

We start by a quantitative exploration of the various self-supervised task proposals without resorting to full fine-tuning on downstream tasks. Our goal here is to understand properties of these task distributions and how they relate to each other. To do this, we consider models meta-trained on a specific type of task proposal (rows in Table 5.1) and evaluate their performance in the few-shot setting on tasks sampled from all of the other task proposal methods (columns therein). We use ri (or cj) below to refer to row i (or column j) in the table.

Model	FREQ	X-C	I-C	I-C (ccnet)	S-C (ccnet)
BERT	43.1	43.3	37.7	38.7	66.3
SMLMT (uniform)	96.2	96.5	**78.4**	**68.5**	91.7
SMLMT (frequency)	96.8	96.9	**79.6**	**70.0**	91.0
SMLMT (clustering)	96.9	97.0	96.9	**75.2**	94.7
Sentence Cluster	69.2	71.2	**53.0**	**45.0**	98.9

Table 5.1. Analysis of task proposals. The columns are the different task proposal methods and rows are models trained on unsupervised task distributions. Low accuracy on a task distribution indicates harder to classify tasks or missing information in the training distribution (see Sec 5.5.2 for details).

We consider the following task proposal methods: Frequency (FREQ, c1): using the frequency-based word sampling in SMLMT; Inter-Cluster (X-C, c2): using the word-clustering approach explained in sec 5.3.2 but sampling all labels of task from different clusters; Intra-Cluster (I-C, c3&4): using the word-clustering approach explained in sec 5.3.2 which samples all labels of task from the same cluster; Sentence Cluster (S-C, c5): this is the sentence clustering approach to task proposal presented in sec 5.3.4. For evaluation, we consider 4-way tasks sampled from the above methods and evaluate average accuracy over 5000 tasks. We consider a BERT model (r1) which is not trained on the SMLMT distribution but is trained on the related masked language modeling (MLM) task. To enable evaluation of this model, we use it as a prototypical network model (Snell et al., 2017). We also consider meta-trained models trained on the SMLMT distribution with uniform sampling (Bansal et al., 2020b) (r2), frequency-based sampling (r3), and intra-cluster sampling (r4). Note that all models are trained on Wikipedia corpus.

Results are in Table 5.1. First, since BERT wasn't trained on any of the task distributions, we find low accuracy on all these tasks on r1, indicating that they contain information different than what is learned from MLM. Moreover, the highest accuracy of this model is on Sentence Cluster tasks (r1c5; random baseline is 25%), even though the domain of this task is quite different than the training data of BERT. Next, lets consider the vanilla SMLMT model which uses uniformly random word sampling to create the meta-learning task distribution. Interestingly, we find that it gives high accuracy on frequency-sampled tasks (r2c1). Similarly, accuracy is high on the inter-cluster tasks (r2c2), even though the model wasn't meta-trained directly on this distribution. More importantly, performance drops significantly ($\approx 18\%$) on the tasks sampled using the intra-cluster approach (r2c3). This performance drops even further ($\approx 10\%$; r2c4) when the tasks are sampled from a different domain (common crawl) than the training domain of the model (Wiki). Accuracy on Sentence Cluster is also very high (r2c5), without training on this distribution. Models trained on frequency-based sampling perform similarly (r3). We also show the performance of a model trained on tasks sampled using the intra-cluster approach. Note that this model was trained on Wikipedia corpus, and even though it was trained on intra-cluster tasks, we still see a significant performance drop on intra-cluster tasks on a different domain (r4c4 vs r4c3). Finally, consider models trained on the sentence clustering tasks. These perform poorly on all of the tasks proposed by SMLMT (r5c1–4), indicating that this task distribution does not contain the same amount of information as SMLMT.

In summary, these results indicate that: (1) the intra-cluster tasks are more difficult than frequency-based sampling, and inter-cluster tasks are as easy as uniform-sampling (r2c2) (2) sentence cluster tasks are the easiest among all task proposals (c5), and training on this task distribution leads to poor performance on the SMLMT distributions (r5c1–4; but not vice versa), indicating lack of information in this distribution as compared to SMLMT. From this analysis we expect intra-cluster task distribution to be richer as compared to the other alternatives and models meta-trained on these should improve downstream performance over the others. As we will see in the next section, the downstream performance improvements are highly correlated with these unsupervised evaluations.

5.5.3 Evaluation on diverse downstream classification tasks

5.5.3.0.1 Datasets We consider all 17 downstream tasks in Bansal et al. (2020b) and 2 additional sentence-pair tasks. We group performance on datasets by the type of the task: (1) *Sentiment classification*: 4 domains (Books, DVD, Kitchen, Electronics) of Amazon review binary sentiment datasets (Blitzer et al., 2007); (2) *Rating classification*: 4 domain of 3-way classification based on ratings of reviews from the above Amazon datasets, 1 dataset on 3-way classification of tweets about sentiment towards Airlines; (3) *Entity typing*: CoNLL-2003 (Sang and De Meulder, 2003) entity mention classification into 4 coarse types, MIT-Restaurant (Liu et al., 2013) task on classifying mentions in user queries about restaurants into 8 types; (4) *Sentence-pair classification*: Scitail, a scientific natural language inference dataset (Khot et al., 2018), RTE task on textual entailment and MRPC task on paraphrase classification from the GLUE benchmark (Wang et al., 2018b). (5) *Other text classification*: multiple social-media datasets on classifying tweets into (a) 2-way: political audience, bias or mention of a disaster, (b) 9-way: classifying based on political message, (c) 13-way: classifying emotion.

5.5.3.0.2 Evaluation Protocol We meta-train separate models on the self-supervised task distributions, without any access to the downstream supervised tasks. The models are then fine-tuned on the downstream task training sets which consist of $k = 8, 16, 32$ examples per class. Note that tasks can have different number of classes. Following Bansal et al. (2020b), we use the development set of Scitail and Amazon-Electronics to select the number of steps of fine-tuning for all models, all other hyper-parameters are kept the same as meta-training. Since few-shot performance is sensitive to the few examples in training, each model is fine-tuned on 10 sets for each task and the average test performance is reported with standard deviation.

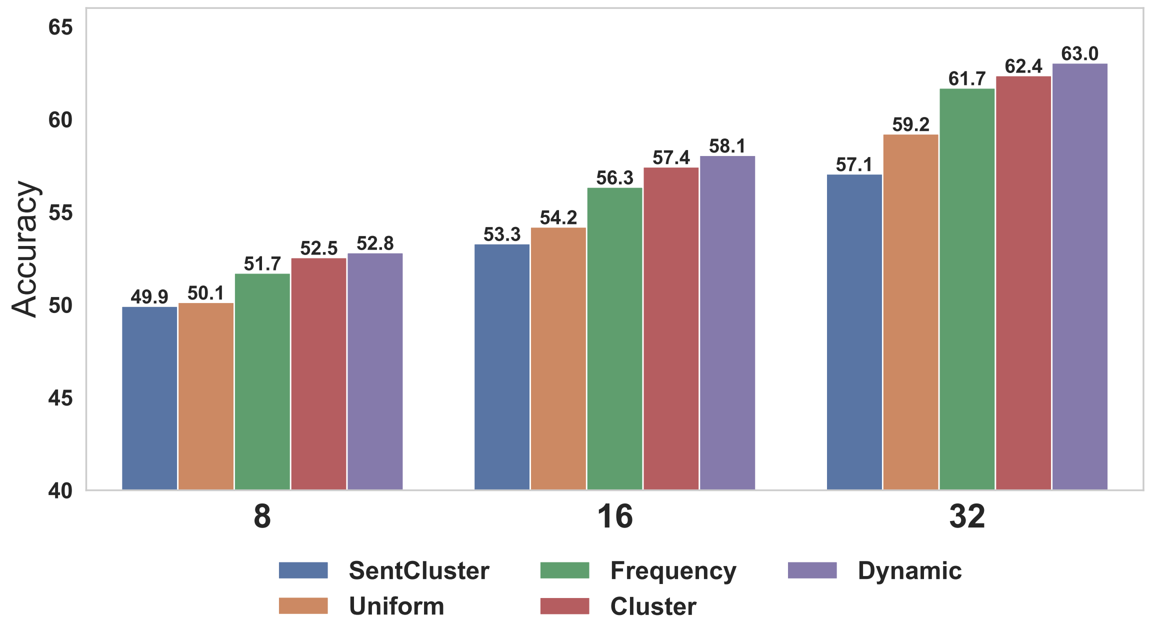


Figure 5.1. Overall average across 19 downstream tasks for the different task distributions proposed in this work. Cluster tasks and Dynamic curriculum lead to the best overall accuracy.

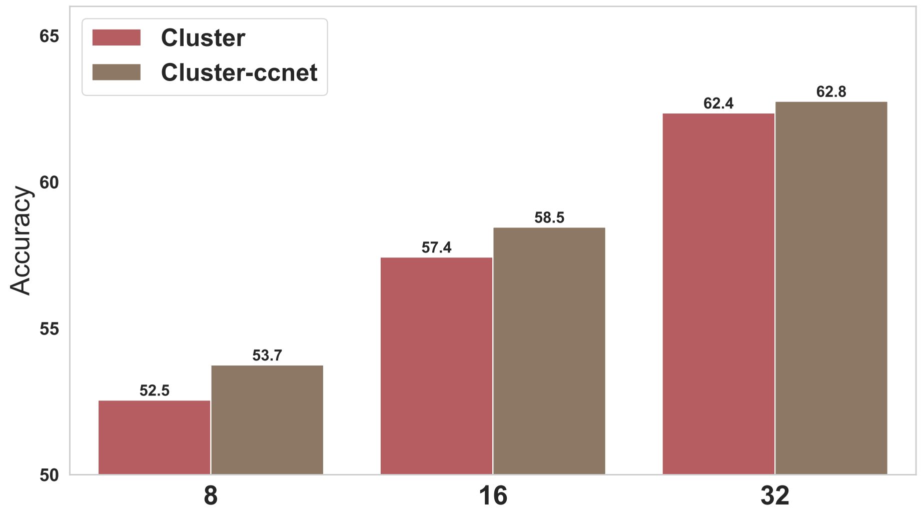


Figure 5.2. Changing domain of tasks from Wikipedia to CommonCrawl (ccnet) while keeping size of data, compute and model fixed. Overall average across 19 downstream tasks is shown. More diverse domain (ccnet) in training leads to improved down-stream accuracy.

Task Group	Model	k-shot		
		8	16	32
Sentiment Classification	Uniform	59.1 ± 5.2	62.6 ± 5.3	69.6 ± 5.1
	Frequency	60.2 ± 4.8	65.2 ± 5.1	74.0 ± 5.4
	Cluster	62.2 ± 5.3	67.3 ± 5.9	75.9 ± 4.0
	Dynamic	**63.6** ± 6.0	**69.3** ± 6.3	**77.3** ± 4.6
	SentCluster	61.1 ± 5.8	64.2 ± 5.7	70.4 ± 4.7
	Cluster-ccnet	**64.7** ± 6.6	**69.9** ± 7.1	**76.2** ± 6.3
Rating Classification	Uniform	41.9 ± 7.2	47.3 ± 7.2	52.9 ± 7.6
	Frequency	42.6 ± 6.9	49.2 ± 7.2	55.1 ± 7.7
	Cluster	45.2 ± 7.7	51.9 ± 6.6	56.5 ± 7.1
	Dynamic	**46.3** ± 8.1	**53.5** ± 7.0	**57.9** ± 7.3
	SentCluster	45.1 ± 8.8	48.7 ± 9.2	50.9 ± 9.0
	Cluster-ccnet	**45.2** ± 7.5	**52.1** ± 7.3	**57.1** ± 7.8
Entity Typing	Uniform	61.4 ± 2.6	72.5 ± 4.8	81.4 ± 3.0
	Frequency	64.0 ± 3.0	**73.0** ± 2.2	**82.1** ± 2.0
	Cluster	**64.5** ± 2.8	72.5 ± 2.6	81.3 ± 1.9
	Dynamic	62.4 ± 3.5	72.3 ± 3.3	81.6 ± 2.1
	SentCluster	51.7 ± 4.9	63.4 ± 4.5	73.4 ± 2.4
	Cluster-ccnet	**70.7** ± 2.8	**78.2** ± 3.1	**84.1** ± 2.5
Sentence Pair Classification	Uniform	52.9 ± 5.2	54.1 ± 4.7	57.4 ± 5.7
	Frequency	**59.5** ± 7.2	**61.0** ± 8.5	**63.6** ± 9.1
	Cluster	56.4 ± 5.3	59.5 ± 7.6	62.8 ± 8.6
	Dynamic	55.0 ± 4.9	57.8 ± 5.7	62.2 ± 8.5
	SentCluster	52.6 ± 4.7	52.9 ± 2.9	54.0 ± 3.8
	Cluster-ccnet	**55.9** ± 5.7	**58.5** ± 6.9	**62.9** ± 6.9
Other Text Classification	Uniform	44.8 ± 3.9	47.5 ± 2.3	49.4 ± 2.1
	Frequency	44.4 ± 3.5	47.3 ± 2.0	49.1 ± 1.9
	Cluster	45.0 ± 3.7	48.1 ± 2.0	49.5 ± 1.9
	Dynamic	**45.5** ± 3.5	**48.5** ± 2.2	**49.8** ± 1.9
	SentCluster	43.5 ± 4.1	45.7 ± 2.5	47.8 ± 1.7
	Cluster-ccnet	**46.6** ± 3.4	**48.9** ± 2.2	**49.9** ± 1.8

Table 5.2. Results on downstream tasks. Best performing model for each k and each task group is in bold and the second best is underlined.

5.5.3.0.3 Results. Table 5.2 shows the performance of all methods on different types of downstream tasks. We group datasets based on task type as described above and report the average performance over all the datasets in each group. First, note that the SentCluster approach is always inferior to any of the cloze-style approach, except on sentiment and rating classification where it is slightly better than SMLMT with Uniform sampling but worse than the other methods proposed here. Interestingly, replacing Uniform sampling with the simple Frequency sampling already leads to significant improvements throughout. Comparing the Cluster approach, we observe that this is better than Frequency on sentence-level tasks (like sentiment, rating,

Task	Model	8	16	32
MRPC	Clustering	54.63 ± 4.69	54.00 ± 3.63	58.28 ± 4.90
	+ SentPair	**55.88** ± 6.68	**57.13** ± 5.15	**60.12** ± 3.58
Scitail	Clustering	**60.63** ± 4.29	59.89 ± 4.20	67.89 ± 5.59
	+ SentPair	58.86 ± 4.81	**67.94** ± 2.92	**73.56** ± 2.79

Table 5.3. Ablation for training with and without contrastive sentence pair task.

others), while slightly worse or comparable on sentence-pair tasks and phrase-level classification tasks (entity typing). Overall, the word-clustering approach to sampling labels for SMLMT are more preferable as they are often among the two highest performing on any task group or close to the highest performance. Note that our unsupervised analysis in Sec 5.5.2 also reflected that training on the Cluster task distribution should be better compared to others. Finally, note that using the Dynamic curriculum over task sampling further improves the performance over cluster-based approach. This overall trend is also clearly reflected in the overall average performance across all the 19 tasks in Figure 5.1. Figure 5.2 further shows that, for the Cluster tasks, constructing tasks from more diverse domain such as CommonCrawl can improve downstream performance even when using the same amount of data for training.

5.5.3.0.4 Ablation over SentPair. We introduced the sentence pair task to enable better learning of sentence pair tasks such as natural language inference. These task remove the train-test mismatch in the input format as the sentence pair tasks contain pairs of sentences as input where as SMLMT only proposes single sentence classification tasks. To assess the efficacy of the SentPair task, we trained a word-cluster model with and without the SentPair task and evaluated it on few-shot sentence pair tasks of Scitail and MRPC. Results are in Table 5.3. We can see that the unsupervised SentPair task improves performance under most settings, sometimes by large margins up to 8% absolute.

Static Distribution	λ_t	8-shot	16-shot	32-shot
Cluster	0.5	54.0	58.7	63.0
Cluster	0.25	55.2	59.0	64.6
Frequency	Anneal	56.5	60.9	65.8
Cluster	Anneal	**56.8**	**61.7**	**66.4**

Table 5.4. Ablation: static tasks and the value of mixing proportion λ_t used in dynamic curriculum.

Model	1-shot		5-shot	
	$N = 5$	$N = 10$	$N = 5$	$N = 10$
SentCluster	34.2	21.2	52.3	36.3
Uniform	55.8	40.1	76.1	61.0
Frequency	57.6	41.6	<u>78.1</u>	61.5
Cluster	<u>60.4</u>	<u>45.2</u>	<u>78.1</u>	<u>63.9</u>
Cluster-ccnet	**60.1**	**46.0**	**81.2**	**67.6**

Table 5.5. Results on Fewrel 2.0 validation.

5.5.3.0.5 Ablation for dynamic curriculum. The dynamic curriculum over tasks requires two crucial choices: the static distribution and the value of mixing proportion λ_t. We ablate over choices for these in Fig. 5.4 which reports average performance over 5 tasks, one each from the task groups considered. We find that using the Cluster tasks, created from static pre-computer word-embeddings, works better than using Frequency-based tasks as the static distribution. Moreover, annealing λ_t from 0 to 1 over the training epochs is better than using a fixed value of λ_t throughout training.

5.5.4 Evaluation on FewRel 2.0 benchmark

FewRel (Han et al., 2018; Gao et al., 2019b) is a common benchmark for few-shot learning in NLP, which consists of many few-shot relation classification tasks created by sub-sampling from a pool of relation labels. The resemblance to the popular few-shot benchmarks like MiniImageNet (Vinyals et al., 2016) makes FewRel one of the few widely used datasets for training and evaluating NLP meta-learning methods.

Before submitting to the competition site for test set results, we first use the validation set to select the best model(s). Results on FewRel 2.0 validation set using

Model	1-shot		5-shot	
	$N = 5$	$N = 10$	$N = 5$	$N = 10$
Unsupervised				
Cluster	60.1	44.1	78.8	65.2
Cluster-ccnet	61.3	46.1	<u>80.4</u>	<u>67.7</u>
Supervised				
Proto-Adv (CNN)	42.2	28.9	58.7	44.4
Proto-Adv (BERT)	41.9	27.4	54.7	37.4
BERT-Pair	<u>67.4</u>	**54.9**	78.6	66.9
Cluster-ccnet	**67.7**	<u>52.9</u>	**84.3**	**74.1**

Table 5.6. Results on Fewrel 2.0 test set.

the different task distributions is shown in Figure 5.5. We observed that the Cluster approaches performs better than the other task proposals (see validation results in Supplementary). We then compare their test set performance with previously published results: the BERT-Pair, Proto-Adversarial (CNN), and Proto-Adversarial (BERT) are *supervised* meta-learning models trained on FewRel training data and using BERT or CNN as the text encoder. See Gao et al. (2019b) for details. Interestingly, our *unsupervised* meta-learned models that do not use any FewRel training data outperform the supervised baselines in the 5-shot setting. Performance is lower than BERT-Pair on 1-shot tasks, potentially because our models have not been trained for 1-shot tasks like BERT-Pair. Finally, fine-tuning our best model on the FewRel training data leads to the best overall performance.

CHAPTER 6

A RETROSPECTIVE COMPARISON WITH CONTEMPORARY METHODS

In this thesis, we developed meta-learning methods that improve few-shot classification. During the course of development of these methods, other contemporary methods for few-shot learning have been proposed. A particularly prominent approach is GPT-3 (Brown et al., 2020) that demonstrated improved few-shot learning by scaling the size of pre-trained transformer language models. Since this method was contemporaneous with the methods developed in this thesis, a direct comparison was not performed. Another successful approach to pre-training has been text-to-text pre-training, which was the approach used in developing the T5 model (Raffel et al., 2019). While T5 was not directly evaluated for few-shot learning, it is also a promising contender for few-shot learning owing to its generative pre-training. In this chapter, we take a retrospective look at some of these methods and directly compare them with the methods presented here.

6.1 Discussion of Contemporary Methods (GPT-3 and T5)

Pre-trained transformers have become widely adopted in NLP research. In the previous chapters, we compared the meta-learning methods with a particular pre-trained model, BERT (Devlin et al., 2019), demonstrating improvements over it in few-shot learning. An alternative to BERT-style pre-training involves generative pre-training (Raffel et al., 2019; Lewis et al., 2020) where a seq2seq model is trained to generate randomly masked tokens in text. An example of such a model is T5

(Raffel et al., 2019) that achieved state-of-the-art results on many evaluation benchmarks. T5 frames all downstream tasks as text generation tasks where the seq2seq transformer model generates the expected labels as a text sequence. In addition, T5 utilized supervised tasks alongside the unsupervised objective in its text to text pre-training, making it's pre-training similar to the hybrid-SMLMT approach developed in Chapter 4. While T5 demonstrated promising results for downstream classification and generation tasks, it wasn't evaluated specifically in the few-shot setting. We thus compare with this approach as well, where the trained T5 model is fine-tuned on the few-shot support set to generate the classification label.

Recently, a contemporary pre-training approach that has been highly successful for few-shot learning is GPT-3 (Brown et al., 2020). GPT-3 is the successor to previous pre-trained models that demonstrated the utility of language modeling for learning general purpose representations (Howard and Ruder, 2018; Peters et al., 2018; Radford et al., 2019). In particular, GPT-3 demonstrated that as one scales the model size and pre-training data for language model training, then the models eventually start to show a remarkable few-shot ability. Language modeling trains the network to predict the next tokens conditioned on a context. Thus, GPT-3 uses a *prompting* approach to few-shot learning, as opposed to a fine-tuning approach. The idea is to input the entire few-shot training data for a task (both instances and their labels) along with the test instance to be queried into the context, known as a prompt, and use the next token predictions from the model to predict the label for the query. This can be considered as an instance of model-based meta-learning (see 2.3.2.1), where the transformer acts as the meta-learning model. This is a memory based model, related to earlier meta-learning methods utilizing LSTMs for quickly learning the dynamics of new sequences (Younger et al., 2001). It was shown in Brown et al. (2020) that really large transformers, with $\sim$175 Billion parameters, trained as language models on internet scale corpora, can show competitive few-shot performance on a range of tasks that include classification, question-answering and generation.

We compare with GPT-3 and T5 models for few-shot classification, where GPT-3 is a prompting approach while T5 is a fine-tuning approach. Both of these methods model classification as a text generation problem. Note that although there has been a lot of subsequent work building on the approach of GPT-3, many other contemporary work often requires task-specific engineering or access to additional task-specific data (Schick and Schütze, 2020) for few-shot learning, which makes them incomparable to the methods developed here.

6.2 Experimental Results

The experimental setup is similar to that used in the previous chapters. We compare the following models: (1) GPT-3 (Ada, Babbage, Curie, DaVinci): prompt-based GPT-3 models (Brown et al., 2020) for few-shot learning, available from OpenAI in four different sizes; (2) T5 (Small, Base): text-to-text transformer models (Raffel et al., 2019) that are trained on a combination of supervised tasks (like GLUE) and an unsupervised span-denoising objective; (3) SMLMT: the completely self-supervised meta-learning model (chapter 4) that is trained on the SMLMT-cluster task distribution presented in the chapter 5; (4) Hybrid-SMLMT: meta-learning model (presented in chapter 4) that utilizes GLUE task during training in addition to the tasks in SMLMT-cluster.

The GPT-3 models are available through a beta API[1] which has a cost associated with each query. Due to this, we will only evaluate on a subset of 5 downstream tasks, chosen to be diverse and representative of the downstream classification tasks in our full evaluation suite. The tasks evaluated are: (1) Amazon Sentiment: binary

[1]`https://beta.openai.com/`

classification of sentiment in Amazon reviews, (2) Political Bias: classifying whether a tweet contains political bias, (3) Scitail: natural language inference on scientific text, (4) Airline Rating: classifying tweets about airlines into ternary sentiment, (5) CoNLL Entity Typing: classifying phrases in sentences into 4 types. These cover a diverse range of classification types and number of classes.

We run all the models, except GPT-3 DaVinci, on 2 different few-shot support sets and report the average. Due to the latency and cost of using GPT-3 DaVinci, at the time of writing, we only report its performance using one support set. OpenAI hasn't officially released the sizes of the four different GPT-3 models. However, based on the performance reported in Brown et al. (2020), a reasonable estimate is available (Gao, 2021). These models input the entire training data as a context string, or prompt. This limits the amount of training data that can be provided to these models without necessitating very large context length. The current API only allows for sequence lengths of upto 2048 for all the models except DaVinci. To make comparisons fair, we restrict evaluation to only the 8-shot setting as these models will not be able to take advantage of more examples. Moreover, the models can be sensitive to the order in which examples are presented and how the prompt is constructed, which are still active areas of research (Lu et al., 2021). We follow the prompt design in the OpenAI classification API for constructing the prompts and use a random order of the examples in each prompt to get an estimate of the average performance. The prompts are given in Appendix D. Finally, fine-tuning hyper-parameters for T5 were picked on the two validation tasks, following the same protocol as used for all the fine-tuning comparisons in the previous chapters.

Figure 6.1 summarizes the overall average accuracy for all the 8 models. Comparing SMLMT with the GPT-3 models, we find that SMLMT is better by a large margin than the two GPT-3 models: Ada and Babbge, which are 3× and 12×, respectively, of the size of SMLMT. However, the Curie and DaVinci models, which are

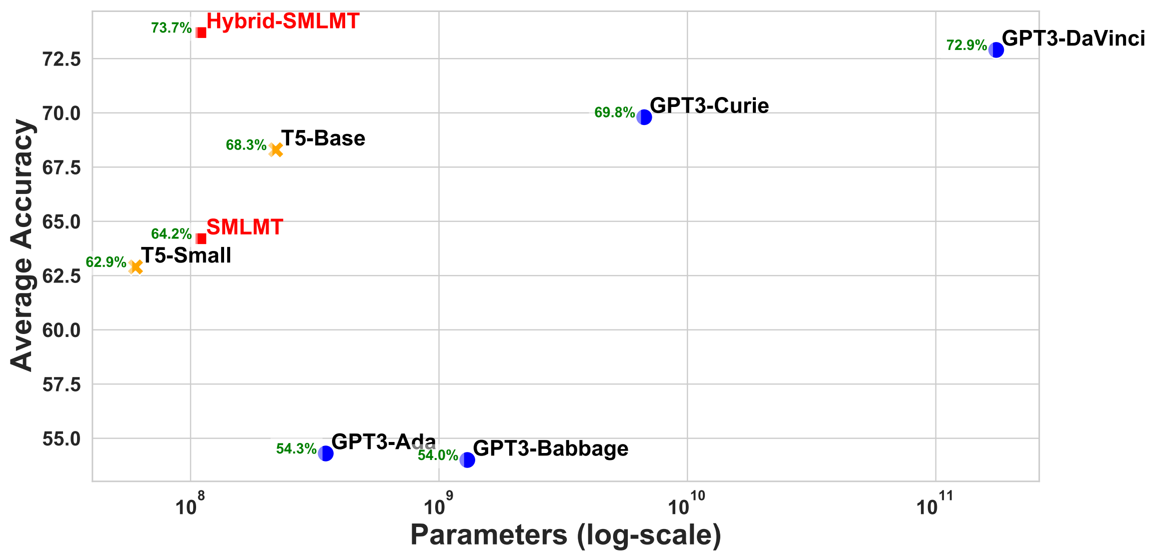


Figure 6.1. Average accuracy across the downstream tasks vs the model size for various contemporary approaches and the meta-learning methods proposed in this thesis (in red). Overall, the Hybrid-SMLMT model outperforms the 2x size T5 model and is competitive with the largest GPT-3 model that is more than 1590x its size.

Task	N	GPT-3				T5		SMLMT	Hybrid-SMLMT
		Ada	Babbage	Curie	DaVinci	Small	Base		
		350M	1300M	6700M	175000M	60M	220M	110M	110M
		3x	12x	61x	1591x	0.6x	2x	1x	1x
Sentiment Classification	2	75.2	71.4	90.0	94.3	79.9	87.7	65.1	82.7
Political Bias	2	50.3	50.3	65.8	62.9	53.6	61.4	69.8	67.2
Scitail NLI	2	56.4	60.7	56.1	60.1	55.6	68.0	58.3	82.6
Airline Rating	3	52.5	48.2	68.9	67.1	63.2	67.7	56.5	64.9
CoNLL Entity Typing	4	36.9	39.5	68.4	80.0	62.1	56.6	71.1	71.2
Overall Average		54.3	54.0	69.8	72.9	62.9	68.3	64.2	73.7

Table 6.1. Results on the individual downstream tasks.

many orders of magnitude larger, are better than SMLMT model. Note that all the GPT-3 and SMLMT models are completely self-supervised without any supervised task data. However, as noted in the GPT-3 paper (Brown et al., 2020), there was overlap in its training data with downstream supervised datasets. Next, consider the T5 and Hybrid-SMLMT models, which also leverage GLUE task data (Wang et al., 2018b) in their training in addition to self-supervised objectives, though none of the models use any of the evaluation task data during training. Interestingly, the Hybrid-SMLMT models performs better on average than all the other models – it is better than T5 models of comparable or 2× the size, better than GPT-3 models up to 61× its size, and competitive to the largest GPT-3 model, which is more than 1590× its size. Full set of results on the tasks can be found in Table 6.1.

These results are promising as they indicate the effectiveness of the proposed meta-learning methods and show that they still have a lot of potential to further improve few-shot learning. On the one end, they don't require prompt engineering, as is required for GPT-3-like models, and also don't require extensive task-specific labeled data (Schick and Schütze, 2020; Du et al., 2020a) to perform competitively with them. On the other end, the current evaluations of these models have only considered significantly smaller model size and training data scales compared to these contemporary methods. This indicates that scaling these models can be a lucrative avenue to improve performance even further. Note that among the two meta-learning models, Hybrid-SMLMT and SMLMT, the Hybrid-SMLMT model that combines supervised tasks along with self-supervised tasks seems to be better by a significant margin. This indicates that better self-supervised task design can still have a role to play in the future to further bridge the gap between these two paradigms. Finally, critically analyzing how scaling on the different dimensions of number of supervised tasks, self-supervised tasks and model size affects performance of meta-learning can be beneficial in future work.

CHAPTER 7
CONCLUSION

In this thesis, we have presented meta-learning methods to improve generalization of NLP models. In chapter 3, we presented an approach to optimization-based meta-learning that enables learning across tasks with a different number of classes. This enables meta-learning over diverse tasks making it competitive with typical multi-task learning applications. Since the availability of a diverse set of related tasks with labeled data is a major limitation for meta-learning in many NLP application domains, we introduced methods for meta-learning from unlabeled text. In chapter 4 and 5, we proposed methods to create meta-learning task distributions from unlabeled text. These methods enable meta-learning of general-purpose model initialization that show improved few-shot generalization through representation learning, learning key hyper-parameters like learning rates, and regularizing meta-overfitting when combined with supervised tasks. We show improvements in few-shot learning across diverse NLP classification tasks over self-supervised learning, multi-task learning, or supervised-only meta-learning.

While we focused on gradient-based and metric-based methods in this thesis and considered only the few-shot learning problem, the developed methods are more broadly applicable. In particular, the self-supervised tasks provide a source of meta-training tasks for many possible meta-learning applications and open up the possibility of exploring large-scale meta-learning in NLP for various meta problems beyond few-shot learning, including neural architecture search, continual learning, hyper-parameter learning, learning in low resource languages, and more.

7.1 Future Directions

In this concluding section, we discuss potential improvements to the methods presented in this thesis as well as some future applications of meta-learning that can benefit from using self-supervised task distributions. This is not an exhaustive list and is only intended to help motivate further applications of the methods presented.

Efficient Fine-tuning The methods developed in this thesis require fine-tuning the models on each task of interest. This can be inefficient in terms of the amount of compute and carbon emissions (Strubell et al., 2019) of using these models on each downstream task. An alternative to fine-tuning methods that has been explored recently is prompting (Brown et al., 2020). However, prompt-based methods lack some of the nice properties of fine-tuning which make them overly sensitive to prompt specifications (Lu et al., 2021) and require additional engineering per task for which there is usually no additional evaluation data, specially in the few-shot regime. Houlsby et al. (2019) proposed adapters to make fine-tuning more efficient. Adapters are small number of parameters added to the model that are fine-tuned per task, freezing the rest of the model. Utilizing such adapters in the inner loop of meta-learning will enable more efficient fine-tuning while also reducing the computational cost of meta-training. Alternatively, one can consider directly optimizing for finding a *sparse* subset of parameters to activate for each task. Both of these applications can be enabled using meta-learning on the self-supervised task distributions. Extensions to enable learning of soft-prompts (Lester et al., 2021) that can generalize to new tasks are also promising directions that can benefit from ideas of parameter generation (Bansal et al., 2020a) and self-supervised task distributions presented here.

Seq2Seq Modeling While the current work focused on classification tasks, it will be useful to extend the ideas to enable self-supervised learning of seq2seq models. Recently, many works have formulated a wide range of NLP tasks as sequence generation tasks (Vinyals et al., 2015; Lewis et al., 2020; Radford et al., 2019; Raffel et al., 2019), showing the versatility of this approach. Meta-learning can help learn better, more adaptable seq2seq models that can also be less sensitive to prompt specifications. Extending the SMLMT ideas to create self-supervised task distributions for meta-learning such seq2seq models will enable wider applicability of these models.

Continual Learning Most supervised machine learning methods consider learning from a static task, whereas typical deployments of machine learning models deal with a continuously changing distribution. Thus, we want models to continuously learn from ever-changing, non-stationary distributions and continuously learn about new, emerging concepts without catastrophic forgetting of previously learned concepts. Self-supervised meta-learning can immediately offer task distributions to optimize models directly for such continual learning scenarios, to learn both model architectures as well as update rules (learning algorithms) that enable forward transfer while minimizing forgetting. For instance, existing meta-learning methods (Javed and White, 2019; Beaulieu et al., 2020) for continual learning are limited by the availability of large task distributions to enable effective meta-learning. Considering sequences of SMLMT as task distribution should remedy this limitation, allowing one to meta-learn models that continuously learn about new words in context. Exploring such self-supervised task distributions for continual learning can help enable development of new methods for continual learning.

Learning Hyper-parameters The work presented in this thesis learned a crucial hyper-parameter for few-shot learning, the learning rate for fine-tuning. Instead of learning a single fixed learning rate, we explored alternatives like learning a per-layer learning rate that gave further improvements for few-shot learning. It will be interesting to explore learning of other hyper-parameters such as drop-out rates or learning task-specific learning rates for fine-tuning. Moreover, it is possible to explore hyper-parameters for improved fine-tuning of existing pre-trained models using the meta-learning methods presented in this work.

Multi-lingual Modeling The self-supervised methods developed previously only used text data in English language to construct task distributions. The approach is also applicable to learning multi-lingual models, where task distributions can be created from unlabeled text of each language and multi-lingual models can be meta-learned across the languages. Such self-supervised task distributions across languages can enable meta-learning of universal linguistic inductive biases, that has previously been shown feasible in a synthetic setting (McCoy et al., 2020). Such methods should particularly help few-shot learning in low resource languages and extensions like zero-shot generalization (Nooralahzadeh et al., 2020) of tasks to a low resource language from a high resource language could be of interest.

Multi-modal Learning It is well argued that in order to learn meaning, form alone is not sufficient (Bender and Koller, 2020; Bisk et al., 2020). Ultimately, it will be desirable to develop multi-modal task distributions that enable models to learn from data in many modalities, including – images, videos, audio, knowledge graphs, cross-modal tasks, embodied environments and interactive games. This will enable meta-learning of generalizable inductive biases across modalities, much like how humans learn from their varied experiences and leverage them to generalize to novel scenarios.

APPENDIX A

ADDITIONAL EXPERIMENTAL DETAILS AND RESULTS FOR LEOPARD

A.1 Datasets

Data Augmentation: Meta-learning benefits from training across many tasks. We thus create multiple versions of tasks with more than 2 classes by considering classifying between every pair of labels as a task. Existing methods (Vinyals et al., 2016; Snell et al., 2017; Finn et al., 2017) treat each random sample of labels from a pool of labels (for example in image classification) as a task. In order to create more diversity during training, we also create multiple versions of each dataset that has more than 2 classes, by considering classifying between every possible pair of labels as a training task. This increases the number of tasks and allows for more per-label examples in a batch during training. In addition, since one of the goals is to learn to classify phrases in a sentence, we modify the sentiment classification task (SST-2) in GLUE, which contains annotations of sentiment for phrases, by providing a sentence in which the phrase occurs as part of the input. That is, the input is the sentence followed by a separator token (Devlin et al., 2019) followed by the phrase to classify. An example of the input to all the models for the entity typing tasks can be found in Table A.1

Input	Label
are there any [authentic mexican]^1 restaurants in [the area]^2	^{1}Cuisine, ^{2}Location
are there any authentic mexican restaurants in the area [SEP] authentic mexican	Cuisine
are there any authentic mexican restaurants in the area [SEP] the area	Location

Table A.1. An example of an input from the MIT restaurants dataset. The first line is the actual example with two mentions. The next two lines are the input to the models – one for each mention.

We use the standard train, dev data split for GLUE and SNLI (Wang et al., 2018b; Bowman et al., 2015). For our ablation studies, on our target task we take 20% of the training data as validation for early stopping and sample from the remaining 80% to create the few-shot data for fine-tuning. For training MT-BERT we use dev data of the training task as the validation set. For meta-learning methods, prototypical network and LEOPARD, we use additional validation datasets as is typical in meta learning (Finn et al., 2017; Snell et al., 2017). We use unlabelled Amazon review data from apparel, health, software, toys, video as categorization tasks and labelled data from music, toys, video as sentiment classification task.

Details of the datasets are present in Table A.2.

Dataset	Labels	Training Size	Validation Size	Testing Size	Source
ARSC Domains	2	800	200	1000	Blitzer et al. (2007)
CoLA	2	8551	1042	—	Warstadt et al. (2019)
MRPC	2	3669	409	—	Dolan and Brockett (2005)
QNLI	2	104744	5464	—	Rajpurkar et al. (2016); Wang et al. (2018b)
QQP	2	363847	40431	—	Wang et al. (2018b)
RTE	2	2491	278	—	Dagan et al. (2005); Haim et al. (2006); Giampiccolo et al. (2007, 2008)
SNLI	3	549368	9843	—	Bowman et al. (2015)
SST-2	2	67350	873	—	Socher et al. (2013)
MNLI (m/mm)	3	392703	19649	—	Williams et al. (2017)
Scitail	2	23,596	1,304	2,126	Khot et al. (2018)
Airline	3	7320	—	7320	`https://www.figure-eight.com/data-for-everyone/`
Disaster	2	4887	—	4887	`https://www.figure-eight.com/data-for-everyone/`
Political Bias	2	2500	—	2500	`https://www.figure-eight.com/data-for-everyone/`
Political Audience	2	2500	—	2500	`https://www.figure-eight.com/data-for-everyone/`
Political Message	9	2500	—	2500	`https://www.figure-eight.com/data-for-everyone/`
Emotion	13	20000	—	20000	`https://www.figure-eight.com/data-for-everyone/`
CoNLL	4	23499	5942	5648	Sang and De Meulder (2003)
MIT-Restaurant	8	12474	—	2591	Liu et al. (2013) `https://groups.csail.mit.edu/sls/downloads/restaurant/`

Table A.2. Dataset statistics for all the datasets used in our analysis. ”-” represent data that is either not available or not used in this study. We have balanced severely unbalanced datasets(Political Bias and Audience) as our training data is balanced. To create training data for few shot experiments we sample 10 datasets for each k-shot. *Sec A.1 for more details

A.1.1 Test Datasets

The tasks and datasets we used for evaluating performance on few-shot learning are as follows:

1. Entity Typing: We use the following datasets for entity typing: CoNLL-2003 (Sang and De Meulder, 2003) and MIT-Restaurant (Liu et al., 2013). Note that

we consider each mention as a separate labelled example. CoNLL dataset consists of text from news articles while MIT dataset contains text from restaurant queries.

2. Sentiment Classification: We use the sentiment annotated data from Amazon Reviews dataset (Blitzer et al., 2007) which contains user reviews and the binary sentiment for various domains of products. We use the Books, DVD, Electronics, and Kitchen & Housewares domains, which are commonly used domains in the literature (Yu et al., 2018).

3. Rating Classification: We use the ratings from the Amazon Reviews dataset (Blitzer et al., 2007) which is not annotated with overall sentiment, and consider classifying into 3 classes: rating ≤ 2, rating $= 4$ and rating $= 5$.

4. Text Classification: We use multiple text classification datasets from crowdflower[1]. These involve classifying sentiments of tweets towards an airline, classifying whether a tweet refers to a disaster event, classifying emotional content of text, classifying the audience/bias/message of social media messages from politicians. These tasks are quite different from the training tasks both in terms of the labels as well as the input domain.

5. NLI: We use the SciTail dataset (Khot et al., 2018), which is a dataset for entailment created from science questions.

A.2 Additional Results

Table A.3 shows the dev-set accuracy of our trained MT-BERT model on the various training tasks.

[1]https://www.figure-eight.com/data-for-everyone/

	MNLI(m/mm)	QQP	QNLI	SST-2	CoLA	MRPC	RTE	SNLI	Average
MT-BERT	82.11	89.92	89.62	90.7	81.30	84.56	78.34	89.97	**85.82**

Table A.3. Dev-set accuracy on the set of train tasks for multi-task BERT.

Figure A.1 shows the target task performance as a function of training tasks for all k. Note that the effect of training tasks starts to decrease as k increases.

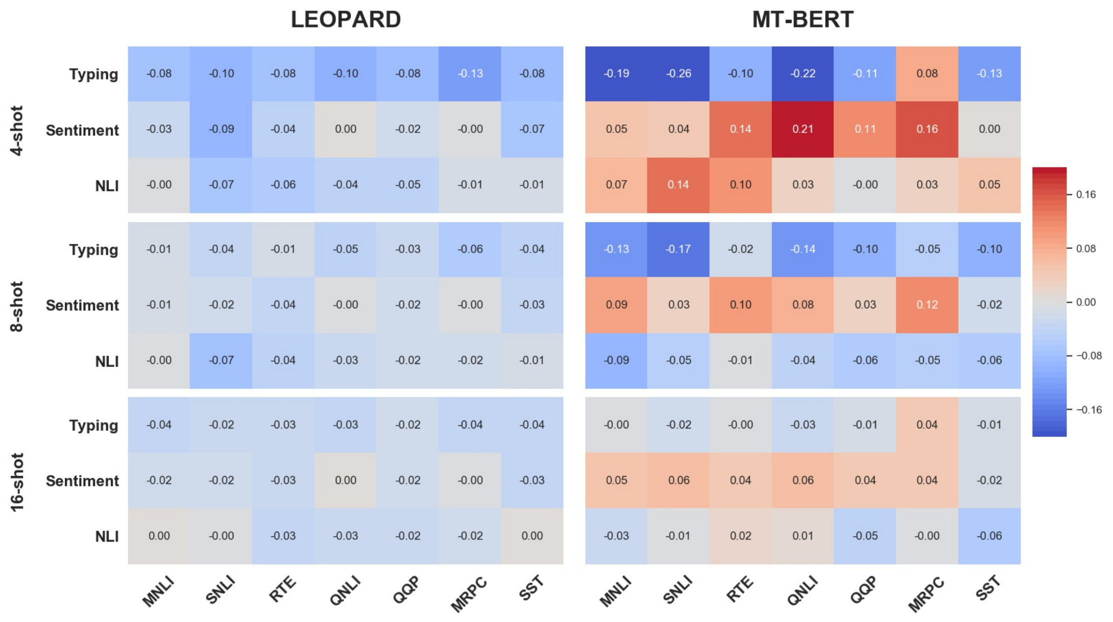


Figure A.1. Analyzing target task performance as a function of training tasks (best viewed in color). Heatmaps on the left are for LEOPARD and on the right are for MT-BERT. Each column represents one held-out training task (name on x-axis) and each row corresponds to one target task (name on y-axis). Each cell is the relative change in performance on the target task when the corresponding training task is held-out, compared to training on all the train tasks. Dark blue indicates large drop, dark red indicates large increase and grey indicates close to no change in performance. In general, LEOPARD's performance is more consistent compared to MT-BERT indicating that meta-training learns more generalized initial parameters compared to multi-task training. Dependence on training tasks is stronger for small k, however even for $k = 16$ MT-BERT shows more sensitivity to training tasks.

A.3 Hyperparameters

Table A.4 shows the hyper-parameter search range as well as the best hyper-parameters for MT-BERT, Proto-BERT and LEOPARD. We use same hyperparameters for prototypical networks except those not relevant to them. For fine-tuning we separately tune number of iterations, and batch size for each k shot for all the baselines. We also tuned warm-up (Devlin et al., 2019) in $\{0, 0.1\}$ and used 0.1 for all the methods. For MT-BERT we found 10 epochs , batch size 8 to be best for 4-shot, 5 epochs, batch size 8 to be best for 8-shot and 5 epoch with 16 batch size gave the best performance for 16 shot. For MT-BERT$_{\text{softmax}}$ we found 125 epoch, batch size 4 to be best for 4-shot, 125 epochs, batch size 4 to be best for 8-shot and 125 epochs with batch size 4 gave the best performance for 16-shot. For BERT$_{\text{base}}$ 10 epochs, batch size 8 for 4 shot, 5 epochs, 16 batch size for 8 shot and 10 epochs, batch size 16 for 16 shot gave the best performance. For MT-BERT$_{\text{reuse}}$ we found 10 epochs , batch size 8 to be best for 4-shot, 5 epochs, batch size 8 to be best for 8-shot and 5 epoch with 16 batch size gave the best performance for 16 shot. Note, for LEOPARD we use learned per-layer learning rates with SGD. We use the following values: 150 epochs for 4-shot, 100 epochs for 8-shot, 100 epochs for 16-shot.

Parameter	Search Space	MT-BERT	Proto-BERT	LEOPARD
Attention dropout	[0.1, 0.2, 0.3]	0.2	0.3	0.1
Batch Size	[16, 32]	32	16	10
Class Embedding Size	[128, 256]	—	256	256
Hidden Layer Dropout	[0.1, 0.2, 0.3]	0.1	0.2	0.1
Inner Loop Learning Rate	—	—	—	Meta-SGD (per-layer)
Min Adapted Layer (ν)	[0, 5, 8, 10, 11]	—	—	0
Outer Loop Learning Rate	[1e-4, 1e-5, 2e-5, 4e-5, 5e-5]	2e-05	2e-05	1e-05
Adaptation Steps (G)	[1, 4, 7]	—	—	7
Top layer [CLS] dropout	[0.45, 0.4, 0.3, 0.2, 0.1]	0.1	0.2	0.1
Train Word Embeddings(Inner Loop)	[True, False]	—	—	True
Data Sampling	[Square Root, Uniform]	Square Root	Square Root	Square Root
Lowercase text	False	False	False	False

Table A.4. Hyper-parameter search space and best hyper-parameters for all models.

APPENDIX B

ADDITIONAL EXPERIMENTAL DETAILS AND RESULTS FOR SELF-SUPERVISED META-LEARNING

B.1 Training Algorithm

The meta-training algorithm is given in 2. Note that π^w are the parameters for the warp layers and π are the remaining transformer parameters. $L_T(\cdot)$ is the cross-entropy loss for N-way classification in task T, calculated from the following prediction:

$$p(y|x) = softmax\left\{\mathbf{W}\; h_\phi(f_\pi(x)) + \mathbf{b}\right\} \tag{B.1}$$

$g_\psi(\cdot)$ and h_ϕ are a two layer MLP with tanh non-linearity Bansal et al. (2020a).

B.2 Additional Results

Fig.B.1 and Fig. B.2 show the CCA similarity on the two datasets: CoNLL and Scitail. Table B.2 shows the accuracy for different model sizes on the three evaluation datasets: Scitail, Amazon DVD, CoNLL.

B.3 Datasets

Supervised Training Tasks: We selected the GLUE Wang et al. (2018b) benchmark tasks: MRPC, SST, MNLI (m/mm), QQP, QNLI, CoLA, RTE, and SNLI Bowman et al. (2015) as the supervised training tasks for the meta-training phase. We used the standard train/dev/test split.

Test Tasks: These are same as the tasks used in Bansal et al. (2020a).

Algorithm 2 Meta-Training

```
Require: SMLMT task distribution $\mathcal{T}$ and supervised tasks $\mathcal{S}$, model parameters
    $\{\pi^w, \pi, \phi, \psi, \alpha\}$, adaptation steps $G$, learning-rate $\beta$, sampling ratio $\lambda$
    Initialize $\theta$ with pre-trained BERT-base;
 1: while not converged do
 2:   for task_batchsize times do
 3:     $t \sim Bernoulli(\lambda)$
 4:     $T \sim t \cdot \mathcal{T} + (1 - t) \cdot \mathcal{S}$
 5:     $\mathcal{D}^{tr} = \{(x_j, y_j)\} \sim T$
 6:     $C^n \leftarrow \{x_j | y_j = n\}; \quad N \leftarrow |C^n|$
 7:     $w^n, b^n \leftarrow \frac{1}{|C^n|} \sum_{x_j \in C^n} g_\psi(f_\pi(\mathcal{D}^{tr}))$
 8:     $\mathbf{W} \leftarrow [w^1; \ldots; w^N]; \quad \mathbf{b} \leftarrow [b^1; \ldots; b^N]$
 9:     $\theta \leftarrow \{\pi, \phi, \mathbf{W}, \mathbf{b}\}; \quad \theta^{(0)} \leftarrow \theta$
10:     $\Theta \leftarrow \{\pi^w, \pi, \psi, \alpha\}$
11:     $\mathcal{D}^{val} \sim T$
12:     $q_T \leftarrow 0$
13:     for $s := 0 \ldots G - 1$ do
14:       $\mathcal{D}_s^{tr} \sim T$
15:       $\theta^{(s+1)} \leftarrow \theta^{(s)} - \alpha \nabla_\theta \mathcal{L}_T(\{\Theta, \theta^{(s)}\}, \mathcal{D}_s^{tr})$
16:       $q_T \leftarrow q_T + \nabla_\Theta \mathcal{L}_T(\{\Theta, \theta^{(s+1)}\}, \mathcal{D}^{val})$
17:     end for
18:   end for
19:   $\Theta \leftarrow \Theta - \beta \cdot \sum_T \frac{q_T}{G}$
20: end while
```

B.4 Implementation Details

B.4.1 Training Hyper-parameters

Table B.1 lists all the hyper-parameters for the Hybrid-SMLMT and SMLMT models. Both models use the same set of hyper-parameters, the difference being in the training tasks. Note, some hyper-parameters such as λ are not valid for SMLMT. We followed Devlin et al. (2019) in setting many hyper-parameters like dropouts, and Bansal et al. (2020a) in setting hyper-parameters related to meta-learning. We use first-order MAML. Meta-training is run for only 1 epoch, so the model always trains on a new SMLMT in every batch. This corresponds to about 500,000 steps of updates during training.

Hyper-parameter	Value
Tasks per batch	4
Support samples per task	80
Query samples per task	10
Number of classes in SSLMT	[2,3,4]
d	256
Attention dropout	0.1
Hidden Layer Dropout	0.1
Outer Loop Learning Rate	1e-05
Adaptation Steps (G)	7
λ	0.5
Meta-training Epochs	1
Lowercase text	False
Sequence Length	128
Learning-rate Warmup	10% of steps

Table B.1. Hyper-parameters.

B.4.2 Sampling for Hybrid-SMLMT

We restrict the word vocabulary for task creation with a term frequency of at least 50 in the corpus. This is then used to create tasks in SMLMT as described. This word vocabulary is discarded at this point and the data is word-piece tokenized using the BERT-base cased model vocabulary for input to the models. Note that after a supervised task is selected to be sampled based on λ, it is sampled proportional to the square-root of the number of samples in the supervised tasks following Bansal et al. (2020a).

B.4.3 Fine-tuning Hyper-parameter

We tune the number of fine-tuning epochs and batch-size using the development data of Scitail and Amazon Electronics tasks following Bansal et al. (2020a). Note that best values are determined for each k. Epochs search range is [5, 10, 50, 100, 150, 200, 300, 400] and batch-size search range is $[4, 8, 16]$. The selected values, for $k = (4, 8, 16, 32)$, are: (1) Hybrid-SMLMT: epochs $= (300, 350, 400, 200)$, batchsize $= (8, 16, 8, 16)$; (2) SMLMT: epochs $= (100, 200, 150, 200)$, batchsize $= (8, 16, 8, 16)$.

	k	Small (29.1 M)		Medium (41.7 M)		Base (110.1 M)	
		MT-BERT	Our	MT-BERT	Our	MT-BERT	Our
Scitail	4	**57.55** ± 8.64	55.70 ± 9.75	54.07 ± 5.43	**54.17** ± 10.34	63.58 ± 14.04	**75.98** ± 2.93
	8	60.13 ± 5.77	**63.85** ± 3.19	55.88 ± 7.04	**60.17** ± 5.86	65.77 ± 10.53	**76.89** ± 2.28
	16	65.00 ± 2.73	**66.98** ± 1.72	63.84 ± 3.91	**65.23** ± 2.23	72.50 ± 10.01	**79.71** ± 1.27
	32	65.40 ± 4.54	**67.23** ± 2.05	**67.40** ± 2.99	65.32 ± 2.76	74.04 ± 03.09	**82.15** ± 1.29
Amazon DVD	4	60.99 ± 5.05	**71.83** ± 6.69	63.66 ± 7.43	**74.72** ± 3.74	64.04 ± 8.53	**83.60** ± 1.49
	8	63.38 ± 6.91	**73.49** ± 1.34	67.30 ± 4.39	**75.24** ± 1.17	66.37 ± 9.12	**83.75** ± 0.61
	16	67.99 ± 2.05	**72.88** ± 0.66	70.73 ± 2.88	**74.72** ± 1.58	68.52 ± 6.76	**82.91** ± 1.20
	32	69.50 ± 1.28	**73.24** ± 1.33	71.35 ± 2.83	**75.20** ± 2.44	76.38 ± 2.00	**84.13** ± 0.68
CoNLL	4	31.57 ± 3.06	**40.91** ± 5.72	35.00 ± 5.11	**43.12** ± 2.60	59.47 ± 4.40	**59.60** ± 5.82
	8	35.97 ± 3.96	**45.96** ± 4.58	36.40 ± 3.41	**49.04** ± 2.84	64.72 ± 5.60	**73.55** ± 3.44
	16	38.89 ± 2.84	**53.14** ± 1.70	39.41 ± 2.21	**55.05** ± 2.54	70.78 ± 2.92	**80.85** ± 2.15
	32	44.50 ± 2.56	**60.74** ± 1.96	44.57 ± 1.64	**62.59** ± 1.83	81.09 ± 1.09	**87.45** ± 1.12

Table B.2. k-shot performance for three models sizes.

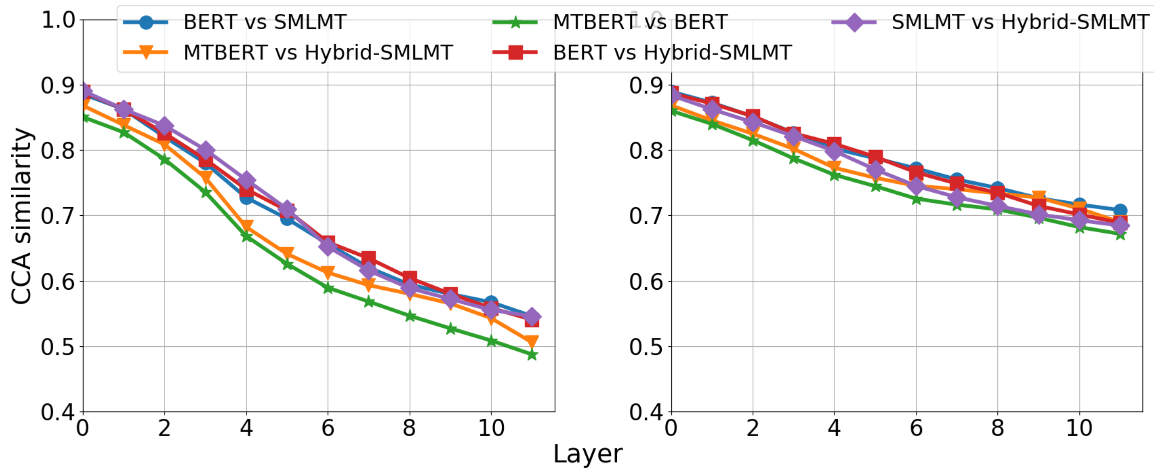


Figure B.1. Cross-model CCA similarity for each layer of the transformer after fine-tuning. Left plot is on CoNLL and right on Scitail.

Expected overall average validation accuracy for these hyper-parameters, for $k \in (4, 8, 16, 32)$ are: (1) Hybrid-SMLMT: (0.80, 0.81, 0.83, 0.84); (2) SMLMT: (0.54, 0.56, 0.62, 0.68). Hyper-parameters for BERT, LEOPARD and MT-BERT are taken from Bansal et al. (2020a).

B.4.4 Training Hardware and Time

We train the SMLMT and Hybrid-SMLMT models on 4 V100 GPUs, each with 16GB memory. Owing to the warp layers, our training time per step and the GPU

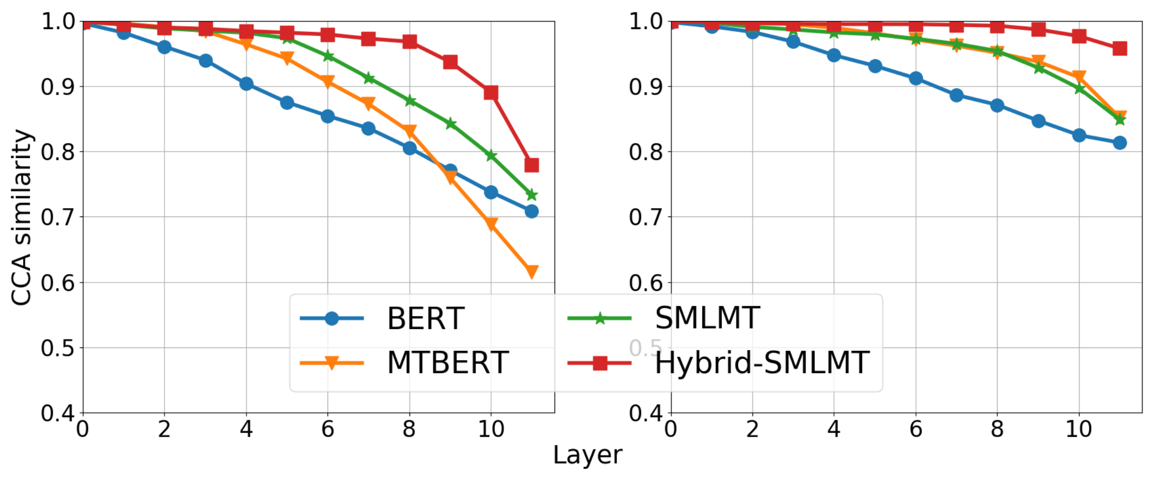


Figure B.2. CCA similarity for each layer of the same model before and after fine-tuning. Left plot is on CoNLL and right on Scitail.

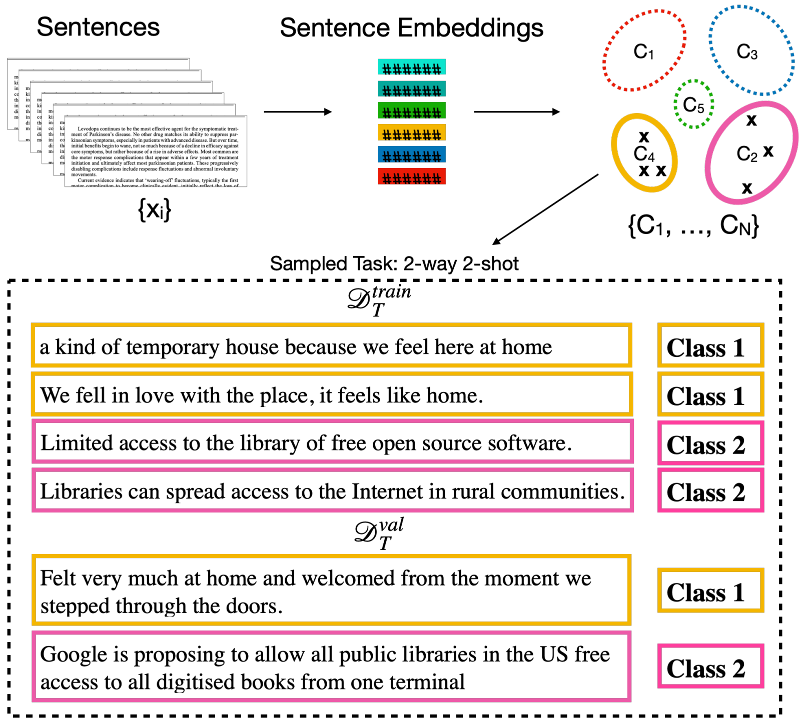
memory footprint is lower than LEOPARD (Bansal et al., 2020a). However, our training typically runs much longer as the model doesn't overfit unlike LEOPARD (see learning rate trajectory in chapter 4). Meta-training takes a total of 11 days and 14hours.

APPENDIX C

ADDITIONAL EXPERIMENTAL DETAILS AND RESULTS FOR DIVERSE DISTRIBUTIONS OF SELF-SUPERVISED TASKS

C.1 Illustration of SentCluster Tasks

Example of SentCluster can be seen in Figure C.1.



a kind of temporary house because we feel here at home	Class 1
We fell in love with the place, it feels like home.	Class 1
Limited access to the library of free open source software.	Class 2
Libraries can spread access to the Internet in rural communities.	Class 2
Felt very much at home and welcomed from the moment we stepped through the doors.	Class 1
Google is proposing to allow all public libraries in the US free access to all digitised books from one terminal	Class 2

Figure C.1. Illustration of SentCluster approach.

C.2 Additional Experiment Results

Full distribution of results in down-stream takss for the various self-supervised tasks can be seen in Fig. C.2

Hyper-parameter	Value
Tasks per batch	4
d	256
Attention dropout	0.1
Hidden Layer Dropout	0.1
Outer Loop Learning Rate	1e-05
Adaptation Steps (G)	7
Meta-training Epochs	1
Lowercase text	False
Sequence Length	128
Learning-rate Warmup	10% of steps
SentPair ratio α	$\frac{1}{16}$
Number of Tasks	4 Million
Support samples per task	80
Query samples per task	10
Number of classes for tasks	[2,3,4,5]
Number of Clusters M	500
Number of Clusters in SentCluster	200k
λ_t in Dynamic	Anneal
m interval in Dynamic	5000

Table C.1. Hyper-parameters. The parameters relating to the task distributions are in the bottom section of the table.

C.3 Fine-tuning Hyper-parameters

The meta-learning methods learn the learn rate for fine-tuning, thus we only tune the number of steps to run fine-tuning by using development data from 2 tasks (Scitail, Amazon Electronics), following Bansal et al. (2020a,b). We found that running fine-tuning until the loss on the support set is small ($<= 0.01$) is an alternative that also performs competitively and does not require tuning the number of steps. The reported results followed the previous approach and the tuned number of steps of fine-tuning for $k = 8, 16, 32$ respectively were: (1) Uniform: 100,75,100 (2) Frequency: 25,150,75 (3) Cluster: 75,50,75 (4) Cluster-ccnet: 150,200,75 (5) SentCluster: 100,250,25 (6) Dynamic: 10, 100, 200. On FewRel we found 20 steps of updates on the support set to perform well on the validation data for all settings.

C.4 Other Implementation Details

Since the Fewrel tasks consist of entity pair in the sentence it is important to mark these entities which define the relation to be classified. We used unused tokens in the BERT vocabulary to mark the positions of the entity mentions. Note the in the unsupervised models these unsused tokens get a zero-embedding and are only fine-tuned from the 1-shot or 5-shot support sets.

Hyper-parameters for meta-training are listed in Table C.1. Dataset statistics for downstream classification tasks can be found in Bansal et al. (2020a) and few-shot splits can be downloaded from `https://github.com/iesl/leopard`.

Training Hardware: The models were trained on 32 V100 GPU. Training takes about 42 hours.

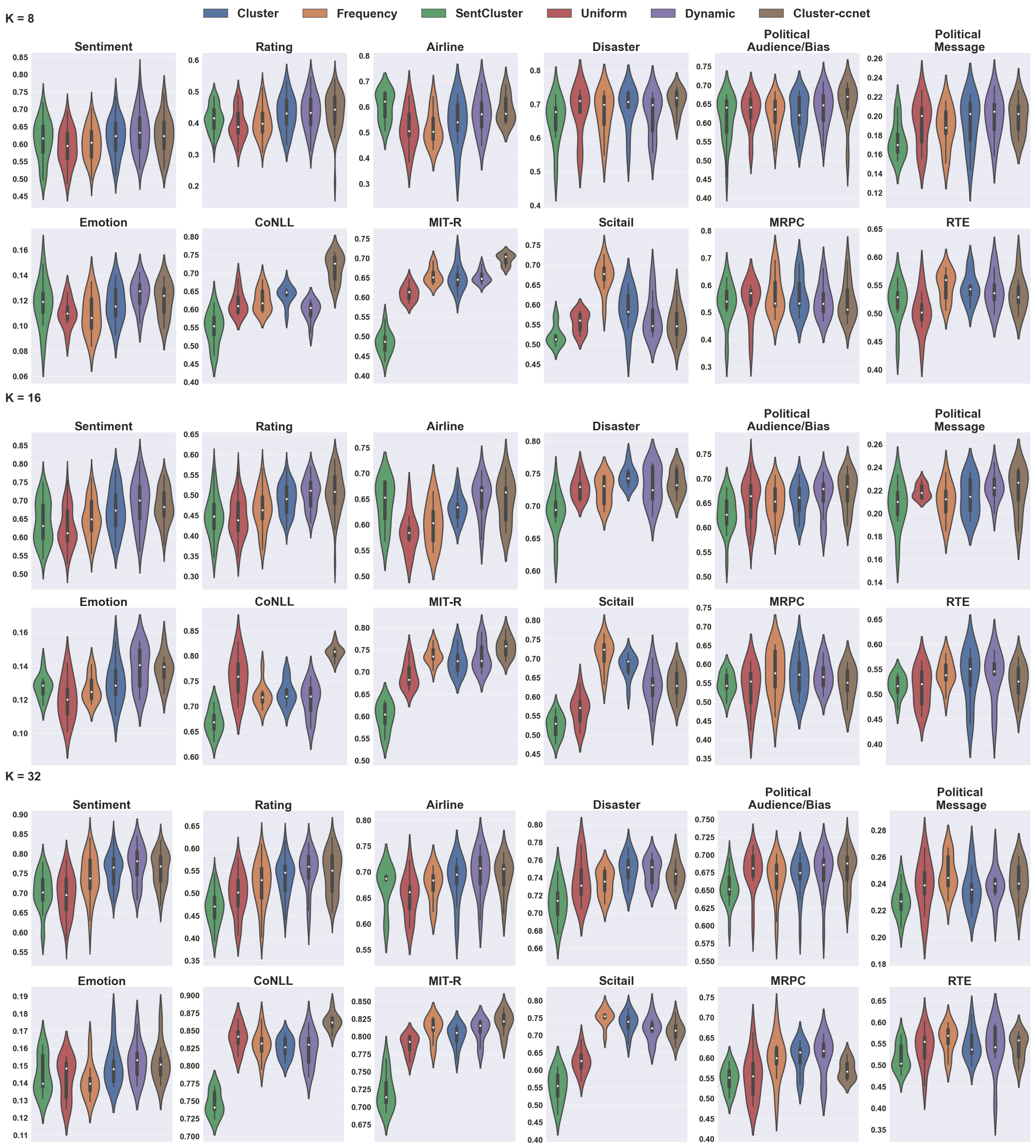


Figure C.2. Results across all tasks. Sentiment and Rating are average of 4 domains used in Bansal et al. (2020a).Each violin plot for a model shows the full distribution of accuracy across multiple runs (and domains).

APPENDIX D

ADDITIONAL EXPERIMENTAL DETAILS FOR GPT-3 AND T5 COMPARISONS

GPT-3 requires the few-shot training data to be concatenated into a prompt for the model. The following are examples of the prompts used for evaluations on all the 5 tasks.

SciTail

```
Please classify a piece of text into categories.

Text: Skin Mesh Human skin has a layered structure consisting of the dermis and epidermis.
Question: Most skin structures originate in the dermis. True or False?
Answer: False
———
Text: The four basic tissue types are epithelial tissue, connective tissues, nervous tissue,
    and muscle tissue.
Question: Four types of tissue are found in animals. True or False?
Answer: True
———
Text: Trees produce oxygen as a byproduct through the photosynthesis process.
Question: Oxygen is made by trees and other plants during photosynthesis. True or False?
Answer:
```

Sentiment Electronics

```
Please classify a piece of text into categories.

Text: The mouse is perfect for games. I use it to play ET and is great. The software provided
    by Logitech is configurable in all ways
Category: Positive
———
Text: I purchased this product and couldn't get it to work on a PC, laptop, or pda ( cingular
    8125 ). Not only does it not register in any SD / MiniSD card reader, but the
    craftsmanship appears to be suspect as well. I am currently pursuing a refund
Category: Negative
———
Text: Code length of this product is very small. I had to buy an extention cord. The sound
    quality is not bad
Category:
```

Airline

Please classify a piece of text into categories.

Text: @SouthwestAir yall still fly in the cold right?
Category: Neutral
———
Text: @SouthwestAir Great, thank you. Best of luck dealing with this horrible winter.
Category: Positive
———
Text: @united I was on UA3782 and it was Cancelled Flightled. I'm waiting at customer service.
Category: Negative
———
Text: @SouthwestAir please reply to my DM
Category:

Political Bias

Please classify a piece of text into categories.

Text: The 1st Amendment protects # ReligiousFreedom for everyone & amp ; no American should be compelled to violate their convictions. #HobbyLobby
Category: Political
———
Text: Thanks for your support! MT @ SenOrrinHatch : Today is # NationalPediatricBrainCancerAwarenessDay. Hope you'll join me in fighting this disease
Category: Neutral
———
Text: Regrettably, the House failed to approve its proposal for a new #FarmBill. Frankly, I was shocked by the outcome.
Category:

CoNLL Entity Typing

Please classify a piece of text into categories.

Text: India. Classify: India.
Category: Location
———
Text: Leading rider Jason Weaver received a 21 – day ban from the disciplinary committee of the Jockey Club on Wednesday. Classify: Jason Weaver.
Category: Person
———
Text: Balloting inside Bosnia is scheduled for September 14, when citizens are slated to elect municipal and cantonal assemblies, separate Moslem – Croat and Serb parliaments, a national House of Representatives and a three – man Presidency. Classify: House of Representatives.
Category: Organization
———
Text: Leading stories in the Greek financial press :. Classify: Greek.
Category: Other
———
Text: HELIBOR INTEREST RATES LARGELY UNCHANGED. Classify: HELIBOR.
Category:

BIBLIOGRAPHY

Alessandro Achille, Michael Lam, Rahul Tewari, Avinash Ravichandran, Subhransu Maji, Charless C Fowlkes, Stefano Soatto, and Pietro Perona. 2019. Task2vec: Task embedding for meta-learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6430–6439.

Maruan Al-Shedivat, Trapit Bansal, Yura Burda, Ilya Sutskever, Igor Mordatch, and Pieter Abbeel. 2018a. Continuous adaptation via meta-learning in nonstationary and competitive environments. In *International Conference on Learning Representations.*

Maruan Al-Shedivat, Trapit Bansal, Yuri Burda, Ilya Sutskever, Igor Mordatch, and Pieter Abbeel. 2018b. Continuous adaptation via meta-learning in nonstationary and competitive environments. In *Proceedings of the International Conference on Learning Representations.*

Jay Alammar. 2018. The illustrated transformer [blog post]. *https://jalammar.github.io/illustrated-transformer/.*

Marcin Andrychowicz, Misha Denil, Sergio Gómez Colmenarejo, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando de Freitas. 2016. Learning to learn by gradient descent by gradient descent. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 3988–3996.

Antreas Antoniou, Harrison Edwards, and Amos Storkey. 2018. How to train your maml. *arXiv preprint arXiv:1810.09502.*

Sanjeev Arora, Hrishikesh Khandeparkar, Mikhail Khodak, Orestis Plevrakis, and Nikunj Saunshi. 2019. A theoretical analysis of contrastive unsupervised representation learning. In *36th International Conference on Machine Learning, ICML 2019*, pages 9904–9923. International Machine Learning Society (IMLS).

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450.*

R Harald Baayen. 2002. *Word frequency distributions*, volume 18. Springer Science & Business Media.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473.*

Trapit Bansal, David Belanger, and Andrew McCallum. 2016. Ask the GRU: Multi-task learning for deep text recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 107–114.

Trapit Bansal, Karthick Gunasekaran, Tong Wang, Tsendsuren Munkhdalai, and Andrew McCallum. 2021. Diverse distributions of self-supervised tasks for meta-learning in NLP. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5812–5824.

Trapit Bansal, Rishikesh Jha, and Andrew McCallum. 2020a. Learning to few-shot learn across diverse natural language classification tasks. In *Proceedings of the 28th International Conference on Computational Linguistics (COLING)*, pages 5108–5123.

Trapit Bansal, Rishikesh Jha, Tsendsuren Munkhdalai, and Andrew McCallum. 2020b. Self-supervised meta-learning for few-shot natural language classification tasks. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 522–534.

Trapit Bansal, Da-Cheng Juan, Sujith Ravi, and Andrew McCallum. 2019. A2N: Attending to neighbors for knowledge graph inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4387–4392.

Trapit Bansal, Arvind Neelakantan, and Andrew McCallum. 2017. RelNet: End-to-end modeling of entities & relations. *NeurIPS Workshop on Automated Knowledge Base Construction (AKBC)*.

Trapit Bansal, Jakub Pachocki, Szymon Sidor, Ilya Sutskever, and Igor Mordatch. 2018. Emergent complexity via multi-agent competition. In *International Conference on Learning Representations*.

Trapit Bansal, Pat Verga, Neha Choudhary, and Andrew McCallum. 2020c. Simultaneously linking entities and extracting relations from biomedical text without mention-level supervision. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7407–7414.

Yujia Bao, Menghua Wu, Shiyu Chang, and Regina Barzilay. 2020. Few-shot text classification with distributional signatures. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Shawn Beaulieu, Lapo Frati, Thomas Miconi, Joel Lehman, Kenneth O Stanley, Jeff Clune, and Nick Cheney. 2020. Learning to continually learn. In *ECAI 2020*, pages 992–1001. IOS Press.

Giannis Bekoulis, Johannes Deleu, Thomas Demeester, and Chris Develder. 2018. Joint entity recognition and relation extraction as a multi-head selection problem. *Expert Systems with Applications*, 114:34–45.

Emily M Bender and Alexander Koller. 2020. Climbing towards nlu: On meaning, form, and understanding in the age of data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5185–5198.

Samy Bengio, Yoshua Bengio, Jocelyn Cloutier, and Jan Gecsei. 1992. On the optimization of a synaptic learning rule. In *Preprints Conf. Optimality in Artificial and Biological Neural Networks*, pages 6–8. Univ. of Texas.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *The journal of machine learning research*, 3:1137–1155.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48.

Yonatan Bisk, Ari Holtzman, Jesse Thomason, Jacob Andreas, Yoshua Bengio, Joyce Chai, Mirella Lapata, Angeliki Lazaridou, Jonathan May, Aleksandr Nisnevich, et al. 2020. Experience grounds language. *arXiv preprint arXiv:2004.10151*.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of the 45th annual meeting of the association of computational linguistics*, pages 440–447.

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100.

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

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Rich Caruana. 1996. Algorithms and applications for multitask learning. In *Proceedings of the Thirteenth International Conference on International Conference on Machine Learning*, pages 87–95.

Rich Caruana. 1997. Multitask learning. *Machine learning*, 28(1):41–75.

Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. 2009. Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, 20(3):542–542.

Junkun Chen, Xipeng Qiu, Pengfei Liu, and Xuanjing Huang. 2018. Meta multi-task learning for sequence modeling. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

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.

Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. 2019. Electra: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations*.

Liam Collins, Aryan Mokhtari, and Sanjay Shakkottai. 2020. Why does maml outperform erm? an optimization perspective. *arXiv preprint arXiv:2010.14672*.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(ARTICLE):2493–2537.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The pascal recognising textual entailment challenge. In *Machine Learning Challenges Workshop*, pages 177–190. Springer.

Giulia Denevi, Carlo Ciliberto, Riccardo Grazzi, and Massimiliano Pontil. 2019. Learning-to-learn stochastic gradient descent with biased regularization. In *International Conference on Machine Learning*, pages 1566–1575. PMLR.

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.

William B Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

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.

Jingfei Du, Edouard Grave, Beliz Gunel, Vishrav Chaudhary, Onur Celebi, Michael Auli, Ves Stoyanov, and Alexis Conneau. 2020a. Self-training improves pre-training for natural language understanding. *arXiv preprint arXiv:2010.02194*.

Simon S Du, Wei Hu, Sham M Kakade, Jason D Lee, and Qi Lei. 2020b. Few-shot learning via learning the representation, provably. *arXiv preprint arXiv:2002.09434*.

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*.

Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. 2010. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11(Feb):625–660.

Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. 2020. On the convergence theory of gradient-based model-agnostic meta-learning algorithms. In *International Conference on Artificial Intelligence and Statistics*, pages 1082–1092. PMLR.

Chelsea Finn. 2018. *Learning to Learn with Gradients*. Ph.D. thesis, UC Berkeley.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, pages 1126–1135.

Chelsea Finn and Sergey Levine. 2018. Meta-learning and universality: Deep representations and gradient descent can approximate any learning algorithm. In *International Conference on Learning Representations*.

Chelsea Finn, Kelvin Xu, and Sergey Levine. 2018. Probabilistic model-agnostic meta-learning. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 9537–9548.

Sebastian Flennerhag, Andrei A Rusu, Razvan Pascanu, Hujun Yin, and Raia Hadsell. 2019. Meta-learning with warped gradient descent. *arXiv preprint arXiv:1909.00025*.

Leo Gao. 2021. On the sizes of OpenAI API models. `https://blog.eleuther.ai/gpt3-model-sizes/`.

Tianyu Gao, Xu Han, Zhiyuan Liu, and Maosong Sun. 2019a. Hybrid attention-based prototypical networks for noisy few-shot relation classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6407–6414.

Tianyu Gao, Xu Han, Hao Zhu, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. 2019b. FewRel 2.0: Towards more challenging few-shot relation 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 6251–6256, Hong Kong, China. Association for Computational Linguistics.

Ruiying Geng, Binhua Li, Yongbin Li, Xiaodan Zhu, Ping Jian, and Jian Sun. 2019. Induction networks for few-shot text classification.

Danilo Giampiccolo, Hoa Trang Dang, Bernardo Magnini, Ido Dagan, Elena Cabrio, and Bill Dolan. 2008. The fourth pascal recognizing textual entailment challenge. In *TAC*. Citeseer.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2007. The third pascal recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL workshop on textual entailment and paraphrasing*, pages 1–9.

Andrew B Goldberg and Xiaojin Zhu. 2006. Seeing stars when there aren't many stars: Graph-based semi-supervised learning for sentiment categorization. In *Proceedings of TextGraphs: The first workshop on graph based methods for natural language processing*, pages 45–52.

Erin Grant, Chelsea Finn, Sergey Levine, Trevor Darrell, and Thomas Griffiths. 2018. Recasting gradient-based meta-learning as hierarchical bayes. In *International Conference on Learning Representations*.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Alex Graves, Marc G Bellemare, Jacob Menick, Remi Munos, and Koray Kavukcuoglu. 2017. Automated curriculum learning for neural networks. In *International Conference on Machine Learning*, pages 1311–1320. PMLR.

Nathan Greenberg, Trapit Bansal, Patrick Verga, and Andrew McCallum. 2018. Marginal likelihood training of BiLSTM-CRF for biomedical named entity recognition from disjoint label sets. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2824–2829.

Jiatao Gu, Yong Wang, Yun Chen, Kyunghyun Cho, and Victor OK Li. 2018. Meta-learning for low-resource neural machine translation. *arXiv preprint arXiv:1808.08437*.

Jiang Guo, Darsh Shah, and Regina Barzilay. 2018. Multi-source domain adaptation with mixture of experts. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4694–4703.

Abhishek Gupta, Benjamin Eysenbach, Chelsea Finn, and Sergey Levine. 2018. Unsupervised meta-learning for reinforcement learning. *arXiv preprint arXiv:1806.04640*.

David Ha, Andrew Dai, and Quoc V Le. 2016. Hypernetworks. *arXiv preprint arXiv:1609.09106*.

R Bar Haim, Ido Dagan, Bill Dolan, Lisa Ferro, Danilo Giampiccolo, Bernardo Magnini, and Idan Szpektor. 2006. The second pascal recognising textual entailment challenge. In *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*.

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.

Dan Hendrycks and Kevin Gimpel. 2016. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *arXiv preprint arXiv:1606.08415*.

R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. 2018. Learning deep representations by mutual information estimation and maximization. In *International Conference on Learning Representations*.

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

Nithin Holla, Pushkar Mishra, Helen Yannakoudakis, and Ekaterina Shutova. 2020. Learning to learn to disambiguate: Meta-learning for few-shot word sense disambiguation. *arXiv preprint arXiv:2004.14355*.

Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. 2020. Meta-learning in neural networks: A survey. *arXiv preprint arXiv:2004.05439*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339.

Kyle Hsu, Sergey Levine, and Chelsea Finn. 2019. Unsupervised learning via meta-learning. In *International Conference on Learning Representations*.

Gabriel Huang, Hugo Larochelle, and Simon Lacoste-Julien. 2019. Are few-shot learning benchmarks too simple? *arXiv preprint arXiv:1902.08605*.

Po-Sen Huang, Chenglong Wang, Rishabh Singh, Wen-tau Yih, and Xiaodong He. 2018. Natural language to structured query generation via meta-learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 732–738.

Mike Huisman, Jan N van Rijn, and Aske Plaat. 2020. A survey of deep meta-learning. *arXiv preprint arXiv:2010.03522*.

Allan Jabri, Kyle Hsu, Abhishek Gupta, Ben Eysenbach, Sergey Levine, and Chelsea Finn. 2019. Unsupervised curricula for visual meta-reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 32, pages 10519–10531. Curran Associates, Inc.

Khurram Javed and Martha White. 2019. Meta-learning representations for continual learning. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pages 1820–1830.

Longlong Jing and Yingli Tian. 2020. Self-supervised visual feature learning with deep neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Armand Joulin, Édouard Grave, Piotr Bojanowski, and Tomáš Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 655–665.

Katharina Kann, Samuel R Bowman, and Kyunghyun Cho. 2020. Learning to learn morphological inflection for resource-poor languages. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8058–8065.

Katharina Kann, Kyunghyun Cho, and Samuel R Bowman. 2019. Towards realistic practices in low-resource natural language processing: The development set. *arXiv preprint arXiv:1909.01522*.

Arzoo Katiyar and Claire Cardie. 2017. Going out on a limb: Joint extraction of entity mentions and relations without dependency trees. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 917–928.

Siavash Khodadadeh, Ladislau Boloni, and Mubarak Shah. 2019. Unsupervised meta-learning for few-shot image classification. *Advances in neural information processing systems*, 32.

Mikhail Khodak, Maria-Florina Balcan, and Ameet Talwalkar. 2019. Provable guarantees for gradient-based meta-learning. In *International Conference on Machine Learning*, pages 424–433. PMLR.

Tushar Khot, Ashish Sabharwal, and Peter Clark. 2018. Scitail: A textual entailment dataset from science question answering. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

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

Vaishnavi Kommaraju, Karthick Gunasekaran, Kun Li, Trapit Bansal, Andrew McCallum, Ivana Williams, and Ana-Maria Istrate. 2020. Unsupervised pre-training for biomedical question answering. In *Working Notes of CLEF 2020 - Conference and Labs of the Evaluation Forum*, volume 2696 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*.

Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. 2019. Meta-learning with differentiable convex optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10657–10665.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880.

Ke Li and Jitendra Malik. 2016. Learning to optimize. *arXiv preprint arXiv:1606.01885*.

Zhenguo Li, Fengwei Zhou, Fei Chen, and Hang Li. 2017. Meta-sgd: Learning to learn quickly for few-shot learning. *arXiv preprint arXiv:1707.09835*.

Jingjing Liu, Panupong Pasupat, Scott Cyphers, and Jim Glass. 2013. Asgard: A portable architecture for multilingual dialogue systems. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8386–8390. IEEE.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019a. Multi-task deep neural networks for natural language understanding. *arXiv preprint arXiv:1901.11504*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Lajanugen Logeswaran and Honglak Lee. 2018. An efficient framework for learning sentence representations. In *International Conference on Learning Representations*.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2021. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. *arXiv preprint arXiv:2104.08786*.

Andrea Madotto, Zhaojiang Lin, Chien-Sheng Wu, and Pascale Fung. 2019. Personalizing dialogue agents via meta-learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5454–5459.

Gideon S Mann and Andrew McCallum. 2010. Generalized expectation criteria for semi-supervised learning with weakly labeled data. *Journal of machine learning research*, 11(2).

Andreas Maurer, Massimiliano Pontil, and Bernardino Romera-Paredes. 2016. The benefit of multitask representation learning. *Journal of Machine Learning Research*, 17(81):1–32.

Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. 2018. The natural language decathlon: Multitask learning as question answering. *arXiv preprint arXiv:1806.08730*.

R Thomas McCoy, Erin Grant, Paul Smolensky, Thomas L Griffiths, and Tal Linzen. 2020. Universal linguistic inductive biases via meta-learning. *arXiv preprint arXiv:2006.16324*.

Luke Metz, Niru Maheswaranathan, Brian Cheung, and Jascha Sohl-Dickstein. 2019. Learning unsupervised learning rules. In *International Conference on Learning Representations*.

Fei Mi, Minlie Huang, Jiyong Zhang, and Boi Faltings. 2019. Meta-learning for low-resource natural language generation in task-oriented dialogue systems. *arXiv preprint arXiv:1905.05644*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems-Volume 2*, pages 3111–3119.

Tomáš Mikolov, Ilya Sutskever, Anoop Deoras, Hai-Son Le, and Stefan Kombrink. 2012. Subword language modeling with neural networks.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1003–1011.

Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. 2018. A simple neural attentive meta-learner. In *International Conference on Learning Representations*.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using lstms on sequences and tree structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1105–1116.

Tsendsuren Munkhdalai and Hong Yu. 2017. Meta networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2554–2563. JMLR. org.

Shikhar Murty, Tatsunori Hashimoto, and Christopher D Manning. 2021. Dreca: A general task augmentation strategy for few-shot natural language inference. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1113–1125.

Alex Nichol and John Schulman. 2018. Reptile: a scalable metalearning algorithm. *arXiv preprint arXiv:1803.02999*, 2.

Farhad Nooralahzadeh, Giannis Bekoulis, Johannes Bjerva, and Isabelle Augenstein. 2020. Zero-shot cross-lingual transfer with meta learning. *arXiv preprint arXiv:2003.02739*.

Abiola Obamuyide and Andreas Vlachos. 2019. 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.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Sinno Jialin Pan and Qiang Yang. 2009. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of NAACL-HLT*, pages 2227–2237.

Jason Phang, Thibault Févry, and Samuel R Bowman. 2018. Sentence encoders on stilts: Supplementary training on intermediate labeled-data tasks. *arXiv preprint arXiv:1811.01088.*

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8).

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683.*

Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. 2017. Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. In *Advances in Neural Information Processing Systems*, pages 6076–6085.

Aravind Rajeswaran, Chelsea Finn, Sham Kakade, and Sergey Levine. 2019. Meta-learning with implicit gradients. *Advances in neural information processing systems.*

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250.*

Alex Ratner, Stephen Bach, Paroma Varma, and Chris Ré. 2017a. Weak supervision: The new programming paradigm for machine learning [blog post]. *https://dawn.cs.stanford.edu/2017/07/16/weak-supervision/.*

Alexander Ratner, Stephen H Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. 2017b. Snorkel: Rapid training data creation with weak supervision. In *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases*, volume 11, page 269. NIH Public Access.

Alexander Ratner, Christopher De Sa, Sen Wu, Daniel Selsam, and Christopher Ré. 2016. Data programming: Creating large training sets, quickly. *Advances in neural information processing systems*, 29:3567.

Sachin Ravi and Hugo Larochelle. 2017. Optimization as a model for few-shot learning. In *Proceedings of the International Conference on Learning Representations.*

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 148–163. Springer.

Sebastian Ruder. 2019. *Neural Transfer Learning for Natural Language Processing.* Ph.D. thesis, NATIONAL UNIVERSITY OF IRELAND, GALWAY.

Andrei A Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. 2018. Meta-learning with latent embedding optimization. *arXiv preprint arXiv:1807.05960.*

Andrei A. Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. 2019. Meta-learning with latent embedding optimization. In *International Conference on Learning Representations.*

Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. 2017. Dynamic routing between capsules. In *Advances in neural information processing systems*, pages 3856–3866.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. 2016. Meta-learning with memory-augmented neural networks. In *International conference on machine learning*, pages 1842–1850.

Victor Garcia Satorras and Joan Bruna Estrach. 2018. Few-shot learning with graph neural networks. In *International Conference on Learning Representations.*

Nikunj Saunshi, Sadhika Malladi, and Sanjeev Arora. 2020. A mathematical exploration of why language models help solve downstream tasks. *arXiv preprint arXiv:2010.03648.*

Timo Schick and Hinrich Schütze. 2020. It's not just size that matters: Small language models are also few-shot learners. *arXiv preprint arXiv:2009.07118.*

Jürgen Schmidhuber. 1987. *Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook.* Ph.D. thesis, Technische Universität München.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823.

Mike Schuster and Kaisuke Nakajima. 2012. Japanese and korean voice search. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5149–5152. IEEE.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725.

Amr Sharaf, Hany Hassan, and Hal Daumé III. 2020. Meta-learning for few-shot NMT adaptation. In *Proceedings of the Fourth Workshop on Neural Generation and Translation*, pages 43–53, Online. Association for Computational Linguistics.

Pranav Shyam, Shubham Gupta, and Ambedkar Dukkipati. 2017. Attentive recurrent comparators. In *International conference on machine learning*, pages 3173–3181. PMLR.

Suzanna Sia, Ayush Dalmia, and Sabrina J Mielke. 2020. Tired of topic models? clusters of pretrained word embeddings make for fast and good topics too! In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1728–1736.

Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pages 4077–4087.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Jost Tobias Springenberg. 2015. Unsupervised and semi-supervised learning with categorical generative adversarial networks. *arXiv preprint arXiv:1511.06390*.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in nlp. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650.

Jong-Chyi Su, Subhransu Maji, and Bharath Hariharan. 2020. When does self-supervision improve few-shot learning? In *European Conference on Computer Vision*, pages 645–666. Springer.

Amarnag Subramanya, Slav Petrov, and Fernando Pereira. 2010. Efficient graph-based semi-supervised learning of structured tagging models. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 167–176.

Yumin Suh, Bohyung Han, Wonsik Kim, and Kyoung Mu Lee. 2019. Stochastic class-based hard example mining for deep metric learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7251–7259.

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

Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip 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.

Ilya Sutskever, James Martens, and Geoffrey Hinton. 2011. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, pages 1017–1024.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. *Advances in Neural Information Processing Systems*, 27:3104–3112.

Wilson L Taylor. 1953. "cloze procedure": A new tool for measuring readability. *Journalism quarterly*, 30(4):415–433.

Dung Thai, Raghuveer Thirukovalluru, Trapit Bansal, and Andrew McCallum. 2021. Simultaneously self-attending to text and entities for knowledge-informed text representations. In *Proceedings of the 6th Workshop on Representation Learning for NLP (RepL4NLP-2021)*, pages 241–247, Online. Association for Computational Linguistics.

Sebastian Thrun and Lorien Pratt. 2012. *Learning to learn*. Springer Science & Business Media.

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

Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Well-read students learn better: The impact of student initialization on knowledge distillation. *ArXiv*, abs/1908.08962.

Jesper E Van Engelen and Holger H Hoos. 2020. A survey on semi-supervised learning. *Machine Learning*, 109(2):373–440.

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*, pages 5998–6008.

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

Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 2*, pages 2773–2781.

Tu Vu, Tong Wang, Tsendsuren Munkhdalai, Alessandro Sordoni, Adam Trischler, Andrew Mattarella-Micke, Subhransu Maji, and Mohit Iyyer. 2020. Exploring and predicting transferability across nlp tasks. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7882–7926.

Alex Waibel, Toshiyuki Hanazawa, Geoffrey Hinton, Kiyohiro Shikano, and Kevin J Lang. 1989. Phoneme recognition using time-delay neural networks. *IEEE transactions on acoustics, speech, and signal processing*, 37(3):328–339.

Alex Wang, Jan Hula, Patrick Xia, Raghavendra Pappagari, R Thomas McCoy, Roma Patel, Najoung Kim, Ian Tenney, Yinghui Huang, Katherin Yu, et al. 2018a. Can you tell me how to get past sesame street? sentence-level pretraining beyond language modeling. *arXiv preprint arXiv:1812.10860*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018b. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Haoxiang Wang, Ruoyu Sun, and Bo Li. 2020a. Global convergence and induced kernels of gradient-based meta-learning with neural nets. *arXiv preprint arXiv:2006.14606*.

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*.

Lingxiao Wang, Qi Cai, Zhuoran Yang, and Zhaoran Wang. 2020b. On the global optimality of model-agnostic meta-learning. In *International Conference on Machine Learning*, pages 9837–9846. PMLR.

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. 2019. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641.

Lilian Weng. 2019. Self-supervised representation learning. *lilianweng.github.io/lil-log*.

Guillaume Wenzek, Marie-Anne Lachaux, Alexis Conneau, Vishrav Chaudhary, Francisco Guzmán, Armand Joulin, and Édouard Grave. 2020. Ccnet: Extracting high quality monolingual datasets from web crawl data. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 4003–4012.

Adina Williams, Nikita Nangia, and Samuel R Bowman. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *33rd annual meeting of the association for computational linguistics*, pages 189–196.

Mingzhang Yin, George Tucker, Mingyuan Zhou, Sergey Levine, and Chelsea Finn. 2020a. Meta-learning without memorization. In *International Conference on Learning Representations*.

Wenpeng Yin, Nazneen Fatema Rajani, Dragomir Radev, Richard Socher, and Caiming Xiong. 2020b. Universal natural language processing with limited annotations: Try few-shot textual entailment as a start. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8229–8239.

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

Jaesik Yoon, Taesup Kim, Ousmane Dia, Sungwoong Kim, Yoshua Bengio, and Sungjin Ahn. 2018. Bayesian model-agnostic meta-learning. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 7343–7353.

A Steven Younger, Sepp Hochreiter, and Peter R Conwell. 2001. Meta-learning with backpropagation. In *IJCNN'01. International Joint Conference on Neural Networks. Proceedings (Cat. No. 01CH37222)*, volume 3. IEEE.

Fisher Yu and Vladlen Koltun. 2015. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*.

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. *arXiv preprint arXiv:1805.07513*.

Xiaohua Zhai, Avital Oliver, Alexander Kolesnikov, and Lucas Beyer. 2019. S4l: Self-supervised semi-supervised learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1476–1485.

Xiaojin Zhu, Zoubin Ghahramani, and John D Lafferty. 2003. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International conference on Machine learning (ICML-03)*, pages 912–919.

Luisa M Zintgraf, Kyriacos Shiarlis, Vitaly Kurin, Katja Hofmann, and Shimon Whiteson. 2019. Cavia: Fast context adaptation via meta-learning. In *International Conference on Machine Learning*.