BE MORE WITH LESS: SCALING DEEP-LEARNING WITH MINIMAL SUPERVISION

by

Yaqing Wang

A Dissertation

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Doctor of Philosophy

School of Electrical and Computer Engineering

West Lafayette, Indiana

May 2022

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Jing Gao, Chair

School of Electrical and Computer Engineering

Dr. Lu Su

School of Electrical and Computer Engineering

Dr. Qiang Qiu

School of Electrical and Computer Engineering

Dr. Xiaoqian Wang

School of Electrical and Computer Engineering

Approved by:

Dr. Dimitrios Peroulis

This thesis is dedicated to

my parents

Mr. Gang Wang and Mrs. Shuxia Zhang.

Without their love and support,

this work would not have been possible.

ACKNOWLEDGMENTS

First and foremost, I am deeply grateful to my advisor Professor Jing Gao, for her support and guidance on my research. Beyond research, I also learned varieties of experience including giving presentations, writing proposals and how to mentor students from her. Without her guidance, patience, encouragement, and immense knowledge, I could not have gone this far. Thank you to my committee members Professor Lu Su, Professor Qiang Qiu and Professor Xiaoqian Wang for their support, guidance, and fruitful conversations.

I am very happy to have had the opportunity to collaborate with an amazing set of faculty and researchers throughout my PhD. Im grateful to have had the opportunity to do internships at Microsoft Research Redmond Lab under Subhabrata Mukherjee, Ahmed Hassan Awadallah, Xiaodong Liu, Jianfeng Gao, Yuancheng Tu and Ming Wu and collaborating with Haoda Chu. I was fortunate enough to have tremendous freedom from Xin Luna Dong, Yifan Ethan Xu and Xian Li who mentor me during internships in Amazon. I would like to thank Professor Aidong Zhang, I learned how to work on fundamental research problems and develop new approaches for those problems from her.

I also sincerely express my gratitude to my amazing collaborators and friends. My time during PhD would not have been the same without them. Special thanks to Fenglong Ma, Kishlay Jha, Haoyu Wang, Ye Yuan, Hanbo Li, Ruya Kang, Houping Xiao, Chuishi Meng, Tianqi Wang, Rui Li, Haoda Chu, Zhengyang Wang, Hengtong Zhang, Qi Li, Yaliang Li, Qiuling Suo, Liuyi Yao, Hongfei Xue, Mengdi Huai, Guangxu Xun, Tianci Liu, Vishrawas Gopalakrishnan, Chenglin Miao, Wenjun Jiang, and those I have not included their names here, for their collaborations, their support, their helpful discussion and their generous sharing.

Finally, this thesis is dedicated to my parents, Gang Wang and Shuxia Zhang, for their love and accompany, and all the support and encouragement.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Large-scale deep learning models have reached previously unattainable performance for various tasks. However, the ever-growing resource consumption of neural networks generates large carbon footprint, brings difficulty for academics to engage in research and stops emerging economies from enjoying growing Artificial Intelligence (AI) benefits. To further scale AI to bring more benefits, two major challenges need to be solved. Firstly, even though large-scale deep learning models achieved remarkable success, their performance is still not satisfactory when fine-tuning with only a handful of examples, thereby hindering widespread adoption in real-world applications where a large scale of labeled data is difficult to obtain. Secondly, current machine learning models are still mainly designed for tasks in closed environments where testing datasets are highly similar to training datasets. When the deployed datasets have distribution shift relative to collected training data, we generally observe degraded performance of developed models. How to build adaptable models becomes another critical challenge. To address those challenges, in this dissertation, we focus on two topics: few-shot learning and domain adaptation, where few-shot learning aims to learn tasks with limited labeled data and domain adaption address the discrepancy between training data and testing data. In Part I, we show our few-shot learning studies. The proposed few-shot solutions are built upon large-scale language models with evolutionary explorations from improving supervision signals, incorporating unlabeled data and improving few-shot learning abilities with lightweight fine-tuning design to reduce deployment costs. In Part II, domain adaptation studies are introduced. We develop a progressive series of domain adaption approaches to transfer knowledge across domains efficiently to handle distribution shifts, including capturing common patterns across domains, adaptation with weak supervision and adaption to thousands of domains with limited labeled data and unlabeled data.

1. OVERVIEW

Large-scale deep learning models [1]–[3] have become the standard starting point and reaches previously unattainable performance for various tasks. Despite the remarkable success, these large models suffer from ever-growing resource consumption on data annotation, training and deployment. Those challenges largely prevent Artificial Intelligence (AI) solutions from benefiting downstream tasks and limiting their impacts in real-world settings. How to further scale deep learning to more broad domains and tasks becomes a challenging and important problem. To solve this problem, we get the inspiration from human learning abilities. Humans are able to grasp new concepts from only a few examples based on accumulated knowledge, and is able to adapt to unforeseen circumstances efficiently. Correspondingly, we study two topics in this dissertation: *few-shot learning* and *domain adaptation*. We devote our efforts on these two topics towards the goal of developing models with human learning abilities for AI scaling purpose.

Few-shot Learning. Few-shot learning is the problem of learning a new task with a small number of annotated examples and has been gaining more attentions with advances in large-scale pre-trained language models. Deep neural networks typically require large amounts of labeled training data to achieve state-of-the-art performance. Recent advances with pre-trained language models like BERT [1], GPT-2 [3] and RoBERTa [2] have reduced this annotation bottleneck. In this paradigm, deep and large neural network models are trained on massive amounts of unlabeled data in a self-supervised manner. However, the success of these large-scale models still relies on fine-tuning them on large amounts of labeled data for downstream tasks This poses several challenges for many real-world tasks. Not only is acquiring large amounts of labeled data for every task expensive and time consuming, but also not feasible in many cases due to data access and privacy constraints, especially when dealing with personal or sensitive data. To address those issues, we propose to develop few-shot learning approaches for large-scale models. My research tackled these challenges with evolutionary explorations from improving supervision signals, incorporating unlabeled data and improving few-shot learning abilities with lightweight tuning design to enable easy deployment. More details are discussed as follows:

- ***Learning from Informative Supervision.*** Sequence labeling task aims at identifying and categorizing spans of text into a pre-defined set of classes. Such a fundamental task is widely

adopted in information extraction, question answering and other language understanding applications. Conventional sequence labeling models usually treat each class as a one-hot vector (represented by a class label), which does not carry semantic information of entity classes and cannot form effective supervision for model training in low resource scenarios. Meanwhile, the trained model could be highly associated with known classes and is difficult to transfer learned knowledge to novel entity classes. To address this challenge, we propose a method `SpanNER` [4], which is the first work to decompose the sequence labeling task into span detection and entity class inference and learn from natural language supervision. Such a design provides a flexible and precise way to capture the semantics of entity classes and brings substantial improvement in low-resource scenarios.

- ***Incorporating Unlabeled Data.*** While recent large-scale pre-trained language models (PLM) have obtained impressive few-shot performance, they still have a significant performance gap relative to fully supervised state-of-the-art (SoTA) models. In the work [5], we propose a meta self-training framework, namely `MetaST`, which leverages very few manually annotated labels and a large amount of unlabeled data for neural sequence labeling model training. While self-training serves as an effective mechanism to learn from large amounts of unlabeled data via iterative knowledge exchange – meta-learning helps in adaptive sample re-weighting to mitigate error propagation from noisy pseudo-labels.

- ***Lightweight Few-shot Learners.*** Pre-trained language models (PLM) have been steadily increasing in size in terms of trainable parameters ranging from millions to billions of parameters, increasing both the computational cost and the serving cost in terms of the storage, where every task requires its customized copy of the large model parameters. In this work [6], we present a new fine-tuning method `LiST` that improves *few-shot learning ability* and *parameter-efficiency* over existing fine-tuning strategies. `LiST` uses self-training and prompt fine-tuning to learn from large amounts of unlabeled data from target domains. In order to reduce the storage and training cost, `LiST` tunes only a small number of adapter parameters with few-shot labels while keeping the large encoder frozen. With only 30 labels for every task, `LiST` improves by upto 35% over classic fine-tuning while reducing 96% of the tunable parameters.

Domain Adaptation. Current machine learning models are only suitable for well-defined and narrow tasks in closed environments where deployed datasets are similar to training dataset. However, the world we see is ever-changing along changes with people, things, and the environment, resulting in distribution shift in real-world applications. In light of this challenge, we develop a progressive series of domain adaption approaches to transfer knowledge across domains efficiently to handle distribution shifts, including capturing common patterns across domains, efficient adaption for novel domains via weak supervision, quick adaption with few-labels and quick adaption with unlabeled data.

- ***Common Patterns across Domains.*** News varies across different events in vocabulary, writing style and involved persons and etc, bringing domain shifts and limiting the applicability of developed models. In the work [7], we first recognized the challenge of fake news detection on emergent events and proposed a fake news detection approach for novel and time-critical events. One typical paradigm to handle domain shift is to repeat the procedures of collecting data from novel domains, re-training the model and re-deploying trained models. Such a paradigm is not only computationally expensive but time consuming. To address those issues, we propose a novel fake news detection model, namely `EANN`, which uses event discriminator to measure the dissimilarities among different events, and further learns the event invariant features which can generalize well for the newly emerged events.

- ***Weak Supervision on Novel Domains.*** Our work [7] helps capture shared patterns over events in training data but cannot learn new patterns for novel events. In the work [8], we develop a framework, namely `WeFEND`, which can leverage users reports as weak supervision to learn new patterns from novel events for fake news detection and significantly reduce data annotation efforts and costs. Furthermore, a data selector based on reinforcement learning techniques is integrated to choose high-quality samples from the weakly labeled data and filter out those low-quality ones that may degrade the detectors performance.

- ***Quick Adaption with Few Labels.*** Adding the knowledge from newly emergent events requires to build a new model from scratch or continue to fine-tune the model on newly collected labeled data, which can be challenging and expensive for real-world settings. We

wonder whether the model is able to quickly adapt to new events without expensive retraining procedure. To overcome this challenge, we propose a quick adaption model design, namely `MetaFEND`, which is able to learn new knowledge within few labels. More specifically, as the writing style, content, vocabularies and even class distributions of news on different events usually tends to differ, `MetaFEND` learns to leverage labeled data instances as conditioning, addressing limitations of meta-learning in handling heterogeneous domain distributions.

Adaption to Millions of Domains with Unlabeled Data. The number of product types is towards millions and thus collecting training data for each of product types is extremely expensive. In the work [9], we propose a meta-learning latent variable approach, namely MetaBridge, for product attribute validation task. The proposed approach effectively leverages a small set of labeled data in limited product types for training and enables quick adaptation to more than thousands of types with unlabeled data.

The rest of this thesis is organized as the following. In Chapter 2, 3 and 4, few-shot learning methods with semantic supervision signals, unlabeled data and light-weight tuning are studied. In Chapter 5, 6, 7 and 8, domain adaptation with adversarial learning, weak supervision, few-labels and unlabeled data are presented. Related work is introduced in Chapter 9. Finally, in Chapter 10, the dissertation is concluded.

Part I

FEW-SHOT LEARNING

2. LEARNING FROM LANGUAGE SUPERVISION

(A version of this chapter has been previously published in EMNLP 2021 [4].)

2.1 Introduction

Named entity recognition (NER) aims at identifying and categorizing spans of text into a pre-defined set of classes, such as people, organizations, and locations. As a fundamental language understanding task, NER is widely adopted in question answering [10], information retrieval [11] and other language understanding applications [12]–[14]. Recent advances with pre-trained language models like BERT [1], GPT-2 [3] and RoBERTa [2] have shown remarkable success in NER . However, the success of these large-scale models still relies on fine-tuning them on large amounts of in-domain labeled data. Unfortunately, obtaining NER annotations not only is expensive and time consuming, but also may not be feasible in many sensitive user applications due to data access and privacy constraints. This motivates us to study the problem of low-shot NER.

Low-shot NER focuses on identifying custom entities in a new domain with only a few in-domain examples or even without any in-domain labeled data, which are refereed to as few-shot NER and zero-shot NER respectively. The success of low-shot NER requires the model to be capable of transferring learned knowledge to recognize new entity classes. Conventional NER models usually treat each class as a one-hot vector (represented by a class label) for training, and thus the trained model cannot capture the semantic meanings of those labels. In fact, the trained model could be highly associated with known classes and it is difficult to transfer learned knowledge to novel entity classes.

To tackle this problem, several recent works [15]–[19] employ prototype-based method to represent each class by a prototype based on the labeled examples and use nearest neighbor method for NER. However, each entity class in NER task may include several fine-grained entity classes and has diverse semantic meanings. Correspondingly, the tokens or entities belonging to the same entity class are not necessarily close to each other [15], making it challenging to represent each entity class by a prototype based on a few examples. For example, `MISC`, one of the four entity classes in the benchmark dataset CoNLL03 [20], is a collection of fine-grained entity classes including events, nationalities, products and works of art. `FAC` is an entity class in the OntoNotes5 [21] collection,

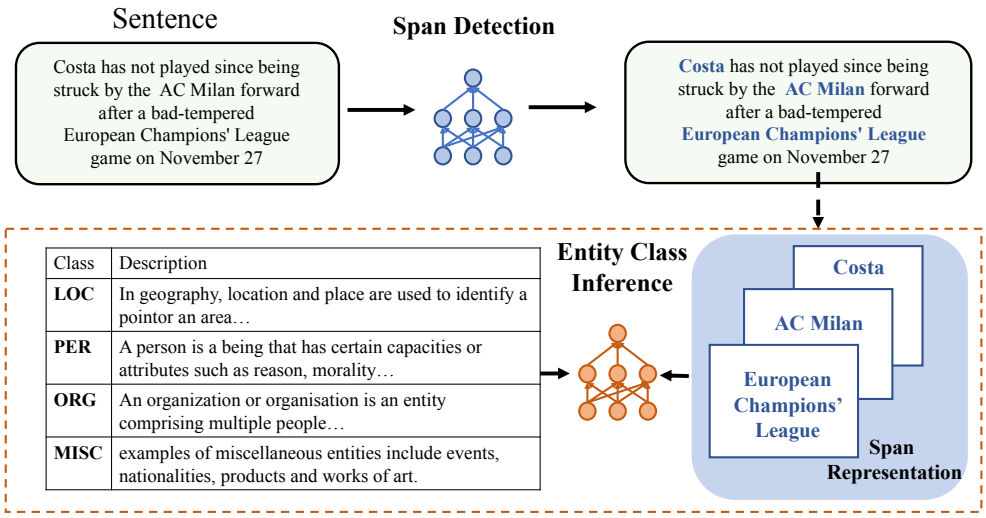


Figure 2.1. An overview of the proposed NER system: SpanNER.

including buildings, airports, highways, bridges and others. Thus, prototype-based methods may end up learning noisy representations of prototypes and cannot achieve satisfactory performance. Moreover, prototype-based methods have unavoidable reliance on labeled examples, thereby making them unable to extend to the zero-shot learning setting, which is also an important and practical scenario in the low-shot NER.

In this work, we propose a simple yet effective method SpanNER which can tackle few-shot as well as zero-shot NER. Instead of deriving the representations of labels purely from a few labeled examples, we propose to directly learn from the natural language descriptions of entity classes. Such a choice provides a flexible and precise way to obtain the semantic meanings of entity classes and enable zero-shot learning. Although using natural language as supervision has been explored in the context of zero-shot text classification, it is challenging to be adapted in the NER task. Unlike its use in text classification, natural language cannot provide direct supervision for token classification. Inspired by machine reading comprehension (MRC) framework, we propose to decompose the NER task into two procedures: span detection and entity class inference, which can be jointly trained together. However, it is challenging to employ MRC framework into NER task especially in the low-resource setting. The MRC framework usually needs a large amount of data for training, which is not available in the low resource scenario. To handle these challenges, we propose a class-agnostic span detection module which is equipped with token sampling mechanism

to mitigate the imbalanced class issue and can be trained with limited labeled data. Moreover, to handle the challenging that several fine-grained classes are included in one entity class, we develop an entity class attention module to focus on the most relevant information in the given entity class description that corresponds to the extracted span. Compared with direct adaption of MRC framework into NER [22], the proposed method can bring more than 54%, 30% and 26% improvement in average under few-shot learning, domain transfer and zero-shot learning settings respectively. Figure 8.1 shows an overview of the proposed framework: the span detection stage aims to identify the span of text, and entity class inference is responsible for categorizing extracted spans based on natural language description of pre-defined entity classes. We perform extensive experiments on 5 benchmark datasets and evaluate the proposed method in the few-shot learning, domain transfer and zero-shot learning settings. The experimental results show that the proposed method brings large improvements over the state-of-the-art methods across different settings.

Contributions. Our model design is simple but distinguishes from that of the other NER works. To the best of our knowledge, we are the first one to learn entity class via natural language for NER task and the proposed method SpanNER achieves around 10%, 23% and 26% improvement over state-of-the-art NER methods in few-shot learning, domain-transfer and zero-shot learning settings respectively.

2.2 Task Formulation

NER is the process of locating and classifying named entities in text into predefined entity categories, such as person names, organizations, and locations. Formally, given a sentence with N tokens $X = \{x_1, ..., x_N\}$, an entity or slot value draws from a span of tokens $s = [x_{\mathrm{i}}, ..., x_{\mathrm{j}}](0 \leq \mathrm{i} \leq \mathrm{j} \leq N)$ associated with an entity class $c \in \mathcal{C}$. The corresponding annotations for given sentence X are denoted as Y.

Few-shot NER focuses on the NER task in a low-resource setting, where a system is only provided with a few in-domain labeled examples per entity class and the system needs to learn to identify custom entities in the domain. The task of K-shot NER refers to the setting with K labeled input sentences per entity class $c \in \mathcal{C}$, and the training data can be denotes as $\mathcal{D}_{\text{train}} = \{X_{\mathrm{i}}, Y_{\mathrm{i}}\}_{\mathrm{i}=1}^{|\mathcal{C}|\times K}$. In this work, we leverage given training data $\mathcal{D}_{\text{train}}$ for model fine-tuning.

Zero-shot NER focuses on a more challenging setting where a model is trained with a set of entity classes and then tested on a dataset with a different set of entity classes. Towards this end, zero-shot NER systems need to learn to generalize to unseen entity classes without using any labeled example. The training data for zero-shot learning can be denoted as $\mathcal{D}_{\text{train}}$ associated with an entity class set $\mathcal{C}_{\text{train}}$, and the test data is denoted as $\mathcal{D}_{\text{test}}$ associated with an entity class set $\mathcal{C}_{\text{test}}$. Note that $\exists c_{\text{test}} \in \mathcal{C}_{\text{test}}$ but $c_{\text{test}} \notin \mathcal{C}_{\text{train}}$.

2.3 Methodology

In this work, we study how to develop an effective model which can identify custom entities in a novel domain with a small set of labeled data or even without using any labeled data. To this end, we decompose NER task into two sub-tasks: span detection and entity class inference. The span detection module is class-agnostic and can transfer knowledge across different entity classes. On top of span detection, the entity class inference module takes extracted spans as input to infer the semantic relationship between the spans with natural language description of entity classes. Learning from natural language has an important advantage over existing categorical label learning methods, which is its ability to capture semantic meanings of labels and enable flexible zero-shot transfer.

2.3.1 Span Detection

Span detection is explored in the machine reading comprehension (MRC) frameworks [23], [24], which predict the probability for each token as the starting or ending of the answer span given a question. However, it is challenging to directly adapt MRC framework for NER task especially in the low-resource setting. First, for each entity class, the model needs to answer its associated natural language question and repeat this procedure until all the questions are answered [22]. Thus, such a method is not scalable when the number of entity classes increases and further exacerbates the imbalanced class issue compared to conventional NER framework. Second, the MRC framework usually needs a large amount of data, which is not available in the low resource scenario. To handle these challenges, we propose a class-agnostic span detection module which can share the knowledge across classes and develop a token sampling mechanism to mitigate imbalanced issue.

The proposed span detection module takes input sequence as input without questions and can be trained with limited labeled data. Given an input sequence $X = \{x_1, ..., x_N\}$, we first feed X into a pre-trained BERT [1] to obtain token representations $\{\mathbf{x}_1, ..., \mathbf{x}_N\} \in \mathbb{R}^{h \times N}$. Besides start and end index predictions, we also classify whether a token is a part of the entity span. For example, we get the score for each token being start as follows:

$$s_{\text{start}}(\mathrm{i}) = \mathbf{w}_{\text{start}}^{\intercal} \cdot \mathbf{x}_{\mathrm{i}}, \tag{2.1}$$

where $\mathbf{w}_{\text{start}} \in \mathbb{R}^{h \times 1}$ is the weight of the linear classifier. Correspondingly, the probability of a token being start index is:

$$p_{\text{start}}(\mathrm{i}) = \text{sigmoid}(s_{\text{start}}(\mathrm{i})). \tag{2.2}$$

The probability calculations for end and a part of a span are the same as that of start index prediction. We then compute the probability of a span [i, j] being an entity as:

$$\begin{aligned} p_{\text{match}}([\mathrm{i}, \mathrm{j}]) \quad = \quad & \text{sigmoid}\Big(s_{\text{start}}(\mathrm{i}) + s_{\text{end}}(\mathrm{j}) \\ & + \sum_{t=\mathrm{i}}^{\mathrm{j}} s_{\text{span}}(t) \Big). \end{aligned}$$

The span detection loss consists of three parts: start prediction loss, end prediction loss and span matching loss. The loss function of start prediction can be represented as:

$$\mathcal{L}_{\text{start}} = \frac{1}{N} \sum_{\mathrm{i}=1}^{N} \text{CE}(p_{\text{start}}(\mathrm{i}), y_{\text{start}}^{\mathrm{i}}), \tag{2.3}$$

where CE represents cross-entropy function and $y_{\text{start}}^{\mathrm{i}} = 1$ if token x_{i} is the start of an entity. The loss of end prediction can be calculated in a similar way.

Mitigation of imbalanced class issue. For an input sequence with length N, the number of span candidate is in a $N \times N$ scale, where most of them are negative span candidates. To mitigate the imbalanced class issue, we sample a subset of negative span candidates, denoted as O^{neg}. The span

candidate set which corresponds to gold spans is denoted as O^{pos}. Instead of using all the negative spans, we propose to sample a subset of negative spans to mitigate imbalance class issue. More specifically, we set the sampling size of negative span candidates as $|O^{\text{neg}}| = N - |O^{\text{pos}}|$ to reach a class ratio between positive and negative labels similar to the one in the start and end prediction losses. The span match loss is:

$$\begin{aligned}\mathcal{L}_{\text{match}} &= -\frac{1}{N}\Big(\sum_{\text{(i,j)}\in O^{\text{pos}}} \log p_{\text{match}}([\text{i,j}]) \\ &\quad + \sum_{\text{(i,j)}\in O^{\text{neg}}} \log\left(1 - p_{\text{match}}([\text{i,j}])\right)\Big).\end{aligned}$$

The overall span objective consisting of three losses to be minimized is as follows:

$$\mathcal{L}_{\text{span}} = \mathcal{L}_{\text{start}} + \mathcal{L}_{\text{end}} + \mathcal{L}_{\text{match}}. \tag{2.4}$$

During inference, start and end indexes are first separately predicted. Then we select the consensus span between match predictions and extracted (start, end) indexes to achieve final predictions.

2.3.2 Natural Language Supervision

Learning based on categorical labels only may discard the semantic meanings of labels, and thus it is difficult to transfer knowledge from known classes to new entity classes. To mitigate this limitation, we propose to use natural language description[1] of entity classes to provide supervision for entity class inference and enable zero-shot learning. However, different with zero-shot text classification or entity linking, the entity class description in NER may describe several fine-grained entity classes, making our setup more challenging.

Mention Representation. Upon span detection, we can first obtain the mention span representation of each span candidate [i, j] by averaging the embeddings of the span tokens. However, entity class

[1] ↑In this work, we use the definitions of entity classes from Wikipedia or annotation guidelines as their language description.

is usually a high level category including many entities. Thus, we need to add a linear transformation $\mathbf{w}_{\text{entity}} \in \mathbb{R}^{h \times h}$ to project average span token embedding into the entity class space:

$$\mathbf{e}_{\text{i,j}} = \mathbf{w}_{\text{entity}} \cdot (\frac{1}{\text{j} - \text{i} + 1} \sum_{t=\text{i}}^{\text{j}} \mathbf{x}_t). \tag{2.5}$$

The description of an entity class c is a sequence of tokens, denoted as $X^c = \{x_1^c, ..., x_K^c\}$. In this work, we feed entity class description into another pre-trained BERT [1] to obtain its representations $\{\mathbf{x}_1^c, ..., \mathbf{x}_K^c\} \in \mathbb{R}^{h \times K}$. Since there is limited data or even no data for training in the novel domain, we fixed the parameters of this BERT to expedite transferring by maintaining the embedding of entity class description from source and novel domains in the same space.

Entity Class Description Attention. The entity class description may describe several fine-grained entity classes. To focus on the information in the description that corresponds to the extracted span, we propose to construct adaptive entity class representation. More specifically, we use multi-headed attention mechanism [25]. Each single attention function can be described as mapping a query and a set of key-value pairs to an output. The query, key and value vector are denoted as $\mathbf{Q}$, $\mathbf{K}$ and $\mathbf{V}$ respectively. We use the aggregated mention vector $\mathbf{e}_{\text{i,j}} \in R^{h \times 1}$ as query vector $\mathbf{Q}$ and use entity class description embedding $\mathbf{X}_c = [\mathbf{x}_1^c, ..., \mathbf{x}_K^c] \in \mathbb{R}^{h \times K}$ as key vector $\mathbf{K}$ and value vector $\mathbf{V}$. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by the dot-product function of the query with the corresponding key. Then multiple parallel attention heads can stabilize the learning mechanism. We represent the procedure of obtaining adaptive entity class representation $\mathbf{x}^c(\mathbf{e}_{\text{i,j}}) \in \mathbb{R}^{h \times 1}$ as:

$$\mathbf{x}^c(\mathbf{e}_{\text{i,j}}) = \text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}). \tag{2.6}$$

The role of such a mechanism is empirically justified by the comparison between the proposed model and a reduced model (i.e., the proposed model without attention mechanism) in the experimental section.

Entity Class Inference. The entity class inference is to infer the relationship between entity class and extracted span. We follow the zero-shot text classification [26] to cast this task into binary

prediction: whether the extracted span belongs to given entity class or not. The probability of the extracted span being in a given entity class c is based on a matching score between them:

$$p(c|\mathrm{e_{i,j}}) = \mathrm{sigmoid}(\mathbf{e}_{\mathrm{ij}}^{\mathrm{T}}\mathbf{x}^{c}(\mathrm{e_{i,j}})). \tag{2.7}$$

The loss for each extracted span $[\mathrm{i}, \mathrm{j}]$ is calculated as:

$$\mathcal{L}_{\mathrm{entity}}([\mathrm{i}, \mathrm{j}]) = \frac{1}{|\mathcal{C}|}\sum_{c\in\mathcal{C}} \mathrm{CE}\Big(p(c|\mathrm{e_{i,j}}), y\Big), \tag{2.8}$$

where $\mathcal{C}$ is a set of entity classes of interest, and y is binary label which equals to 1 when extracted span belongs to entity class c and 0 otherwise. We use $\mathcal{L}_{\mathrm{entity}}$ to denote the entity class inference loss for all extracted spans.

Final Loss. We jointly train span detection and entity class inference modules by optimizing the sum of their losses.

2.4 Experiments

In this section, we empirically study and compare the proposed method with state-of-the-art methods in few-shot learning, domain transfer and zero-shot learning settings.

2.4.1 Experimental Setup

Dataset. We perform large-scale experiments with five different datasets[2] including Named Entity Recognition tasks and user utterances for task-oriented dialog systems as summarized in Table 3.1. (a) CoNLL03 [20] is a collection of news wire articles from the Reuters Corpus with 4 entity classes. (b) OntoNotes5 [21] is in general domain including 18 entity classes. (c) WNUT 2017 [27] is collected from social media with 6 entity classes. (d) MIT Movie and Restaurant corpus [28] consist of user utterances for movie and restaurant domains with 12 and 8 classes.

Backbone We use the the pre-trained $\mathrm{BERT}_{\mathrm{base}}$ uncased model ($\sim$110M parameters) as the backbone network. The inputs during training and inference are lowercased to make them case-insensitive.

[2] ↑https://github.com/juand-r/entity-recognition-datasets

Table 2.1. Dataset summary.

Dataset	Domain	# Classes	# Train	# Test
CoNLL03	News	4	14K	3.6K
OntoNotes5	General	18	60K	8.3K
WNUT	Social Media	6	3.4K	1.6K
Movie	Moive	12	8.8K	2.4K
Restaurant	Restaurant	8	6.9K	1.5K

2.4.2 Few-shot Learning

Setting. In this subsection, we study how the proposed method performs in a few-shot supervision setting, For 5-shot setting, we sample 5 sentences for each entity class from the training set and fine-tune models with sampled sentences. The experiment is repeated for 10 times to report the average F1 score.

Baselines The first baseline we use is a fully supervised BERT model trained on all available training data (3.4K-60K sentences) which provides the ceiling performance for every task. Each of the other models are trained on 5 training sentences per class. We compare our method with BERT (same backbone with ours) with Beginning-Intermediate-Outside (BIO) tagging mechanism as a comparison to evaluate the proposed model design besides the backbone choice. LC and Prototype are abbreviations for linear classifier and prototype-based methods from a recent few-shot NER work [15]. They use pre-trained model RoBERTa-base as their backbone model. MRC-NER [22] casts NER task into machine reading comprehension and achieves the state-of-the-art performance on several benchmark datasets. To study the role of attention mechanism proposed in subsection 2.3.2, we propose a reduced model SpanNER-NoAttn, which uses average operation instead of attention to aggregate entity class description.

Performance We report the results of 5-shot supervision and distantly supervising pre-training plus 5-shot supervision in Table 2.2. In the 5-shot supervision setting, we can observe that our methods outperform baseline BERT consistently, which shows the advantage of the proposed model design in addition to the benefits from backbone. The baseline Prototype leverages given support examples to conduct NER task and achieves lower performance compared with LC with the same backbone according to average F1. The reason may lie in that the tokens belonging to the same entity class are

Table 2.2. F1 score comparison of models on different datasets. All models (except LC and Prototype) use the same BERT backbone. † indicates results from [15]. The highest scores are **bolded**, while the second highest score is underlined. F1 score of our model for each task is followed by standard deviation and percentage improvement [↑] is over the best baseline. *RoBERTta is pre-trained on reddit dataset which is similar to WNUT. We change backbone from BERT to Roberta (same with Prototype's) and F1 of SpanNER on WNUT is **31.5** (0.3).

Method	CoNLL03	OntoNotes5	WNUT	Movie	Restaurant	Average
Full-supervision						
BERT	91.1	87.8	47.1	87.9	79.0	78.6
5-shot supervision						
BERT	61.6	60.1	21.2	61.9	48.6	50.7
LC†	53.5	57.7	25.7	51.3	48.7	47.4
Prototype†	58.5	53.3	**29.5**	38.0	44.1	44.7
MRC-NER	28.5	49.8	0.4	58.7	43.1	36.1
SpanNER-NoAttn (ours)	68.4 (0.5)	65.1 (0.3)	22.8 (0.4)	64.8 (0.3)	48.9 (0.2)	54.0
SpanNER (ours)	**71.1** (0.4)	**67.3** (0.5)	25.8 (0.3)*	**65.4** (0.4)	**49.1** (0.2)	**55.7** [↑9.9%]

not necessarily close to each other [15]. Prototype achieves better performance on WNUT compared to SpanNER since Prototype is based on Roberta which is pre-trained on social media dataset reddit. We change backbone of the proposed model SpanNER from BERT to Roberta-bas and observe that SpanNER achieves 31.5 in term of F1 score and outperforms Prototype. The MRC-NER framework is a reading comprehension framework whose success relies on training on large-scale data and thus cannot achieve satisfactory performance in a few-shot setting. Overall, we observe that our methods largely outperform all methods including the models with the same BERT encoder as ours across different datasets. The average performance improvement over the best baseline BERT is around 10%. Moreover, the comparison between SpanNER and SpanNER-NoAttn demonstrates that the improvement brought by attention mechanism is around 3.1%.

Varying the number of shots. Table 2.2 shows the improvement in the performance of SpanNER and BERT when increasing the number of labels for each NER type in the CoNLL03 dataset. As we increase the amount of labeled training instances, SpanNER improves over BERT consistently.

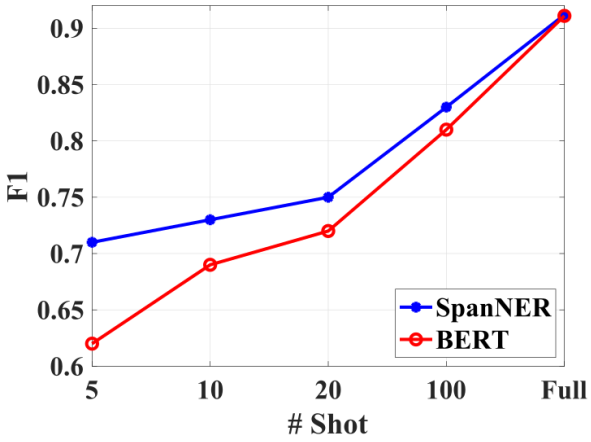


Figure 2.2. Variation in model performance on varying shots on CoNLL03. "Full" indicates full supervision.

2.4.3 Domain Transfer

We evaluate the proposed model in another common scenario of adapting a NER model to a novel domain [16]. In this setting, we have a fully supervised source domain and a target domain with few-shot supervision. Following [16], we use general domain (OntoNotes5) as a source domain and evaluate models on News (CoNLL) and Social (WNUT) domains.

Table 2.3. F1 score comparison of models on CoNLL03 and WNUT datasets with 5-shot supervision for domain transfer. † indicates results from [16].

Models	CoNLL03	WNUT	Average
SimBERT †	28.6	7.7	18.2
Prototypical Network †	65.9	19.8	42.9
PrototypicalNet+P&D †	67.1	23.8	45.4
NNShot †	74.3	23.9	49.1
StructShot †	75.2	27.2	51.2
MRC-NER	64.1	32.6	48.3
SpanNER-NoAttn (ours)	80.1 (0.4)	42.0 (0.7)	61.1
SpanNER (ours)	**83.1** (0.5)	**43.1** (0.6)	**63.1** [↑23.2%]

Baselines. We adopt six state-of-the-art methods in the domain transfer setting as baselines. SimBERT is based on a pre-trained BERT encoder and the predictions are conducted by a nearest neighbor classifier [16]. Prototypical Network [29] is a state-of-the-art few-shot classification

system and is adopted by [30] for few-shot NER task. Upon Prototypical Network, Prototypical-Net+P&D [17] adds pairwise embedding and dependency mechanism to gain further improvements. StructShot and NNShot are proposed in [16], which achieve state-of-the-art performance in this domain transfer setting. We include our reduced baseline SpanNER-NoAttn in this task as an ablation study and follow [16] to run our models five times and report average performance with standard deviation.

Performance Table 2.3 shows the results of baselines and the proposed methods on CoNLL03 and WNUT datasets. The proposed models achieve F1 scores 83.1 and 43.1 on CoNLL03 and WNUT respectively, bringing improvements over the best baseline 10.5% and 32.2% correspondingly. The proposed model effectively transfers learned knowledge to novel domains by learning from natural language descriptions instead of simple one-hot representation of entity classes.

2.4.4 Zero-shot NER

Setting The zero-shot learning setting is motivated by the fact that new types of entities often emerge in some domains and sometimes the annotations in the target domain are not accessible. Following zero-shot text classification setting [26], we evaluate the proposed model in a common setting: *label-partially-unseen*. In label-partially-unseen setting, a part of labels are unseen, enabling us to check the performance on unseen labels as well as seen labels.

Baselines Zero-shot NER task is rarely studied. The most state-of-the-art model for zero-shot NER is MRC-NER [22], which conducts NER task by extracting answer spans given the questions of entity classes. Another baseline we use is the reduced model SpanNER-NoAttn. The comparison with this reduced model can demonstrate the role of entity class attention mechanism.

Table 2.4. F1 score comparison of models on CoNLL03 and WNUT datasets. *Overall* and *Unseen* indicate F1 scores of all entity classes and never-seen entity classes, respectively.

Method	**CoNLL03**		**WNUT**	
	Overall	Unseen	Overall	Unseen
MRC-NER	39.1	14.5	24.0	7.4
SpanNER-NoAttn	39.0	0.5	31.4	16.8
SpanNER	**53.0**	**33.5**	**35.4**	**18.8**

Performance Table 2.4 shows the F1 scores of MRC-NER and the proposed methods. MRC-NER based on reading comprehension framework is capable of conducting NER task for never-seen classes. However, the span detection in MRC-NER is tightly coupled with question understanding, leading to more difficulty in handling unseen entity classes. In contrast, the proposed framework decomposes the NER task into span detection and entity class inference, avoiding the error propagation between two modules and thus delivering better performance. The comparison between SpanNER-NoAttn and SpanNER indicates the importance of attention mechanism in entity class description understanding, especially for never-seen entity classes.

Entity Class Inference We conduct experiments that disentangle entity class inference module from SpanNER so that the capability of this module can be evaluated. The span detection module cannot be separately evaluated because the span annotations on both datasets are associated with pre-defined entity classes. To demonstrate the capability of entity class inference, we use gold spans to evaluate the performance of entity class inference. Table 2.5 shows the performance of SpanNER-NoAttn and SpanNER. Comparing these two methods, we can observe that the attention mechanism helps improve the performance of the entity class inference.

Table 2.5. Experiments that demonstrate the performance of the entity class inference module and adopt annotating guidelines as entity class descriptions on CoNLL03 and WNUT datasets.

	CoNLL03	**WNUT**
Entity Class Inference		
SpanNER-NoAttn	56.2	53.7
SpanNER	60.2	57.0
Annotation guidelines		
SpanNER-NoAttn	31.8	11.5
SpanNER	42.1	15.7

Class Description Construction We set up experiments to study how the entity class description affects the model performance. In this experiment, we replace Wikipedia description of entity classes by annotation guidelines from CoNLL03 and WNUT datasets in the testing stage. We can observe that the proposed models are still capable of identifying entities belonging to never-seen entity classes even though the descriptions in the testing stage are different from those in the training

stage. The F1 scores drop compared to the scores when Wikipedia descriptions are used because training and test stages use different descriptions and the annotation guidelines do not include the semantic explanation of entity classes.

Performance per Entity Class We show F1 score per entity class on CoNLL03 and WNUT datasets in Table 2.6. We can observe the various degrees of recognizing different entity classes. First, the person names are easily recognized across different domains. The performance of person entity class on WNUT is worse compared to that on CoNLL03, which may be due to the large domain shift in social media data. It is interesting to see that the performance of seen entity classes `LOC`, `location` and `product` is even worse than that of never-seen entity classes. To explain this interesting phenomenon, we provide a detailed analysis of error cases below.

Table 2.6. F1 score of SpanNER per entity class on CoNLL03 and WNUT datasets. * indicates unseen entity classes.

CoNLL03		**WNUT**	
Entity Class	F1	Entity Class	F1
PER	77.4	person	59.1
ORG	58.5	creative-work*	19.3
MISC*	33.5	corporation*	19.1
LOC	5.9	group*	18.3
-	-	location	14.4
-	-	product	11.0

Error Analysis We manually examine the errors made by the proposed model on the CoNLL03 and WNUT test datasets and categorize these errors into 3 types. (1) Different annotation guidelines on datasets. For instance, the description of `location` entity class in the source domain (OntoNotes5) is limited to mountain ranges and bodies of water, excluding countries, cities, states (these are included in the entity class `GPE`). Such a description is different from entity class `LOC` on CoNLL03 and `location` on WNUT. (2) Domain shift. The domain shift leads to the difficulty in recognizing the entities belonging to seen entity classes. (3) Description understanding. Description understanding is a crucial step for the success of zero-shot NER. For example, `MISC` on CoNLL03 is a collection of diverse fine-grained entity classes including events, nationalities, products and works of art.

3. META SELF-TRAINING

(A version of this chapter has been previously published in KDD 2021 [5].)

3.1 Introduction

Motivation. Deep neural networks typically require large amounts of labeled training data to achieve state-of-the-art performance. Recent advances with pre-trained language models like BERT [1], GPT-2 [3] and RoBERTa [2] have reduced this annotation bottleneck. In this paradigm, deep and large neural network models are trained on massive amounts of unlabeled data in a self-supervised manner. However, the success of these large-scale models still relies on fine-tuning them on large amounts of labeled data for downstream tasks. For instance, our experiments show 27% average improvement on multiple tasks when fine-tuning BERT with the full labeled training set (2.5K-705K labels) versus fine-tuning with limited amount of labels (e.g., 10 per class). This poses several challenges for many real-world tasks.

Not only is acquiring large amounts of labeled data for every task expensive and time consuming, but also not feasible in many cases due to data access and privacy constraints, especially when dealing with personal or sensitive data. This issue is exacerbated for sequence tagging tasks that require annotations at *token-* and *slot-level* as opposed to instance-level classification tasks. For example, an NER task can have slots like *B-PER, I-PER, O* marking the beginning, intermediate and out-of-span markers for person names, and similar slots for the names of location and organization. Similarly, language understanding models for dialog systems rely on effective identification of what the user intends to do (*intents*) and the corresponding values as arguments (*slots*) for use by downstream applications. Therefore, fully supervised neural sequence taggers are expensive to train for such tasks, given the requirement of thousands of annotations for hundreds of slots corresponding to the many different intents.

State-of-the-art. Semi-supervised learning (SSL) [31] is one of the approaches to address labeled data scarcity by making effective use of large amounts of unlabeled data in addition to task-specific labeled data. Self-training (ST, [32]), one of the earliest SSL approaches, has recently shown state-of-the-art performance for

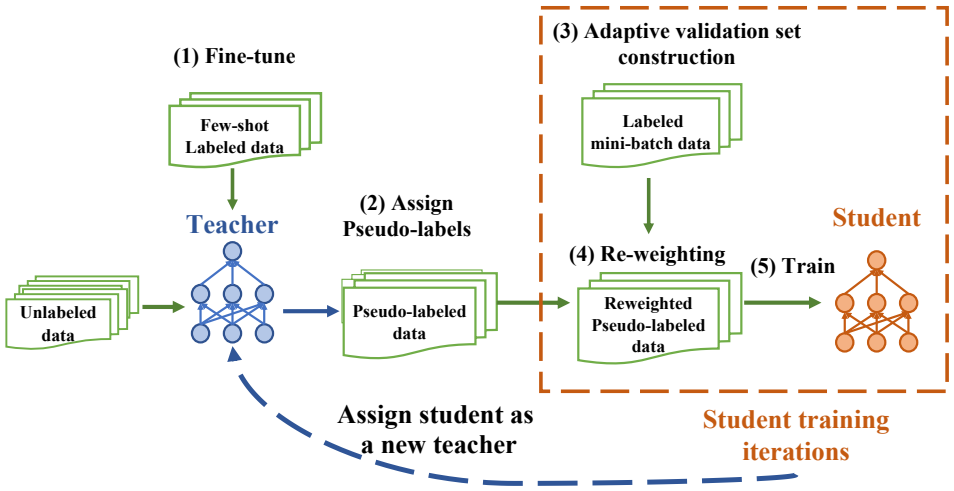


Figure 3.1. MetaST framework.

instance-level classification tasks like image or text classification [33]–[36] performing at par with supervised systems while using very few training labels. In contrast to such instance-level classification tasks, slot tagging or alternatively, *token-level classification* tasks have dependencies between the slots demanding different design choices for slot-level loss optimization for the limited labeled data setting. For instance, prior work [37] observe that standard self-training techniques do not work for slot tagging tasks in the low-data regime (e.g., with 10% labeled data for the target domain) due to error propagation and amplification in the iterative learning framework. On the positive side, there has been some success with careful task-specific data selection [8], [38], and more recently with distant supervision [39] leveraging external resources like knowledge bases (e.g., Wikipedia). In contrast to these prior work, we develop techniques for self-training with limited training labels and without any task-specific assumption or external knowledge resources.

Challenges. For self-training, a base model (*teacher*) is trained on some amount of labeled data and used to pseudo-annotate (task-specific) unlabeled data. The original labeled data is augmented with the pseudo-labeled data and used to train a *student* model. The student-teacher training is repeated until convergence. Traditionally in self-training frameworks, the teacher model pseudo-annotates unlabeled data without any sample selection. This may result in gradual drifts from self-training on noisy pseudo-labeled instances [37], [40]. In order to deal with noisy labels and training set biases, recent works have developed techniques to re-weight noisy samples leveraging prior knowledge of

the task [41], [42], or automatically learning from the underlying model and task data [33], [43], [44]. These prior techniques for learning to re-weight samples have been primarily developed for instance-level tasks like image [43], [44] and text [33] classification. A vanilla token-level extension of these techniques for slot tagging would assume a similar quality of the token-level pseudo-labels in a sequence disregarding the slot distribution and difficulty. This is not desirable for tasks like Named Entity Recognition in WikiAnn [45] involving 123 slots over 41 languages with variable difficulty and distribution in the data and across languages. This makes it imperative to design better sampling and re-weighting strategies for slot tagging tasks in contrast to random sampling or token-agnostic re-weighting employed for instance-level classification tasks [43].

To address the aforementioned challenges, we develop an adaptive learning mechanism to *re-weight noisy token-level pseudo-labels* to mitigate the effect of error propagation during self-training. To this end, we employ *meta-learning* [46]–[48] with the following meta-objective: *the best token-level re-weighting should minimize the model loss on a set of representative clean validation examples*. This formulation requires us to address two key research questions, namely, (i) How to construct an informative validation set for the meta-objective? and (ii) How to re-weight token-level noisy pseudo-labels to optimize the meta-objective for sequence labeling?

Prior works on meta-learning for instance-level tasks employ random sampling to construct this validation set for the meta-objective. However, we observe this to be detrimental for our setting as the model over-samples from the most populous categories and slot types ignoring their distribution and difficulty. To this end, we develop an adaptive mechanism to construct an informative validation set for meta-learning considering the diversity and uncertainty of the model for different slot types. Furthermore, we leverage this validation set to optimize the meta-objective for token-level loss estimation and re-weighting pseudo-labeled sequences from the teacher in a meta-learning framework.

Our task and framework overview. We focus on sequence labeling tasks with only a few annotated examples (e.g., $K = \{5, 10, 20\}$) per slot type for training and large amounts of task-specific unlabeled data. Figure 8.1 shows an overview of our framework with the following components and research contributions:

(i) *Self-training:* Our self-training framework leverages a pre-trained language model as a teacher and co-trains a student model with iterative knowledge exchange for neural sequence tagging with very few manually annotated training labels.
(ii) *Adaptive validation set construction for meta-learning:* Our few-shot learning setup assumes a small number of labeled training samples per slot type that are not equally informative. We develop an adaptive mechanism to select informative examples to construct the validation set for our meta-objective. To this end, we leverage stochastic loss decay of the student model as a proxy for its uncertainty for sample selection. This strategy is used in conjunction with the re-weighting mechanism in the next step.
(iii) *Token-level re-weighting with meta-learning:* Since pseudo labels from the teacher can be noisy, we leverage a meta-objective to re-weight them to improve the student model performance on the validation set obtained in previous step. In contrast to prior work on instance-level re-weighting, we perform *token-level* re-weighting for slot tagging tasks. Finally, we learn all of the above steps jointly with end-to-end learning in the self-training framework. We refer to our adaptive self-training framework with meta-learning based sample re-weighting mechanism as MetaST.
(iv) *Experiments:* We perform extensive experiments on six benchmark datasets for several tasks including multilingual Named Entity Recognition and slot tagging for user utterances from task-oriented dialog systems to demonstrate the generalizability of our approach across diverse tasks, slots, shots and languages. We adopt BERT and multilingual BERT as encoders and show that their performance can be significantly improved by nearly 10% for the few-shot settings with very few training labels (e.g., 10 manually labeled examples per slot type) and large amounts of unlabeled data.

3.2 Background and Problem Formulation

Sequence labeling and slot tagging. This is the task of identifying the entity *span* of several slot types (e.g., names of person, organization, location, date, etc.) in a text sequence. Formally, given a sentence with N tokens $X = \{x_1, ..., x_N\}$, an entity or slot value is a span of tokens $s = [x_{\mathrm{i}}, ..., x_{\mathrm{j}}](0 \leq \mathrm{i} \leq \mathrm{j} \leq N)$ associated with an entity class $c \in \mathcal{C}$. This task assumes a pre-defined tagging policy like `BIO` [49], where `B` marks the beginning of the slot, `I` marks

an intermediate token in the span, and `O` marks out-of-span tokens. These span markers are used to extract multi-token values for each of the slot types with phrase-level evaluation for the performance. For illustration, a user utterance can be labeled as "`play:O a:O popular:B-sort chant:B-music_item by:O brian:B-artist epstein:I-artist`", with slot types like *sort, music_item* and *artist*, with `BIO` denoting the span markers.

Few-shot semi-supervised sequence labeling. In this work, we study few-shot semi-supervised sequence labeling, where a model is trained with *very few manually labeled* and large amounts of unlabeled data. Formally, a few-shot semi-supervised setting for this task considers K labeled sentences that are manually annotated at token-level for each slot type $c \in C$, and M unlabeled sentences. The labeled samples are denoted as $(X_m^l = \{x_{m,n}^l\}, Y_m^l = \{y_{m,n}^l\})_{m=1,n=1}^{K\times|\mathcal{C}|,N}$ where $y_{m,n}^l \in \mathcal{C}$ are the slot labels. The M unlabeled sentences are denoted as $(X_m^u = \{x_{m,n}^u\})_{m=1,n=1}^{M,N}$, where $M \gg K \times |\mathcal{C}|$. Let $f(X;\theta)$ denote a tagging model that assigns a label to each token in the sequence with trainable parameters θ.

Self-training is one of the earliest semi-supervised approaches and has recently shown state-of-the-art performance for instance-level classification tasks. Consider $f(\cdot;\theta_{tea})$ and $f(\cdot;\theta_{stu})$ to denote the teacher and student models respectively in the self-training framework. The role of the teacher model (e.g., a pre-trained language model) is to assign pseudo-labels to unlabeled data that is used to train a student model. The teacher and student model can exchange knowledge and the training schedules are repeated till convergence. The success of self-training with deep neural networks in recent works has been attributed to a number of factors including stochastic regularization with dropouts [50] and data regularization with unlabeled / augmented data [34]. Formally, given m-th unlabeled sentence with N tokens $X_m^u = \{x_{m,1}^u, ..., x_{m,N}^u\}$ and C pre-defined labels, consider the pseudo-labels $\hat{Y}_m^{(t)} = [\hat{y}_{m,1}^{(t)}, ..., \hat{y}_{m,N}^{(t)}]$ generated by the teacher model at the t-th iteration where,

$$\hat{y}_{m,n}^{(t)} = \arg\max_{c\in C} f_{n,c}(x_{m,n}^u; \theta_{tea}^{(t)}). \tag{3.1}$$

The pseudo-labeled sequence data, denoted as $(X^u, \hat{Y}^{(t)}) = \{(x_{m,n}^u, \hat{y}_{m,n}^{(t)})\}_{m,n}^{M,N}$, is used to train the student model and learn its parameters as:

$$\hat{\theta}_{stu}^{(t)} = \arg\min_{\theta} \frac{1}{M}\frac{1}{N}\sum_{m=1}^{M}\sum_{n=1}^{N} \mathcal{L}(\hat{y}_{m,n}^{(t)}, f(x_{m,n}^u; \theta_{stu}^{(t-1)})), \tag{3.2}$$

where $\mathcal{L}(\cdot, \cdot)$ can be modeled as the cross-entropy loss.

3.3 Self Training with Adaptive Re-weighting

Given a pre-trained language model (e.g., BERT [1]) as the teacher, we first fine-tune it on the small labeled data with $K \times |C|$ annotated examples to make it aware of the underlying task. The fine-tuned teacher model is now used to pseudo-label the large unlabeled data. We consider the student model as another instantiation of the pre-trained language model that is trained over the pseudo-labeled data. However, our few-shot setting with limited labeled data results in a noisy teacher. A naive transfer of teacher knowledge to the student results in the propagation of noisy labels [37], [40] limiting the performance of the student model. To address this challenge, we develop an *adaptive* self-training framework to re-weight pseudo-labeled predictions from the teacher with a meta-learning objective that optimizes the token-level loss from the student model on a *judiciously constructed validation set* based on the model uncertainty (discussed next).

3.3.1 Adaptive Validation Set Construction for Meta-learning

Standard meta-learning techniques [43] for instance-level classification tasks, construct the validation set to optimize the meta-objective via random sampling. However, a naive sample selection is detrimental for the sequence labeling setup involving many slot types with variable difficulty and distribution in the data and across languages. Therefore, we develop an adaptive strategy to construct the validation set for effective data exploration. We empirically demonstrate its benefit over classic meta-learning approaches from prior works in experiments.

Prior works in meta-learning and active learning broadly leverage random sampling [43], easy [42] and hard example mining [41] or uncertainty methods [51] for sample selection. These strategies have been compared in prior works [51], [52] that show uncertainty-based methods to have better generalizability across diverse settings. While there are several approaches to uncertainty estimation including error decay [53] and predictive variance [51], these techniques have been developed for instance-level classification tasks, thereby, generating an overall estimate for the entire instance. In contrast, in this work, we are interested in leveraging token-level estimates corresponding to the different slot types and their associations.

Specifically, we leverage token-level uncertainty estimates to select samples that the model is uncertain about and can correspondingly benefit from knowing their labels. To this end, we leverage stochastic token-level loss decay from the model as a proxy for the model uncertainty to generate a validation set. This is used for estimating token-level weights and re-weighting pseudo labeled data in Section 3.3.2. This is an adaptive process as the model and corresponding uncertainty estimates improve over time, thereby, generating stochastic validation sets that are most representative of the difficulty of the underlying task at a given step during learning.

Consider the loss of the student model with parameters $\theta_{stu}^{(t)}$ on the labeled data $(\{x_{m,n}^l\}, \{y_{m,n}^l\})$ in the t-th iteration as $\mathcal{L}(\{y_{m,n}^l\}, \{f(x_{m,n}^l; \theta_{stu}^{(t)})\})$. We use the loss decay at any iteration as a proxy for the model uncertainty. This is measured by the difference between the successive stochastic losses encountered by the model for a token in any instance. Since the losses may widely vary across iterations given the few-shot assumption, we adopt the moving average of the stochastic losses for $(\{x_{m,n}^l\}, \{y_{m,n}^l\})$ in the latest R iterations as baseline $\mathcal{B}_m^{(t)}$ for smoothing the loss decay estimation. The baseline measure $\mathcal{B}_m^{(t)}$ at iteration t is given as:

$$\mathcal{B}_m^{(t)} = \frac{1}{min(R,t) \cdot N} \sum_{r=1}^{min(R,t)} \sum_{n=1}^{N} \mathcal{L}(y_{m,n}^l, f(x_{m,n}^l; \theta_{stu}^{(t-r)})). \tag{3.3}$$

Since the loss decay values are estimated on the fly, we want to balance exploration and exploitation. To this end, we add a smoothness factor δ to prevent the low loss decay samples (i.e. samples with low uncertainty in the constituent tokens) from never being selected again. Considering all of the above factors, we obtain the sampling weight of labeled data $(X_m^l = \{x_{m,n}^l\}, Y_m^l = \{y_{m,n}^l\})$ in iteration t as follows:

$$W_m^{(t)} \propto \max\left(\mathcal{B}_m^{(t)} - \frac{1}{N} \sum_{n=1}^{N} \mathcal{L}(y_{m,n}^l, f(x_{m,n}^l; \theta_{stu}^{(t)})), 0\right) + \delta. \tag{3.4}$$

A low value of $W_m^{(t)}$ indicates that the model loss for tokens in sequence $\{x_{m,n}^l\}$ in iteration t is similar to the average loss $\mathcal{B}_m^{(t)}$ encountered in last R iterations – depicting lower model uncertainty. In contrast, a higher value of $W_m^{(t)}$ depicts higher model uncertainty and therefore potential benefit in learning from knowing token-level labels, similar to the objective in an active learning setting.

The smoothness factor δ needs to be adaptive since the training loss is dynamic. To ensure the scale of smoothness factor δ is similar to loss decay value, we adopt the maximum of the loss decay values as the smoothness factor δ to encourage exploration.

For practical implementation considerations and speed-up, we re-estimate Equation 3.4 after a fixed number of steps to adapt to model changes and sample mini-batches of labeled data $\{\mathcal{V}_s^l\}$ as validation set for our meta-objective. This is used by the student model in the next step for re-weighting pseudo-labeled sequences from the teacher model. We demonstrate the impact of this adaptive sampling strategy via ablation study in experiments. As a minor note, the labeled data is only used to compute sample weight and not used for explicit training of the student model in this step.

3.3.2 Re-weighting Noisy Pseudo-Labeled Tokens

To mitigate error propagation from noisy pseudo-labeled sequences from the teacher, we leverage meta-learning to adaptively re-weight them based on the student model loss on the sampled validation set as our meta-objective. The validation set is obtained by our adaptive sampling strategy from the previous step. In contrast to prior work on instance-level image and text classification, we adapt the meta-learning framework to re-weight noisy pseudo-labeled samples at a token-level resolution for the sequence labeling task.

Consider the pseudo-labels $\{\hat{Y}_m^{(t)} = [\hat{y}_{m,1}^{(t)}, ..., \hat{y}_{m,N}^{(t)}]\}_{m=1}^{M}$ from the teacher in the t-th iteration with m and n indexing the instance and a token in the instance, respectively. In classic self-training, we update the student parameters leveraging pseudo-labels with inner step size α as follows:

$$\hat{\theta}_{stu}^{(t)} = \hat{\theta}_{stu}^{(t-1)} - \alpha \nabla \Big(\frac{1}{M} \frac{1}{N} \sum_{m=1}^{M} \sum_{n=1}^{N} \mathcal{L}(\hat{y}_{m,n}^{(t)}, f(x_{m,n}^{u}; \hat{\theta}_{stu}^{(t-1)})) \Big). \tag{3.5}$$

Now, to downplay noisy token-level labels, we leverage meta-learning to re-weight pseudo-labeled data. Our objective is to measure the impact of a training example towards the performance on

validation set $\mathcal{V}^l$ at iteration t. To this end, we leverage the idea of weight perturbation [43], [54] to change the weight of each token in each sequence of the mini-batch by $\epsilon^{(t)}_{m,n}$ at iteration t as:

$$\hat{\theta}^{(t)}_{stu}(\epsilon) = \hat{\theta}^{(t-1)}_{stu} - \alpha \nabla (\frac{1}{M}\frac{1}{N}\sum_{m=1}^{M}\sum_{n=1}^{N}[\epsilon^{(t)}_{m,n} \cdot \mathcal{L}(\hat{y}^{(t)}_{m,n}, f(x^u_{m,n}; \hat{\theta}^{(t-1)}_{stu}))]). \tag{3.6}$$

Weight perturbation is used to discover data points that are most important to improve the model performance on the validation set where the sample importance is given by the magnitude of the negative gradients. We can now find the optimal value for the perturbation $\epsilon^{(t)*}_{m,n}$ that minimizes the student model loss on the validation set $\mathcal{V}^l$ at iteration t as:

$$\epsilon^{(t)*}_{m,n} = argmin_{\epsilon_{m,n}} \frac{1}{M}\frac{1}{N}\sum_{m=1}^{M}\sum_{n=1}^{N}\mathcal{L}(\hat{y}^{(t)}_{m,n}, f(x^u_{m,n}; \hat{\theta}^{(t)}_{stu}(\epsilon_{m,n})) \tag{3.7}$$

The token weights are obtained by minimizing the student model loss on sampled mini-batches of validation data $\{\mathcal{V}^l_s\}$ obtained from Eq. 3.4. To obtain a cheap estimate of the meta-weight at step t, we take a single gradient descent step for the sampled validation mini-batch $\mathcal{V}^l_s$ as:

$$u^{(t)}_{m,n,s} = -\frac{\partial}{\partial \epsilon_{m,n,s}}\left(\frac{\sum_{m=1}^{|\mathcal{V}^l_s|}\sum_{n=1}^{N}\mathcal{L}(y^l_{m,n}, f(x^l_{m,n}; \hat{\theta}^{(t)}_{stu}(\epsilon)))}{|\mathcal{V}^l_s| \cdot N}\right)\Big|_{\epsilon_{m,n,s}=0} \tag{3.8}$$

We set the token weights to be proportional to the negative gradients to reflect the importance of pseudo-labeled tokens in the sequence. Since sequence labeling tasks have dependencies between the slot types and tokens, it is difficult to obtain a good estimation of the weights based on a single mini-batch of examples. Therefore, we sample S mini-batches of validation sets $\{\mathcal{V}^l_1, ..., \mathcal{V}^l_S\}$ with the adaptive sampling strategy in Equation 3.4 and calculate the mean of the gradients to obtain a robust gradient estimate. The overall meta-weight of pseudo-labeled token $(x^u_{m,n}, \hat{y}_{m,n})$ is obtained as:

$$w^{(t)}_{m,n} = \max(\frac{1}{S}\sum_{s=1}^{S} u^{(t)}_{m,n,s}, 0). \tag{3.9}$$

Since a negative weight indicates a pseudo-label of poor quality that would potentially degrade the model performance, we set such weights to 0 to filter them out. We empirically study the impact of S in experiments.

Finally, we update the student model parameters while accounting for token-level re-weighting as:

$$\hat{\theta}^{(t)}_{stu} = \hat{\theta}^{(t-1)}_{stu} - \alpha \nabla \Big(\frac{1}{M}\frac{1}{N}\sum_{m=1}^{M}\sum_{n=1}^{N}[w^{(t)}_{m,n} \cdot \mathcal{L}(\hat{y}^{(t)}_{m,n}, f(x^u_{m,n}; \hat{\theta}^{(t-1)}_{stu}))]\Big). \tag{3.10}$$

We demonstrate the impact of this token-level re-weighting mechanism with ablation study in experiments.

3.3.3 Student Teacher Iterative Training

We first fine-tune the teacher model with few labeled data for each slot for each task and initialize the student as a copy of the teacher. In every self-training iteration, the teacher generates noisy token-level pseudo-labels for each sequence which are used to train the student model with sample selection and token-level re-weighting in a meta-learning framework.

At the end of given self-training iterations T, we assign the student model to be the new teacher model (i.e., $\theta_{tea} = \theta_{stu}^{(T)}$), and repeat the above steps till convergence. We further utilize the labeled data ($\{x_{m,n}^l, y_{m,n}^l\}$) to fine-tune the new teacher model $f(\cdot, \theta_{tea}^{(t)})$ with standard cross-entropy loss minimization. We explore the effectiveness of this step with an ablation study in experiments.

3.4 Experiments

We evaluate the proposed method MetaST across diverse tasks, slot (entity) types, number of shots (manually labeled instances) and languages to demonstrate its impact for the few-shot learning setup for sequence labeling with limited amount of training labels. We compare against several state-of-the-art existing methods and demonstrate significant improvements in diverse settings along with ablation studies to evaluate the contribution of different components.

3.4.1 Experimental Setup

Datasets. We perform large-scale experiments with six different datasets including user utterances from task-oriented dialog systems and multilingual Named Entity Recognition tasks as summarized in Table 3.1. *(a) Email*. This consists of natural language user utterances for email-oriented user actions like sending, receiving or searching emails with attributes like date, time, topics, and people. *(b) SNIPS* is a public benchmark dataset [55] of user queries from multiple domains including music, media, and weather. *(c) MIT Movie and Restaurant* corpus [28] consist of similar user utterances for movie and restaurant domains. (d) CoNLL03 [20] and Wikiann [45] are public benchmark datasets for multilingual Named Entity Recognition. CoNLL03 is a collection of news wire articles

from the Reuters Corpus from 4 languages with manual annotations, whereas Wikiann comprises of extractions from Wikipedia articles from 41 languages with automatic annotation leveraging meta-data for different entity types like `ORG, PER, LOC`.

For every dataset, we sample $K \in \{5, 10, 20, 100\}$ manually labeled sequences for each slot type from the training data, and add the remaining to the unlabeled set while ignoring their labels – following standard setups for semi-supervised learning. We repeatedly sample K labeled instances three times for multiple runs and report average F1 score with standard deviation across the runs in Table 3.2 and Table 3.3.

Table 3.1. Dataset summary. We sample $K \in \{5, 10, 20, 100\}$ labeled sequences for each slot type from #Train, and add the remaining to the Unlabeled set while ignoring their labels.

Dataset	#Slots	#Train/ #Unlabeled	#Test	#Lang
Email	20	2.5K	1k	EN
SNIPS	39	13K	0.7K	EN
MIT Movie	12	8.8K	2.4K	EN
MIT Restaurant	8	6.9K	1.5K	EN
Wikiann (EN)	3	20K	10K	EN
CoNLL03 (EN)	4	15K	3.6K	EN
CoNLL03	16	38K	15K	4
Wikiann	123	705K	329K	41

Encoder. Pre-trained language models like BERT [1], GPT-2 [3] and RoBERTa [2] have shown state-of-the-art performance for various natural language processing tasks. In this work, we adopt one of them as a base encoder by initializing the teacher with pre-trained BERT-base model and a randomly initialized token classification layer.

Baselines. The first baseline we consider is the fully supervised BERT model trained on all available training data which provides the ceiling performance for every task. Each of the other models are trained on K training labels per slot type. We adopt several state-of-the-art semi-supervised methods as baselines: (1) CVT [56] is a semi-supervised sequence labeling method based on cross-view training. For unlabeled data, CVT matches auxiliary prediction based on parts of a sentence with prediction based on the whole input to improve its representation learning. (2) SeqVAT [57]

Table 3.2. F1 score comparison of models for sequence labeling on different datasets averaged over multiple runs. All models (except CVT and SeqVAT) use the same BERT encoder. F1 score of our model for each task is followed by standard deviation and percentage improvement (std dev; ↑) over BERT with 10 manually labeled training examples per slot.

Method	SNIPS	Email	Movie	Restaurant	CoNLL03 (EN)	Wikiann (EN)
# Slots	39	20	12	8	4	3
Full-supervision						
BERT	95.80	94.44	87.87	78.95	92.40	84.04
Few-shot supervision (10 labels per slot)						
BERT	79.01	87.85	69.50	54.06	71.15	45.61
Few-shot supervision (10 labels per slot) + unlabeled data						
CVT	78.23	78.24	62.73	42.57	54.31	27.89
SeqVAT	78.67	72.65	67.10	51.55	67.21	35.16
MT	79.48	89.53	67.62	51.75	68.67	41.43
VAT	79.08	89.71	70.17	53.34	65.03	38.81
Classic ST	83.26	90.70	71.88	56.80	70.99	46.15
BOND	83.54	89.75	70.91	55.78	69.56	48.73
MetaST	**88.23 (0.04;↑12%)**	**92.18 (0.47;↑4.93%)**	**77.67 (0.10;↑11.76%)**	**63.83 (1.62;↑18.07%)**	**76.65 (0.73;↑7.73%)**	**56.61 (0.4;↑24.12%)**

Table 3.3. F1 score comparison of models for sequence labeling on multilingual datasets using the same multilingual mBERT encoder. F1 score of MetaST for each task is followed by standard deviation in parentheses and percentage improvement (↑) over mBERT with 10 manually labeled training examples per slot.

Dataset	#Lang	#Slots	Full Sup.	10 labels per slot	10 labels per slot + unlabeled data				
			mBERT	mBERT	MT	VAT	Classic ST	BOND	MetaST
CoNLL03	4	16	87.67	70.77	68.34	67.63	72.69	72.79	**76.41 (0.47) (↑ 7.97%)**
Wikiann	41	123	87.17	79.67	80.23	78.82	80.24	79.57	**81.61 (0.14) (↑ 2.42%)**

incorporates adversarial training with conditional random field layer for semi-supervised sequence labeling. (3) Mean Teacher (MT) [58] averages model weights to obtain an aggregated teacher and applies a consistency loss between the predictions from the student model and that from the aggregated teacher on unlabeled data. (4) VAT [59] improves the robustness of the conditional label distribution for each input data point against local perturbation. (5) Classic ST [32] is simple self-training method with hard pseudo-labels; (6) BOND[1] [39] is a recent work on self-training for sequence labeling with confidence-based sample selection and forms a strong baseline for our work.

[1]↑ We replace fine-tuning step with distant supervision by fine-tuning on labeled data.

The above semi-supervised learning (SSL) methods augment task-specific knowledge from manually annotated data with domain knowledge from unlabeled data. In contrast to traditional SSL methods, few-shot learning settings involve *very few* manually annotated training labels resulting in a noisy / weak model to start with. Consequently, a naive augmentation from large amounts of unlabeled data results in drift without accounting for the noise and model uncertainty. To this end, we develop a robust sample selection and re-weighting mechanism for adaptive learning.

We implement our framework in Pytorch and use Tesla V100 gpus for experiments.

3.4.2 Experimental Results

We first present the overall performance comparison of MetaST with several state-of-the-art methods for few-shot sequence labeling followed by several control experiments.

10-shot sequence labeling performance comparison. Table 3.2 shows the performance comparison among different models with K=10 labeled examples per slot type. The fully supervised BERT baseline trained on thousands of labeled examples provides the ceiling performance for the few-shot setting. We observe that the proposed method MetaST significantly outperforms all other methods across all datasets – including the models that also use the same BERT encoder as ours like MT, VAT, Classic ST and BOND with corresponding average performance improvements as 14.22%, 14.90%, 8.46% and 8.82% respectively. This demonstrates the advantage of our adaptive / meta self-training design. Non-BERT models like CVT and SeqVAT are consistently worse than other baselines.

Task variation. We also observe variable performance of the models across different tasks. Specifically, the performance gap between the best few-shot model and the fully supervised model varies significantly across tasks. MetaST achieves close performance to the fully-supervised model in some datasets (e.g. SNIPS and Email) but has bigger room for improvement in others (e.g. CoNLL03 (EN) and Wikiann (EN)). This can be attributed to the following factors.

(i) *Labeled training examples and slots.* The total number of labeled training instances for our K-shot setting is given by $K \times \#Slots$. Therefore, for tasks with higher number of slots and consequently more training labels, most of the models perform better including MetaST. Task-oriented dialog

systems with more slots and inherent dependency between the slot types benefit more than NER tasks.

(ii) *Task difficulty:* User utterances from task-oriented dialog systems for some of the domains like weather, music and emails contain predictive query patterns and limited diversity. In contrast, Named Entity Recognition datasets are comparatively diverse and require more training labels to generalize well. Similar observations are also depicted in Table 3.3 for multilingual NER tasks with more slots and consequently more training labels from multiple languages as well as richer interactions across the slots from different languages.

Table 3.4. F1 scores of different models with 200 manually labeled examples for each task. The percentage improvement (↑) is over the BERT model with few-shot supervision.

Dataset	BERT (Full Sup.)	BERT (Few-shot Sup.)	MetaST (%Improvement)
MIT Movie	87.87	75.81	**80.33 (↑ 5.96%)**
MIT Restaurant	78.95	60.12	**67.86 (↑ 12.87%)**
CoNLL03 (EN)	92.40	77.48	**81.61 (↑ 5.33%)**
Wikiann (EN)	84.04	62.04	**71.27 (↑ 14.88%)**
Average	85.82	68.86	**75.27 (↑ 9.31%)**

Controlling for the total amount of labeled data. In order to control for the variable amount of training labels across different datasets / tasks, we perform another experiment where we vary the number of training labels for different slot types while keeping the total number of labeled instances for each dataset similar (ca. 200). Results are shown in Table 3.4. To better illustrate the effect of the number of training labels, we choose tasks with lower performance in Table 3.2 for this experiment. Comparing the results in Tables 3.2 and 3.4, we observe that the performance of MetaST improves with more training labels for all the tasks .

Effect of varying the number of training labels K per slot. Table 3.5 shows the improvement in the performance of MetaST when increasing the number of training labels for each slot type in the SNIPS dataset. Similar trends can be found in other datasets. As we increase the amount of labeled training instances, the performance of BERT and all the models improve. Correspondingly, the relative improvement between MetaST and the baselines decreases although MetaST still improves

Table 3.5. Variation in model performance on varying K training labels per slot on SNIPS dataset with 39 slots. The percentage improvement (↑) is over the BERT model with few-shot supervision.

#Shots	5	10	20	100
Few-shot supervision				
BERT	70.63	79.01	86.81	93.90
Few-shot supervision + unlabeled data				
CVT	69.82	78.23	86.81	94.61
SeqVAT	69.34	78.67	85.05	91.46
MT	70.85	79.48	87.31	94.26
VAT	71.34	79.08	88.19	94.53
Classic ST	72.59	83.26	88.32	93.92
BOND	72.85	83.54	88.93	94.22
MetaST	**81.56 (↑15%)**	**88.22 (↑12%)**	**91.99 (↑6%)**	**95.39 (↑2%)**

over all of them. For example, while MetaST improves over BERT by 15% for the 5-shot setting, the corresponding improvement reduces to 2% for the 100-shot setting.

In the self-training framework, given the ceiling performance for every task and the improved performance of the teacher with more training labels – there is less room for (relative) improvement of the student over the teacher model. Consider SNIPS for an illustration. Our model obtains 12% and 2% improvement over the few-shot BERT model for the 10-shot and 100-shot setting with F1-scores as 88.22% and 95.39%, respectively. The ceiling performance for this task at 95.8% is obtained by training BERT on the fully labeled dataset with $13K$ labeled examples. This demonstrates that MetaST is most impactful for low-resource settings with few training labels for a given task.

3.4.3 Ablation analysis

Table 4.4 demonstrates the impact of different MetaST components with ablation analysis. We observe that soft pseudo-labels hurt the model performance compared to hard pseudo-labels, as also shown in recent work [60]. Such a performance drop may be attributed to soft labels being less informative compared to sharpened ones. Removing the iterative teacher fine-tuning step (Section 3.3.1) also hurts the overall performance.

Continued pre-training versus self-training. Recent work [61] show the benefit of continued pre-training with task-specific unlabeled data for adapting pre-trained language models to the task-domain. To contrast continued pre-training with self-training, we *further pre-train* BERT with masked language modeling objective on in-domain unlabeled data and then fine-tune it with few labeled examples denoted as "BERT (Continued Pre-training + Few-shot Supervision)". The pre-training step improves BERT performance over the baseline on SNIPS but degrades the performance on CoNLL03. This indicates that continued pre-training can improve the performance of few-shot supervised BERT on specialized tasks (e.g., SNIPS) with different data distribution than the original pre-training data (e.g., Wikipedia), but may not help for general domain ones like CoNLL03 with overlapping data from Wikipedia. In contrast to the above baseline, MetaST brings significant improvements on both datasets. This demonstrates the generality and flexibility of self-training over pre-training as also observed in contemporary work [62] on image classification.

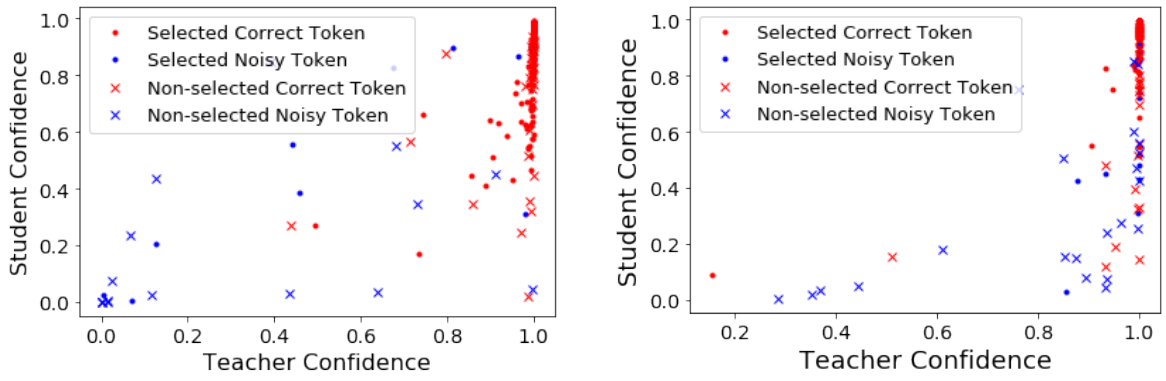
Table 3.6. Ablation analysis of our framework MetaST with 10 labeled examples per slot on SNIPS and CoNLL03 (EN).

Method	Datasets	
	SNIPS	CoNLL03
BERT w/ Few-shot Supervision	79.01	71.15
BERT w/ Continued Pre-training + Few-shot Supervision	83.96	69.84
Classic ST w/ Hard Pseudo-Labels	83.26	70.99
Classic ST w/ Soft Pseudo-Labels	81.17	71.87
MetaST w/ Soft Pseudo-Labels	86.16	75.84
MetaST w/o Iterative Teacher Fine-tune	85.64	72.74
MetaST w/o Adaptive Valid Set Construction	86.63	75.02
Pseudo-labeled Data Selection and Re-weighting Strategies		
MetaST w/o Re-weighting	85.48	73.02
MetaST (Easy Sample Selection)	85.56	74.53
MetaST (Difficult Sample Selection)	86.34	68.06
MetaST (Instance-level Re-weighting)	86.46	74.54
MetaST (ours) w/ Hard Pseudo-Labels, Token-level Re-weighting, Adaptive Valid Set Construction	**88.23**	**76.65**

Adaptive Validation Set Construction. We perform an ablation study by removing adaptive validation set construction from the proposed MetaST (denoted as "MetaST w/o Adaptive Valid Set Construction"). Removing this component leads to around 2% performance drop on an average demonstrating the impact of adaptive validation set for meta-learning. Moreover, the performance drop on SNIPS (39 slots) is larger than that on CoNLL03 (4 slots). This demonstrates that adaptive validation set construction is more helpful for tasks with more slot types – where diversity and data distribution necessitate a better exploration strategy in contrast to random sampling employed in prior meta-learning works.

Re-weighting strategies. To explore the role of token-level re-weighting for pseudo-labeled sequences (discussed in Section 3.3.2), we replace our meta-learning component with different data selection strategies based on the model confidence. One data selection strategy chooses pseudo-labeled tokens uniformly without any re-weighting (referred to as "MetaST w/o Re-weighting"). The sampling strategy with weights proportional to the model confidence favors easy instances (referred to as "MetaST (Easy Sample Selection)"), whereas the converse favors difficult ones (referred to as "MetaST (Difficult Sample Selection)"). We observe that the meta-learning based re-weighting strategy performs the best. Interestingly, "MetaST (Easy Sample Selection)" outperforms "MetaST (Difficult Sample Selection)" significantly on CoNLL03 (EN) but achieves slightly lower performance on SNIPS. This demonstrates that difficult samples are more helpful when the quality of pseudo-labeled data is relatively high. In contrast, the sample selection strategy focusing on difficult samples introduces noisy examples with lower pseudo-label quality. Therefore, sampling strategies may need to vary for different datasets, thereby, demonstrating the necessity of adaptive data re-weighting as in our framework MetaST. Moreover, MetaST significantly outperforms classic self-training strategies with hard and soft pseudo-labels demonstrating the effectiveness of our design.

Token-level re-weighting versus instance-level re-weighting. Prior meta-learning works [43] re-weight entire instances for classification tasks. In order to compare our token-level re-weighting mechanism for sequence labeling tasks, we replace our token-level re-weighting component by sentence-level re-weighting – which uses average of token weights in the same sentence as the sentence weight (referred to as MetaST (Instance-level Re-weighting)"). Table 4.4 shows that token-



(a) SNIPS (b) CoNLL03

Figure 3.2. Visualization of MetaST re-weighting examples on SNIPS and CoNLL03 (EN).

level re-weighting outperforms instance-level re-weighting on SNIPS and CoNLL03 by 2.05% and 2.76% respectively, demonstrating the benefit of token-level choice for sequence labeling.

Analysis of pseudo-labeled data re-weighting. To visually explore the adaptive re-weighting mechanism, we illustrate token-level re-weighting of MetaST on SNIPS and CoNLL03 (EN) datasets with K=10 shot at step 100 in Fig. 3.2. We observe that the selection mechanism filters out most of the noisy pseudo-labels (colored in blue) including even those with high teacher confidence (X-axis).

4. LITE PROMPTED SELF-TRAINING

(A version of this chapter has been previously published in NAACL 2022 [6].)

4.1 Introduction

Large pre-trained language models (PLMs) have obtained state-of-the-art performance in several natural language understanding tasks [1], [2], [63]. Despite their remarkable success, their performance is still not satisfactory when fine-tuning with only a handful of examples, thereby hindering widespread adoption in real-world applications where a large scale of labeled data is difficult to obtain. While models like GPT-3 [64] have obtained impressive few-shot performance with in-context task adaptation, they have a significant performance gap relative to fully supervised SoTA models. For instance, the few-shot GPT-3 performance is 20 points worse than the fully-tuned DeBERTa [65] on SuperGLUE. This poses significant challenges for many real-world tasks where large labeled data is difficult to obtain.

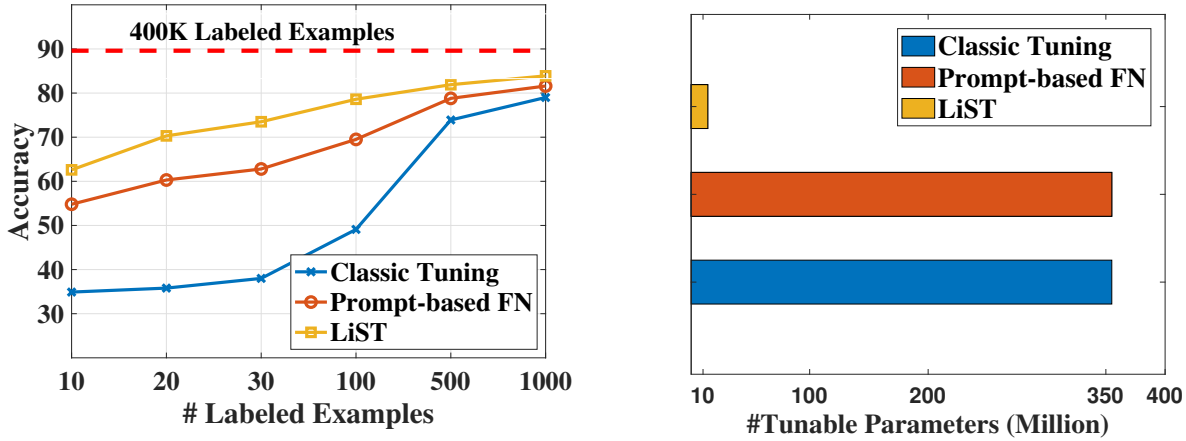
In this work, we present a new fine-tuning method `LiST` that aims to improve few-shot learning ability over existing fine-tuning strategies using two techniques as follows.

The first one is to leverage self-training with large amounts of unlabeled data from the target domain to improve model adaptation in few-shot settings. Prompt-based fine-tuning [66] have recently shown significant improvements over classic fine-tuning in the few-shot learning setting. In this work, we demonstrate that self-training with unlabeled data is able to significantly improve prompt-based fine-tuning [66] where we iteratively update a pair of teacher and student models given *natural language prompts* and *very few labeled examples for the task*. Since the uncertain teacher in few-shot setting produces noisy pseudo-labels, we further use meta-learning to re-weight the pseudo-prompt labels.

Traditional self-training can be expensive if we have to update all model parameters iteratively. To improve the efficiency of self-training, the second key technique is that we use a small number of task-specific adapter parameters in the PLM that are updated with the above technique, while keeping the large PLM encoder fixed. We demonstrate such light-weight tuning with self-training can match the setting where all model parameters are tuned. This enables **efficient use of self-training** and **reduces the storage cost** of the fine-tuned model since multiple fine-tuned models

can now share the same PLM as backbone during inference. Note that the computational cost of inference is out of the scope of this work and previous work has studied several ways to address it including model distillation [67], pruning [68], etc.

We perform extensive experiments in six natural language understanding tasks to demonstrate the effectiveness of LiST. We devise a comprehensive evaluation framework considering the variance in few-shot performance of PLMs with different shots, random seeds and splits. Results show that LiST improves over traditional and more recent prompt-based FN methods by 35% and 6%, respectively, with 96% reduction in number of trainable parameters given only 30 labeled examples for each downstream task. Figure 4.1 shows the results on MNLI [69] as an example. Meanwhile, we compare our proposed method to several SoTA few-shot semi-supervised learning approaches and the experimental results show that LiST brings improvement over the best baseline by 6% given 30 labeled examples for each downstream task.



(a) Performance Comparison

(b) Tunable Parameters

Figure 4.1. LiST leverages prompt-based fine-tuning (FN) with unlabeled data for label-efficiency and adapters for reducing tunable parameters. (a) shows classic tuning, prompt-based FN and LiST using RoBERTa-large as backbone on MNLI task for a comparison. The red dash line depicts ceiling performance with full supervision with RoBERTa-large. (b) shows the number of tunable parameters for each method.

Problem statement. Each downstream task in our framework consists of very few labeled training examples $\mathcal{D}_K^{Train}$ for different shots $K \in \{10, 20, 30\}$ where $|\mathcal{D}_K^{Train}| = K$, unlabeled data $\mathcal{D}^U$ where $\mathcal{D}^U \gg \mathcal{D}_K^{Train}$, and a test set $\mathcal{D}^{Test}$.

Given above dataset $\mathcal{D}_K = \mathcal{D}_K^{Train} \cup \mathcal{D}^U$ for a task with shots K, a PLM with parameters Θ_{PLM} and loss function $\mathcal{L}$, we want to adapt the model for the few-shot learning task by introducing a small number of tunable model parameters $\psi \ll \Theta_{\text{PLM}}$.

4.2 Background on Model Fine-tuning

Given a text sequence x or a pair of sequences $\{x_1, x_2\}$ separated by special operators (e.g., [CLS] and [SEP]) and a language model encoder $enc(\theta)$ parameterized by θ – classic fine-tuning popularized by [1] leverages hidden state representation $h_{[CLS]}$ of the sequence(s) obtained from enc([[CLS] x_1 [SEP] x_2 [SEP]]) as input to a task-specific head $softmax(W^T \cdot h_{[CLS]})$ for classification, where $W \in \mathbb{R}^{d \times L}$ with d and L representing the hidden state dimension and number of classes, are randomly initialized tunable parameters. In the process it updates both task-specific head W and encoder θ parameters jointly.

However, this introduces a gap between pre-training and fine-tuning objective with disparate label spaces and additional randomly initiated parameters W introduced for task-specific fine-tuning. This is particularly challenging for few-shot classic fine-tuning, where the limited labeled data is inadequate for adapting the task-specific head and PLM weights effectively. Prompt-based FN [66], [70] addresses this gap, by re-formulating the objective as a cloze-style auto-complete task. This is done by adding a phrase (also called *prompt*) to a sentence like $x_1 =$ "`contains no wit, only labored gags`" in the form of $\tilde{x} = x_1 \oplus$ "`It was [MASK]`", where $\oplus$ denotes concatenation of two strings; and output mappings (also called *verbalizers*) from vocabulary $\mathcal{V}$ to the label space $\mathcal{Y}$ like "`{great, terrible}`" corresponding to positive and negative classes (refer to Figure 4.3 for an example). The probability of predicting class $y \in \mathcal{Y}$ is equal to calculating the probability of corresponding label word $v \in \mathcal{V}$:

$$p([\text{MASK}] = v|\tilde{x}) = \frac{\exp(W_v^T \cdot h_{[\text{MASK}]})}{\sum_{v' \in V} \exp(W_{v'}^T \cdot h_{[\text{MASK}]})} \tag{4.1}$$

where W_v indicates the tunable parameters. Since it is identical to masked language modeling (MLM), W_v is initialized by pre-trained weights of PLMs.

In this work, we demonstrate lite self-training with unlabeled data to significantly improve prompt fine-tuning of PLMs in few-shot settings.

4.3 Methodology

4.3.1 Overview

We adopt a PLM (e.g., RoBERTa [2]) as the *shared encoder* for both the student and teacher for self-training. The shared PLM encoder is frozen and not updated during training. We introduce *tunable adapter parameters* in both teacher and student (discussed in Section 4.3.2) that are iteratively tuned during self-training. Refer to Figure 8.1 for steps in the following discussion.

We first use prompt-based fine-tuning to update the teacher adapter (*Step 1*) with few-shot labeled examples and leverage the teacher model to assign pseudo-prompt labels (*Step 2*) on unlabeled data $\mathcal{D}^u$. The teacher is often uncertain in few-shot learning and produces noisy pseudo-labels. Therefore, we adopt meta-learning (discussed in Section 4.3.3) to re-weight the noisy pseudo-labeled samples (*Step 3*). The re-weighted data is used to train the student adapter (*Step 4*). Since adapter training with noisy pseudo labels is quite unstable, we introduce knowledge distillation warmup (discussed in Section 4.3.3). Finally, we assign the trained student adapter to be the new teacher adapter (*Step 5*). Following *true* few-shot learning settings, we do not use any held-out development or validation set. Therefore, we repeat the above steps for a pre-defined number of times ($M = 6$). Throughout the training, we keep the shared student and teacher encoder parameters frozen and update the corresponding adapter parameters along with their language model heads.

4.3.2 Lightweight Prompt Adapter Tuning

The predominant methodology for task adaptation is to tune all of the trainable parameters of the PLMs for every task. This raises significant resource challenges both during training and deployment. A recent study [71] show that PLMs have a low instrinsic dimension that can match the performance of the full parameter space. To adapt PLMs for downstream tasks with a small number of parameters, adapters [72] have recently been introduced as an alternative approach for lightweight tuning. Consider the following scenario for demonstration, where we want to use RoBERTa-large with $\mathcal{M} = 355M$ parameters as the PLM for $\mathcal{T} = 100$ tasks. Full fine-tuning for this scenario requires updating and storing $\mathcal{M} \times \mathcal{T} = 35.5B$ parameters. Now, consider fine-tuning with `LiST` that requires $\mathcal{A} = 14M$ (tunable) adapter parameters for every task while keeping the

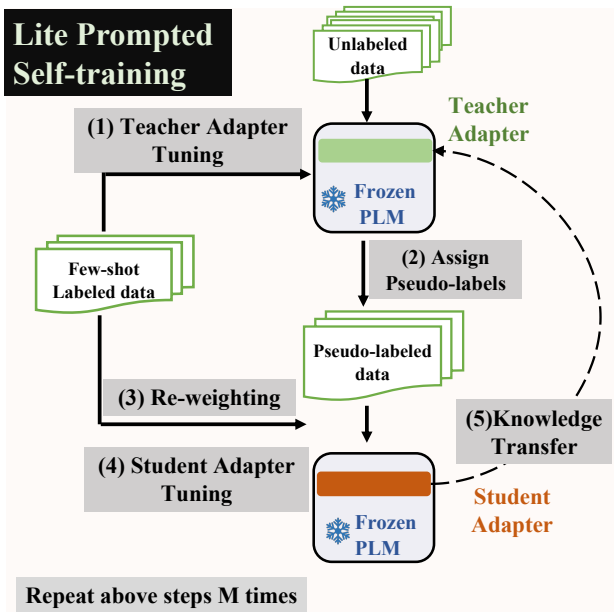


Figure 4.2. Lite prompted self-training on unlabeled data with prompts and adapters make efficient few-shot learners with `LiST`.

PLM fixed. This results in overall $\mathcal{M} + \mathcal{A} \times \mathcal{T} = 1.8B$ parameters, thereby, reducing the overall storage cost by 20x. Adapters have been shown to match the PLM performance in fully supervised settings with thousands of training labels in classic fine-tuning. In contrast, this is the first work to study the role of adapters in few-shot prompt-based FN. We explore different design and placement choices of adapters in few-shot settings and investigate the performance gap with fully supervised as well as fully tunable parameter space.

The adapter tuning strategy judiciously introduces new parameters into the original PLMs. In contrast to standard prompt-based FN that updates all the PLM parameters Θ_{PLM}, prompt-adapter tuning only updates the newly introduced adapter parameters as well as the (masked) language model head of the PLM (jointly denoted as ψ), while keeping the remaining parameters of the original network frozen. The adapter used in `LiST` consists of two fully connected layers as shown in Figure 4.4, where a feedforward layer down projects input representations to a low dimensional space d (referred as the bottleneck dimension), and another feedforward layer up projects the low-dimensional features back to the original dimension. However, these newly-inserted parameters can cause divergence resulting in up to 20% performance degradation in few-shot settings (discussed in

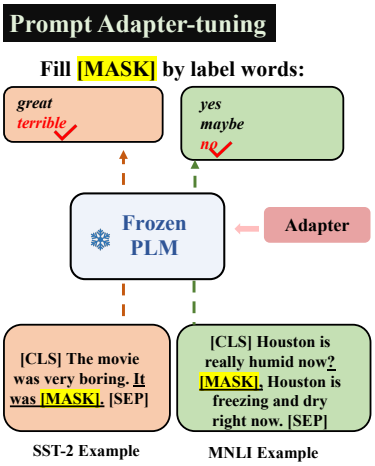


Figure 4.3. The underlined text depicts task prompt to transform classification into Fill-in-MASK task. Label words are used as proxy for original task labels.

Section 4.4.3). To handle this issue, we adopt a skip-connection design where the adapter parameters are initialized with zero-mean small Gaussian noise.

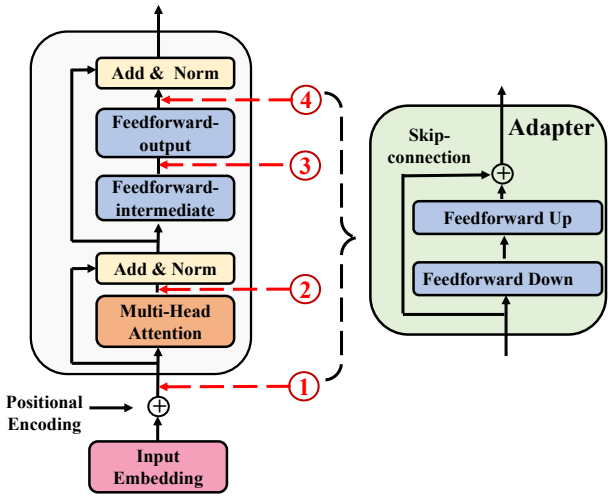


Figure 4.4. `LiST` explores several adapter placement choices (numbered positions in left) in standard Transformer architecture, with adapter design shown in right.

Adapter placement. Prior works on lightweight adaptation tune bias [73] or embeddings [74] of Transformers in fully-supervised settings for improving parameter-efficiency with minimal performance loss. However, for few-shot settings, we note that adapter placement is critical to bridge the performance gap with that of a fully tunable model and the choices of tuning bias or embedding can result in upto 10% performance degradation (discussed in Section 4.4.3). To this

end, we explore several choices of adapter placement (refer to Figure 4.4) corresponding to the most important transformer modules, namely, embedding, intermediate feedforward, output feedforward and attention module in *every layer* of the Transformer. Based on empirical experiments (refer to Section 4.4.3) across six diverse NLU tasks, we observe the feedforward output and attention modules to be the most important components for parameter-efficient adaption in few-shot settings.

Formally, consider $\widetilde{\mathcal{D}}_K^{Train} = \{\widetilde{x}^l, \widetilde{y}^l\}$ to be the few-shot labeled data and $\widetilde{\mathcal{D}}^U = \{\widetilde{x}_u\}$ to be the unlabeled data, where we transform the input sequences x to cloze-style input $\widetilde{x}$ containing a single mask following the prompting strategy outlined in Section 4.2. We use the same pattern templates and verbalizers (output mapping from the task-specific labels $\mathcal{Y}$ to single tokens in the vocabulary $\mathcal{V}$) from traditional prompt-based FN works [66]. Given the above adapter design and placement of choice with parameters ψ, a dataset $\widetilde{\mathcal{D}}_K^{Train}$ with shots K, a PLM encoder enc with parameters Θ_{PLM}, where $\Theta_{\text{PLM}} \gg \psi$, we want to perform the following optimization for efficient model adaptation:

$$\psi \leftarrow \arg\min_{\psi} \mathcal{L}(\widetilde{\mathcal{D}}_K^{Train}; \Theta_{\text{PLM}}, \psi) \tag{4.2}$$

4.3.3 Re-weighting Noisy Prompt Labels

Consider $\{\hat{y}_n^{(t)}\}_{n=1}^N$ to be the pseudo prompt-labels (for the masked tokens in $\widetilde{x}_n^u \in \widetilde{X}$) from the teacher $(\Theta_{\text{PLM}}, \hat{\psi}_{\text{tea}})$ in the t-th iteration where N is the number of unlabeled instances and $\hat{\psi}_{\text{tea}}$ represent the teacher adapter parameters. In self-training, the student model is trained to mimic the teacher predictions on the transfer set. Consider $\mathcal{L}(\hat{y}_n^{(t)}, \text{enc}(\widetilde{x}_n^u; \Theta_{\text{PLM}}, \psi_{\text{stu}}^{(t)}))$ to be the loss of the student model with parameters $(\Theta_{\text{PLM}}, \psi_{\text{stu}}^{(t)})$ on the pseudo-labeled data in the t-th iteration, where Θ_{PLM} and ψ_{stu} represent the PLM and the student adapter parameters respectively. The student update (with step size α) can be formalized as:

$$\hat{\psi}_{\text{stu}}^{(t)} = \hat{\psi}_{\text{stu}}^{(t-1)} - \alpha \nabla \left(\frac{1}{N} \sum_{\text{i}=1}^{N} \mathcal{L}(\hat{y}_{\text{i}}^{(t)}, \text{enc}(x_{\text{i}}^u; \Theta_{\text{PLM}}, \hat{\psi}_{\text{stu}}^{(t-1)})\right). \tag{4.3}$$

In order to reduce error propagation from noisy pseudo-labels, we leverage meta-learning to re-weight them based on the student model loss on the validation set as our meta-objective. The intuition of meta re-weighting is to measure the impact or weight of a pseudo-labeled example given by its performance on the **validation set ($\widetilde{\mathcal{D}}_K^{Train}$ in our work)**. To this end, we leverage the idea

of weight perturbation [43] to set the weight of pseudo-labeled example $(\widetilde{x}_{\mathrm{i}}^{u}, \hat{y}_{\mathrm{i}}^{(t)})$ to $\epsilon_{\mathrm{i}}^{(t)}$ at iteration t as:

$$\mathcal{L}_{r}^{(t)}(\epsilon, \psi) = \frac{\sum_{\mathrm{i}=1}^{N} [\epsilon_{\mathrm{i}}^{(t)} \cdot \mathcal{L}(\hat{y}_{\mathrm{i}}^{(t)}, \mathrm{enc}(\widetilde{x}_{\mathrm{i}}^{u}; \Theta_{\mathrm{PLM}}, \hat{\psi}_{\mathrm{stu}}^{(t-1)}))]}{N}. \tag{4.4}$$

$$\hat{\psi}_{\mathrm{stu}}^{(t)}(\epsilon) = \hat{\psi}_{\mathrm{stu}}^{(t-1)} - \alpha \nabla \mathcal{L}_{r}^{(t)}(\epsilon, \psi). \tag{4.5}$$

Weight perturbation is used to discover data points that are most important to improve performance on the validation set. Optimal value for the perturbation $\epsilon_{\mathrm{i}}^{(t)*}$ can be obtained via minimizing student model loss on the validation set at iteration t as:

$$\epsilon_{\mathrm{i}}^{(t)*} = \arg\min_{\epsilon_{\mathrm{i}}} \frac{\sum_{\mathrm{i}=1}^{|\widetilde{\mathcal{D}}_{K}^{Train}|} \mathcal{L}(y_{\mathrm{i}}, \mathrm{enc}(x_{\mathrm{i}}; \Theta_{\mathrm{PLM}}, \hat{\psi}_{stu}^{(t)}(\epsilon_{\mathrm{i}}))}{|\widetilde{\mathcal{D}}_{K}^{Train}|} \tag{4.6}$$

To obtain a cheap estimate of the meta-weight at step t, we take a single gradient descent step on a mini-batch $\widetilde{\mathcal{D}}^{(t)} \in \widetilde{\mathcal{D}}_{K}^{Train}$ as:

$$u_{\mathrm{i}}^{(t)} = -\frac{\partial}{\partial \epsilon_{\mathrm{i}}} \left(\frac{\sum_{\mathrm{i}=1}^{|\widetilde{\mathcal{D}}^{(t)}|} \mathcal{L}(y_{\mathrm{i}}, \mathrm{enc}(\widetilde{x}_{\mathrm{i}}; \Theta_{\mathrm{PLM}}, \hat{\psi}_{stu}^{(t)}(\epsilon)))}{|\widetilde{\mathcal{D}}^{(t)}|} \right) \tag{4.7}$$

The weight $w_{\mathrm{i}}^{(t)}$ of $(\widetilde{x}_{\mathrm{i}}^{u}, \hat{y}_{\mathrm{i}}^{(t)})$ at iteration t is set to be proportional to the negative gradient $u_{\mathrm{i}}^{(t)}$ to reflect the importance of pseudo-labeled samples. Samples with negative weights are filtered out since they could potentially degrade the student performance. Finally, we update student adapter parameters ψ_{stu} while accounting for re-weighting as:

$$\mathcal{L}^{(t)} = \frac{1}{N} \sum_{\mathrm{i}=1}^{N} [w_{\mathrm{i}}^{(t)} \cdot \mathcal{L}(\hat{y}_{\mathrm{i}}^{(t)}, \mathrm{enc}(\widetilde{x}_{\mathrm{i}}^{u}; \Theta_{\mathrm{PLM}}, \hat{\psi}_{\mathrm{stu}}^{(t-1)}))]). \tag{4.8}$$

Knowledge Distillation For Student Warmup. Meta re-weighting leverages gradient as a proxy to estimate the weight of noisy pseudo labels. However, the gradients of adapter parameters ψ are not stable in the early stages of training due to random initialization and noises in pseudo labels. This instability issue is further exacerbated with adapter tuning that usually requires a larger learning rate [75]. Therefore, to stabilize adapter tuning, we propose a warmup training stage via knowledge distillation [67] to first tune adapter parameters via knowledge distillation loss for T_{warm} steps and then we continue self-training with re-weighted updates via Eq. 4.8. Since the re-weighting procedure requires held-out validation set (few-shot training examples in our setting),

we do not use labeled data in knowledge distillation while using only the consistency loss between teacher model $(\Theta_{\text{PLM}}, \hat{\psi}_{\text{tea}})$ and student model $(\Theta_{\text{PLM}}, \hat{\psi}_{\text{stu}})$ on unlabeled data as.

$$\arg\min_{\hat{\psi}_{stu}} \text{KL}(f(\widetilde{x}^u; \Theta_{\text{PLM}}, \hat{\psi}_{\text{tea}}) \mid\mid f(\widetilde{x}^u; \Theta_{\text{PLM}}, \hat{\psi}_{\text{stu}})). \tag{4.9}$$

We further validate the effectiveness of knowledge distillation for warmup with ablation analysis.

Student Adapter Re-initialization A typical challenge in few-shot settings is the lack of a separate validation set. In the spirit of *true* few-shot learning, we use only the available few-shot labeled examples $\widetilde{\mathcal{D}}_K^{Train}$ as the validation set for meta-learning of the student model. This poses an interesting challenge of preventing label leakage. To address this issue, we *re-initialize the student adapter parameters* every time at the start of each self-training iteration to mitigate interference with labeled data. Note that the student and teacher model share the encoder parameters Θ_{PLM} that are always kept frozen and not updated during training.

4.4 Experiments

4.4.1 Experimental Setup

Dataset. We perform large-scale experiments with six natural language understanding tasks. We use four tasks from GLUE [76], including MNLI [77] for natural language inference, RTE [78]–[81] for textual entailment, QQP[1] for semantic equivalence and SST-2 [82] for sentiment classification. The results are reported on their development set following [83]. MPQA [84] and Subj [85] are used for polarity and subjectivity detection, where we follow [66] to keep $2,000$ examples for testing and use remaining examples for semi-supervised learning.

For each dataset, we randomly sample $|\mathcal{K}| \in \{10, 20, 30\}$ manually labeled samples from the training data, and add the remaining to the unlabeled set while ignoring their labels – following standard setups for semi-supervised learning. **We repeatedly sample K labeled instances five times, run each model with 5 different seeds and report average performance with standard deviation across the runs.** For the average accuracy over 6 tasks, we did not include standard deviation across tasks. Furthermore, for every split and shot, we sample the labeled data such that $\mathcal{D}_{10}^{Train} \subset \mathcal{D}_{20}^{Train} \subset \mathcal{D}_{30}^{Train}$ to evaluate the impact of incremental sample injection.

[1] ↑https://www.quora.com/q/quoradata/

Following *true few-shot learning* setting [86], we do not use additional *development set* beyond $|\mathcal{K}|$ labeled samples for any hyper-parameter tuning or early stopping. The performance of each model is reported after fixed training epochs.

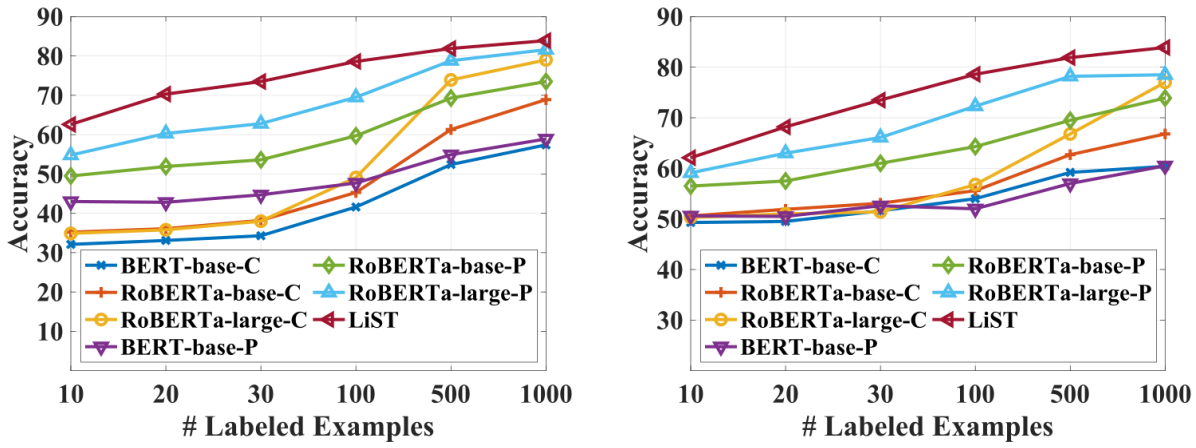
Baselines. In addition to classic-tuning (Classic FN), we adopt prompt-based fine-tuning (Prompt FN) from [66] as labeled-only baselines. We also adopt several state-of-the-art semi-supervised baselines including UST [36], MetaST [87] and iPET [88]. UST and MetaST are two self-training methods which are based on classic fine-tuning strategies. iPET is a semi-supervised method leveraging prompt-based fine-tuning and prompt ensembles to obtain state-of-the-art performance. While iPET ensembles multiple fully-tuned models, we develop a lite self-training framework to achieve both data and parameter efficiency. As the strongest semi-supervised baseline, **we implement a new method PromptST based on self-training using prompts and adapters (as a subset of the methods used in** `LiST`**)**, but without any re-weighting, or KD warmup that are additionally used in `LiST`. The methods Prompt FN, PromptST and `LiST` adopt same prompts and label words as in [66]. We implement our framework in Pytorch and use Tesla V100 gpus for experiments.

Table 4.1. Performance comparison of different tuning strategies on different tasks with RoBERTa-large as the encoder with standard deviation in parantheses. UST, MetaST, PromptST and iPET are semi-supervised methods using unlabeled data, whereas Classic and Prompt FN only use labeled data. GPT-3 [64].

Labels	**Models**	**Avg**	**#Tunable Params**	**MNLI (m/mm)** (acc)	**RTE** (acc)	**QQP** (acc)	**SST-2** (acc)	**Subj** (acc)	**MPQA** (acc)
$\|\mathcal{K}\| = 30$	Classic FN	60.9	355M	38.0 (1.7) / 39.0 (3.1)	51.4 (3.7)	64.3 (8.1)	65.0 (11.5)	90.2 (2.2)	56.1 (5.3)
	Prompt FN	77.6	355M	62.8 (2.6) / 64.1 (3.3)	66.1 (2.2)	71.1 (1.5)	91.5 (1.0)	91.0 (0.5)	82.7 (3.8)
$\|\mathcal{K}\| = 30$ +Unlabeled Data	UST	65.8	355M	40.5 (3.3) / 41.5 (2.9)	53.4 (1.7)	61.8 (4.3)	76.2 (11.4)	91.5 (2.1)	70.9 (6.2)
	MetaST	62.6	355M	39.4 (3.9) / 40.5 (4.4)	52.9 (2.0)	65.7 (6.2)	65.3 (15.2)	91.4 (2.3)	60.5 (3.6)
	iPET	75.5	355M	61.0 (5.8) / 61.8 (4.7)	54.7 (2.8)	67.3 (4.1)	**93.8 (0.6)**	92.6 (1.5)	83.1 (4.8)
	PromptST	77.2	14M	61.8 (1.9) / 63.1 (2.9)	66.2 (5.1)	71.4 (2.1)	91.1 (1.4)	90.3 (1.5)	81.8 (2.5)
	`LiST`	**82.0**	14M	**73.5 (2.8) / 75.0 (3.7)**	**71.0 (2.4)**	**75.2 (0.9)**	92.8 (0.9)	**93.5 (2.2)**	**85.2 (2.1)**
Supervision with # Full Train	Classic FN	90.9	355M	89.6 / 89.5	83.0	91.8	95.2	97.2	88.8
	Prompt FN	92.0	355M	89.3 / 88.8	88.4	92.1	95.9	97.1	89.3

4.4.2 Key Results

Table 4.1 shows the performance comparison among different models with $|\mathcal{K}| = 30$ labeled examples with fixing RoBERTa-large as the encoder. Fully-supervised RoBERTa-large trained on thousands of labeled examples provides the ceiling performance for the few-shot setting. We observe LiST to significantly outperform other state-of-the-art baselines along with 96% reduction in tunable parameters, achieving both labeled data- and parameter-efficiency. More specifically, LiST improves over Classic FN, Prompt FN, iPET and PromptST by 34.6%, 5.7%, 8.6% and 6.2% respectively in terms of average performance on six tasks. This demonstrates the impact of self-training with unlabeled data and prompt-based FN. Additionally, iPET and LiST both leverage prompt-based FN to significantly improve over UST and MetaST that use classic fine-tuning strategies, confirming the effectiveness of prompt-based FN in the low data regime. iPET ensembles multiple prompts with diverse qualities and under-performs Prompt FN on average in our few-shot setting without using any development set.



(a) MNLI (b) RTE

Figure 4.5. Performance comparison of Classic-tuning (denoted as "C") and prompt-based fine-tuning (denoted as "P") with LiST on MNLI and RTE using language model encoders of different sizes.

Figure 4.5 compares the performance of tuning methods with varying number of training labels and encoders of different sizes. We observe that large models are more data-efficient compared to smaller models. However, large fully-tunable models are expensive to use in practise. We observe

Table 4.2. Average accuracy on tuning different modules of RoBERTa-large with $|\mathcal{K}| = 30$ labels on **six tasks**. Diff shows performance change relative to Full tuning.

Tuning	#Params	Avg	Diff
Full	355M	77.6	—
Embedding	53M	67.0	-10.7
Attention	101M	77.0	-0.6
FF-output	102M	**77.6**	+0.0
FF-intermediate	102M	75.9	-1.7

that LiST with small number of tunable parameters consistently outperforms fully-tunable classic and prompt-based FN strategies in all labeled data settings, demonstrating both data and parameter efficiency. Additional results with different backbone encoders and varying number of shots and fine-tuning strategies are presented in the .

4.4.3 Adapter Analysis

In this section, we explore adapter design choices for prompt-based FN with RoBERTa-large as encoder *using only few-shot labeled data*.

Where to insert an adapter in Transformers? In order to answer this question, we conduct an experiment to study the role of various Transformer modules in few-shot prompt-based FN. To this end, we tune a given module along with the language model head while keeping all other parameters frozen. Table 4.2 shows the performance comparison of tuning specific modules on six tasks with varying number of labeled examples. The main modules of RoBERTa include *Embedding*, *Attention*, *Feedforward Output* and *Feedforward Intermediate* layers. We observe that tuning only the *Feedforward Output* or the *Attention* module delivers the best performance across most tasks with few-shot labels. Correspondingly, this motivated us to insert our adapter parameters into these two modules..

Comparison with other lightweight parameter efficient model tuning strategies. To validate the effectiveness of LiST adapters, we compare it against several baselines in Table 4.3. For

[1]The average accuracy of GPT-3 in-context learning with 30 labeled examples over 6 tasks is 61.5.

Table 4.3. Average accuracy of several lightweight parameter-efficient tuning strategies with $|\mathcal{K}| = 30$ labels without unlabeled data on **six tasks** along with the number ($\#$) of tunable parameters. Each task is run with 5 different seeds. `LiST` Adapter performance with different bottleneck dimension d of its adapters is shown in parentheses.

Tuning	**#Params**	**Avg**
Head-only	1M	66.9
Bias-only [73]	1M	68.3
Prompt-tuning [89]	1M	56.4
`LiST` Adapter (2)	1M	72.7
Houlsby Adapter [72]	14M	57.9
`LiST` Adapter (128)	14M	**77.7**
Full tuning	355M	77.6

a fair comparison, we present two variations of our `LiST` adapters with bottleneck dimensions d= $\{2, 128\}$ corresponding to $1M$ and $14M$ parameters to match other adapter capacities; all the approaches in Table 4.3 are *trained with 30 labels only without unlabeled data* for a fair comparison. (1) Bias-only is a simple but effective lightweight method, which tunes bias terms of PLMs while keeping other parameters frozen. (2) Tuning head layers is widely used as a strong baseline for lightweight studies [72], where we tune last two layers including language model head while freezing other parameters. (3) prompt-tuning is a lightweight method which only updates task prompt embedding while keeping entire model frozen. (4) Houlsby Adapter tunes inserted adapter parameters keeping the encoder frozen by adopting classic tuning strategy. Besides these lightweight methods, we also present a performance comparison with full model tuning as a strong baseline.

Table 4.3 shows that `LiST` is able to match the performance of full model prompt-based FN with bottleneck dimension $d = 128$ and outperforms all other baselines with similar capacities. While lightweight model tuning choices like tuning the bias or inserting adapters into classic tuning models are shown to be effective in fully-supervised settings [72], [73], we observe them to under-perform for few-shot learning. We observe that simpler tuning choices like Head-only and Bias-only results in upto 10% performance degradation. Houlsby adapter and Prompt-only results in upto 20% performance degradation. In constrast, `LiST` adapter is able to match the performance

Table 4.4. Ablation analysis of LiST with 30 labels on MNLI and RTE with tunable parameters in parantheses.

Method	Avg Acc	Avg Std	Datasets	
			MNLI (m/mm)	RTE
LiST ($14M$)	72.6	2.8	73.5 (2.8) / 75.0 (3.7)	71.0 (2.4)
w/o re-init	68.3	4.2	66.7 (2.8) / 68.3 (4.3)	69.0 (4.9)
w/o KD Warmup	68.8	8.8	67.9 (12.9) / 69.0 (13.1)	69.2 (4.5)
w/o Re-weighting	71.6	4.0	72.9 (3.4) / 74.2 (4.5)	69.7 (4.1)
w/ Hard Pseudo-Labels	70.9	4.4	71.7 (3.8) / 73.0 (5.4)	69.5 (4.2)
LiST w/o Adapter ($355M$)	72.6	2.5	73.6 (2.7) / 74.8 (2.7)	71.2 (2.3)

of full tuning in few-shot setting, demonstrating the importance of adapter placement choices and parameter initialization.

4.4.4 Ablation Analysis

Table 4.4 demonstrates the impact of different components and design choices of LiST.

• **Adapter training stability.** Training with very few labels and noisy pseudo labeled data results in instability for adapter tuning. To demonstrate training stability, we include the average accuracy and standard deviation across several runs and splits as metrics. We observe that hard pseudo-labels hurt the model performance compared to soft pseudo-labels and exacerbate the instability issue. This is in contrast to observations from classic fine-tuning [87]. A potential reason could be that the well pre-trained language model head for prompt-based FN is able to capture better associations among different prompt labels.

• **Knowledge Distillation Warmup.** In this ablation study, we remove the warmup phase with knowledge distillation from LiST (denoted as "LiST w/o KD Warmup"). Removing this component results in 4% performance drop in terms of average accuracy and 300% larger standard deviation – demonstrating the importance of KD Warmup in stabilizing LiST training.

• LiST **versus** LiST **w/o Adapter.** In LiST, we only fine-tune the adapter and language model head while keeping other encoder parameters frozen to achieve parameter efficiency. Table 4.4 shows that LiST using only 4% tunable parameters is able to match the performance of fully tunable LiST (that is without using any adapters and tuning all encoder parameters) on MNLI and RTE – demonstrating the effectiveness of our lightweight design.

Part II

DOMAIN ADAPTATION

5. DOMAIN-ADVERSARIAL FAKE NEWS DETECTION

(A version of this chapter has been previously published in KDD 2018 [7].)

5.1 Introduction

The recent proliferation of social media has significantly changed the way in which people acquire information. Nowadays, there are increasingly more people consuming news through social media, which can provide timely and comprehensive multimedia information on the events taking place all over the world. Compared with traditional text news, the news with images and videos can provide a better storytelling and attract more attention from readers. Unfortunately, this is also taken advantage by fake news which usually contain misrepresented or even forged images, to mislead the readers and get rapid dissemination. The dissemination of fake news may cause large-scale negative effects, and sometimes can affect or even manipulate important public events. For example, within the final three months of the 2016 U.S. presidential election, the fake news generated to favor either of the two nominees was believed by many people and was shared by more than 37 million times on Facebook [90], [91]. Therefore, it is in great need of an automatic detector to mitigate the serious negative effects caused by the fake news.

Thus far, various fake news detection approaches, including both traditional learning [92]–[94] and deep learning based models [95], [96], have been exploited to identify fake news. With sufficient verified posts on different events, existing deep learning models have achieved performance improvement over traditional ones due to their superior ability of feature extraction. However, they are still not able to handle the unique challenge of fake news detection, i.e., detecting fake news on newly emerged and time-critical events [97]. Due to lack of the corresponding prior knowledge, the verified posts about such events can be hardly obtained in a timely manner, which leads to the unsatisfactory performance of existing models. Actually, existing models tend to capture lots of event-specific features which are not shared among different events. Such event-specific features, though being able to help classify the posts on verified events, would hurt the detection with regard to newly emerged events. For this reason, instead of capturing event-specific features, we believe that *learning the shared features among all the events* would help us with the detection of fake news from unverified posts. Therefore, the goal of this work is to design an effective model to remove the

nontransferable event-specific features and preserve the shared features among all the events for the task of identifying fake news.

To remove event-specific features, the first step is to identify them. For posts on different events, they have their own unique or specific features that are not sharable. Such features can be detected by measuring the difference among posts corresponding to different events. Here the posts can be represented by the learned features. Thus, identifying event-specific features is equivalent to measuring the difference among learned features on different events. However, it is a technically challenging problem. First, since the learned feature representations of posts are high-dimensional, simple metrics like the squared error may not be able to estimate the differences among such complicated feature representations. Second, the feature representations keep changing during the training stage. This requires the proposed measurement mechanism to capture the changes of feature representations and consistently provide the accurate measurement. Although this is very challenging, the effective estimation of dissimilarities among the learned features on different events is the premise of removing event-specific features. Thus, how to effectively estimate the dissimilarities under this condition is the challenge that we have to address.

In order to address the aforementioned challenges, we propose an end-to-end framework referred to as Event Adversarial Neural Networks (EANN) for fake news detection based on multi-modal features. Inspired by the idea of adversarial networks [98], we incorporate the event discriminator to predict the event auxiliary labels during training stage, and the corresponding loss can be used to estimate the dissimilarities of feature representations among different events. The larger the loss, the lower the dissimilarities. Since the fake news takes advantage of multimedia content to mislead readers and gets spread, our model needs to handle the multi-modal inputs. The proposed model EANN consists of three main components: the multi-modal feature extractor, the fake news detector, and the event discriminator. tW For multi-modal feature extractor, We employ Convolutional Neural Networks (CNN) to automatically extract features from both textual and visual content of posts. Experimental results on two large scale real-world social media datasets show that the proposed EANN model outperforms the state-of-the-art approaches.

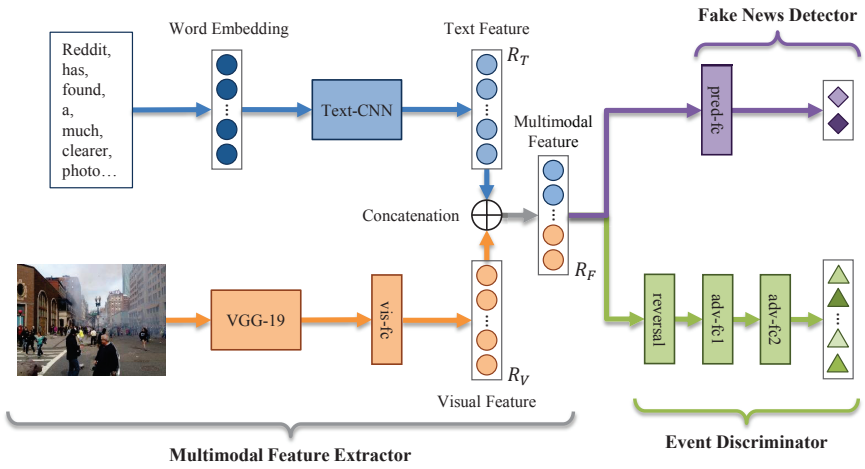


Figure 5.1. The architecture of Event Adversarial Neural Networks (EANN). The blue colored network is the textual feature extractor, the orange colored network is visual feature extractor, the fake news detector is purple colored, and event discriminator is green colored.

5.2 Methodology

In this section, we first introduce the three components of the proposed EANN model: the multimodal feature extractor, the fake news detector, and the event discriminator, then describe how to integrate these three components to learn the transferable feature representations. The detailed algorithm flow is also shown in the last subsection.

5.2.1 Model Overview

The goal of our model is to learn the transferable and discriminable feature representations for fake news detection. As shown in Figure 5.1, in order to achieve this, the proposed EANN model integrates three major components: the multi-modal feature extractor, the fake news detector, and the event discriminator. First of all, since the posts on social media usually contain information in different modalities (e.g., textual post and attached image), the multi-modal feature extractor includes both textual and visual feature extractors to handle different types of inputs. After the textual and visual latent feature representations are learned, they are concatenated together to form the final multi-modal feature representation. Both of the fake news detector and the event discriminator are built on top of the multi-modal feature extractor. The fake news detector takes

the learned feature representation as input to predict whether the posts are fake or real. The event discriminator identifies the event label of each post based on this latent representation.

5.2.2 Multi-Modal Feature Extractor

Textual Feature Extractor

The sequential list of the words in the posts is the input to the textual feature extractor. In order to extract the informative features from textual content, we employ convolutional neural networks (CNN) as the core module of our textual feature extractor. CNN has been proven to be effective in many fields such as computer vision and text classification [99], [100]. As can be seen in Figure 5.1, we incorporate a modified CNN model, namely Text-CNN [101], in our textual feature extractor. The architecture of Text-CNN is shown in Figure 5.2. As seen, it takes advantage of multiple filters with various window sizes to capture different granularities of features to identify fake news.

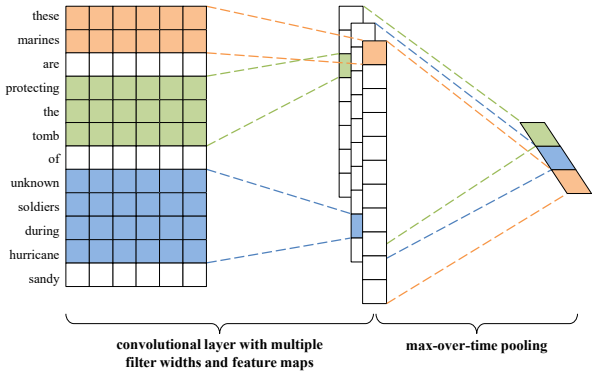


Figure 5.2. The architecture of Text-CNN.

For detailed procedures of the textual feature extractor, each word in the text is represented as a word embedding vector. The embedding vector for each word is initialized with the pre-trained word embedding on the given dataset. For the i-th word in the sentence, the corresponding k dimensional word embedding vector is denoted as $T_{\mathrm{i}} \in \mathbb{R}^k$. Thus, a sentence with n words can be represented as:

$$T_{1:n} = T_1 \oplus T_2 \oplus ... \oplus T_n, \tag{5.1}$$

where $\oplus$ is the concatenation operator. A convolutional filter with window size h takes the contiguous sequence of h words in the sentence as input and outputs one feature. In order to show the procedure clearly, we take the contiguous sequence of h words starting with the i-th word as example, the filter operation can be represented as:

$$t_{\mathrm{i}} = \sigma(W_c \cdot T_{\mathrm{i:i+h-1}}). \tag{5.2}$$

Here $\sigma(\cdot)$ is the ReLU activation function and W_c represents the weight of the filter. The filter can also be applied to the rest of words and then we get a feature vector for this sentence:

$$t = [t_1, t_2, ..., t_{n-h+1}]. \tag{5.3}$$

For every feature vector t, we use max-pooling operation to take the maximum value so as to extract the most important information. Now, we get the corresponding feature for one particular filter. The process is repeated until we get the features for all filters. In order to extract textual features with different granularities, various window sizes are applied. For a specific window size, we have n_h different filters. Thus, assuming there are c possible window sizes, we have $c \cdot n_h$ filters in total. The textual features after the max-pooling operation is written as $R_{T_c} \in \mathbb{R}^{c \cdot n_h}$. Following the max-pooling operations, a fully connected layer is used to ensure the final textual feature representation (denoted as $R_T \in \mathbb{R}^p$) has the same dimension (denoted as p) as the visual feature representation through the following operation:

$$R_T = \sigma(W_{tf} \cdot R_{T_c}), \tag{5.4}$$

where W_{tf} is the weight matrix of the fully connected layer.

Visual Feature Extractor

The attached images of the posts are inputs to the visual feature extractor and are denoted as V. In order to efficiently extract visual features, we employ the pre-trained VGG19 [102]. On top of the last layer of VGG19 network, we add a fully connected layer to adjust the dimension of final

visual feature representation to p. During the joint training process with the textual feature extractor, the parameters of pre-trained VGG19 neural network are kept static to avoid overfitting. Denoting p dimensional visual feature representation as $R_V \in \mathbb{R}^p$, the operation of the last layer in the visual feature extractor can be represented as:

$$R_V = \sigma(W_{vf} \cdot R_{V_{vgg}})), \tag{5.5}$$

where $R_{V_{vgg}}$ is the visual feature representation obtained from pre-trained VGG19, and W_{vf} is the weight matrix of the fully connected layer in the visual feature extractor.

The textual feature representation R_T and visual feature representation R_V will be concatenated to form the multi-modal feature representation denoted as $R_F = R_T \oplus R_V \in \mathbb{R}^{2p}$, which is the output of the multi-modal feature extractor. We denote the multi-modal feature extractor as $G_f(M; \theta_f)$ where M, which is usually a set of textual and visual posts, is the input to the multi-modal feature extractor, and θ_f represents the parameters to be learned.

5.2.3 Fake News Detector

In this subsection, we introduce the fake news detector. It deploys a fully connected layer with softmax to predict whether the posts are fake or real. The fake news detector is built on top of the multi-modal feature extractor, thus taking the multi-modal feature representation R_F as input. We denote the fake news detector as $G_d(\cdot\,; \theta_d)$, where θ_d represents all the parameters included. The output of the fake news detector for the i-th multimedia post, denoted as m_{i}, is the probability of this post being a fake one:

$$P_\theta(m_{\mathrm{i}}) = G_d(G_f(m_{\mathrm{i}}; \theta_f); \theta_d). \tag{5.6}$$

The goal of the fake news detector is to identify whether a specific post is fake news or not. We use Y_d to represent the set of labels and employ cross entropy to calculate the detection loss:

$$L_d(\theta_f, \theta_d) = -\mathbb{E}_{(m,y)\sim(M,Y_d)} \left[y \log(P_\theta(m)) + (1-y)(\log(1 - P_\theta(m))\right]. \tag{5.7}$$

We minimize the detection loss function $L_d(\theta_f, \theta_d)$ by seeking the optimal parameters $\hat{\theta}_f$ and $\hat{\theta}_d$, and this process can be represented as:

$$(\hat{\theta}_f, \hat{\theta}_d) = arg \min_{\theta_f, \theta_d} L_d(\theta_f, \theta_d). \tag{5.8}$$

As previously discussed, one of the major challenges for fake news detection stems from the events that are not covered by the training dataset. This requires us to be able to learn the transferable feature representations for newly emerged events. Direct minimization of detection loss only helps detect fake news on the events included in the training dataset, since this captures only event-specific knowledge (e.g., keywords) or patterns, which cannot generalize well. Thus, we need to enable the model to learn more general feature representations that can capture the common features among all the events. Such representation should be event-invariant and does not include any event-specific features. To achieve this goal, we need to remove the uniqueness of each event. In particular, we measure the dissimilarities of the feature representations among different events and remove them in order to capture the event invariant feature representations.

5.2.4 Event Discriminator

Event discriminator is a neural network which consists of two fully connected layers with corresponding activation functions. It aims to correctly classify the post into one of K events based on the multi-modal feature representations. We denote the event discriminator as $G_{\mathrm{e}}(R_F; \theta_{\mathrm{e}})$ where θ_{e} represents its parameters. We define the loss of event discriminator by cross entropy and use Y_{e} to represent the set of the event labels:

$$L_{\mathrm{e}}(\theta_f, \theta_{\mathrm{e}}) = -\mathbb{E}_{(m,y)\sim(M,Y_{\mathrm{e}})} [\sum_{k=1}^{K} \mathbf{1}_{[k=y]} \log(G_{\mathrm{e}}(G_f(m; \theta_f)); \theta_{\mathrm{e}})], \tag{5.9}$$

The parameters of event discriminator minimizing the loss $L_{\mathrm{e}}(\cdot, \cdot)$ are written as:

$$\hat{\theta}_{\mathrm{e}} = arg \min_{\theta_{\mathrm{e}}} L_{\mathrm{e}}(\theta_f, \theta_{\mathrm{e}}). \tag{5.10}$$

The above loss $L_{\mathrm{e}}(\theta_f, \hat{\theta}_{\mathrm{e}})$ can be used to estimate the dissimilarities of different events' distributions. The large loss means the distributions of different events' representations are similar and the learned features are event-invariant. Thus, in order to remove the uniqueness of each event, we need to maximize the discrimination loss $L_{\mathrm{e}}(\theta_f, \hat{\theta}_{\mathrm{e}})$ by seeking the optimal parameters θ_f.

The above idea motivates a minimax game between the multi-modal feature extractor and the event discriminator. On one hand, the multi-modal feature extractor tries to fool the event discriminator to maximize the discrimination loss, and on the other hand, the event discriminator aims to discover the event-specific information included in the feature representations to recognize the event. The integration process of three components and the final objective function will be introduced in the next subsection.

5.2.5 Model Integration

During the training stage, the multi-modal feature extractor $G_f(\cdot; \theta_f)$ needs to cooperate with fake news detector $G_d(\cdot; \theta_d)$ to minimize the detection loss $L_d(\theta_f, \theta_d)$, so as to improve the performance of fake news detection task. Simultaneously, the multi-modal feature extractor $G_f(\cdot; \theta_f)$ tries to fool the event discriminator $G_{\mathrm{e}}(\cdot; \hat{\theta}_{\mathrm{e}})$ to achieve event invariant representations by maximizing the event discrimination loss $L_{\mathrm{e}}(\theta_f, \theta_{\mathrm{e}})$. The event discriminator $G_{\mathrm{e}}(R_F; \theta_{\mathrm{e}})$ tries to recognize each event based on the multi-modal feature representations by minimizing the event discrimination loss. We can define the final loss of this three-player game as

$$L_{final}(\theta_f, \theta_d, \theta_{\mathrm{e}}) = L_d(\theta_f, \theta_d) - \lambda\, L_{\mathrm{e}}(\theta_f, \theta_{\mathrm{e}}), \tag{5.11}$$

$$\hat{\theta}_f = arg \min_{\theta_f} L_d(\theta_f, \theta_d) - \lambda\, L_{\mathrm{e}}(\theta_f, \theta_{\mathrm{e}}), \tag{5.12}$$

where λ controls the trade-off between the objective functions of fake news detection and event discrimination. In this work, we simply set λ as 1 without tuning the trade-off parameter. For the minimax game, the parameter set we seek is the saddle point of the final objective function:

$$(\hat{\theta}_f, \hat{\theta}_d) = arg \min_{\theta_f, \theta_d}\ L_{final}(\theta_f, \theta_d, \hat{\theta}_{\mathrm{e}}), \tag{5.13}$$

$$\hat{\theta}_{\mathrm{e}} = arg\max_{\theta_{\mathrm{e}}} L_{final}(\hat{\theta}_f, \theta_{\mathrm{e}}). \tag{5.14}$$

We use stochastic gradient descent to solve the above problem. The θ_f is updated according to Eq. 5.15. Here we adopt the gradient reversal layer (GRL) introduced in [103]. The gradient reversal layer acts as an identity function during forward stage, and it multiplies gradient with $-\lambda$ and passes the results to the preceding layer during backprop stage. GRL can be easily added between the multi-modal feature extractor and the event discriminator. We denote it as the reversal layer in the Figure 5.1.

$$\theta_f \leftarrow \theta_f - \eta\,(\frac{\partial L_d}{\partial \theta_f} - \lambda\frac{\partial L_{\mathrm{e}}}{\partial \theta_f}). \tag{5.15}$$

In order to stabilize the training process, we follow the approach in [103] to decay the learning rate η:

$$\eta' = \frac{\eta}{(1+\alpha\cdot p)^{\beta}}, \tag{5.16}$$

where $\alpha = 10$, $\beta = 0.75$, and p is linearly changing from 0 to 1 corresponding to the training progress.

5.3 Experiments

In this section, we first introduce two large social media datasets used in the experiments, then present the state-of-the-art fake news detection approaches, and finally analyze the performance of the proposed model.

5.3.1 Datasets

To fairly evaluate the performance of the proposed model, we conduct experiments on two real social media datasets, which are collected from Twitter and Weibo. Next, we provide the details of both datasets respectively.

Table 5.1. The Statistics of the Real-World Datasets.

	Twitter	Weibo
# of fake News	7898	4749
# of real News	6026	4779
# of images	514	9528

Twitter Dataset

The Twitter dataset is from MediaEval Verifying Multimedia Use benchmark [104], which is used for detecting fake content on Twitter. This dataset has two parts: the development set and test set. We use the development as training set and test set as testing set to keep the same data split scheme. The tweets in the Twitter dataset contain text content, attached image/video and additional social context information. In this work, we focus on detecting fake news by incorporating both text and image information. Thus, we remove the tweets without any text or image. For this two sets, there is no overlapping events among them. For model training on Twitter dataset, we adopt early stop strategy.

Weibo Dataset

The Weibo dataset is used in [105] for detecting fake news. In this dataset, the real news are collected from authoritative news sources of China, such as Xinhua News Agency. The fake news are crawled from May, 2012 to January, 2016 and verified by the official rumor debunking system of Weibo. This system encourages common users to report suspicious posts and examines suspicious posts by a committee of trusted users. According to the previous work [96], [106], this system also acts as the authoritative source for collecting rumor news. When preprocessing this dataset, we follow the same steps in the work [105]. We first remove the duplicated and low quality images to ensure the quality of entire dataset. Then we apply a single-pass clustering method [107] to discover newly emerged events from posts. Finally, we split the whole datasets into the training, validation, testing sets in a 7:1:2 ratio, and ensure that they do not contain any common event.

5.3.2 Baselines

To validate the effectiveness of the proposed model, we choose baselines from the following three categories: single modality models, multi-modal models, and the variant of the proposed model.

Single Modality Models

In the proposed model, we leverage both text and image information to detect fake news. For each modality, it can also be solely used to discover fake news. Thus, we proposed the following two simple baselines:

• **Text**. We use 32 dimensional pre-trained word-embedding weights of text content from all of posts to initialize the parameters of the embedding layer. Then CNN is used to extract the textual feature R_T for each post. Finally, an additional fully connected layer with softmax function is used to predict whether this post is fake or not. We use 20 filters with window size ranging from 1 to 4, and the hidden size of fully connected layer is 32.

• **Vis**. The input of **Vis** is an image. Pre-trained VGG-19 and a fully connected layer are used to extract the visual feature R_V. Then, R_V is fed into a fully connected layer to make prediction. We set the hidden size of fully connected layer as 32.

Multi-modal Models

All the Multi-modal approaches take the information from multiple modalities into account, including VQA [108], NeuralTalk [109] and att-RNN [105].

• **VQA** [108]. Visual Question Answering (**VQA**) model aims to answer the questions based on the given images. The original VQA model is designed for multi-class classification tasks. In this work, we focus on binary classification. Thus, when implementing VQA model, we replace the final multi-class layer with the binary-class layer. Besides, for fair comparison, we use one-layer LSTM, and the hidden size of LSTM is 32.

• **NeuralTalk** [109]. NeuralTalk is a model to generate captions for the given images. The latent representations are obtained by averaging the outputs of RNN at each timestep, and then these representations are fed into a fully connected layer to make prediction. The hidden size of both LSTM and the fully connected layer is 32.

• **att-RNN** [105]. att-RNN is the state-of-the-art model for multi-modal fake news detection. It uses attention mechanism to fuse the textual, visual and social context features. In our experiments, we remove the part dealing with social context information, but the remaining parts are the same. The parameter settings are the same as [105].

A Variant of the Proposed EANN

The complete EANN model consists of three components: multi-modal feature extractor, fake news detector and event discriminator. Only using multi-modal feature extractor and fake news detector, we still can detect fake news. Thus, we design a variant of the proposed model, named **EANN**−. In EANN−, we do not include the event discriminator.

5.3.3 Implementation Details

In the textual feature extractor, we set $k = 32$ for dimensions of word-embedding. We set $n_h = 20$, and the window size of filters varies from 1 to 4 in Text-CNN. The hidden size of the fully connected layer in textual and visual extractor is 32. For fake news detector, the hidden size of the fully connected layer is 64. The event discriminator consists of two fully connected layers: the hidden size of first layer is 64, and the hidden size of second layer is 32. For all the baselines and the proposed model, we use the same batch size of 100 instances in the training stages, and the training epoch is 100.

5.3.4 Performance Comparison

Table 5.2 shows the experimental results of baselines and the proposed approaches on two datasets. We can observe that the overall performance of the proposed EANN is much better than the baselines in terms of *accuracy*, *precision* and *F1 score*.

On the Twitter dataset, the number of tweets on different events is imbalanced and more than 70% of tweets are related to a single event. This causes the learned text features mainly focus on some specific events. Compared with visual modality, the text modality contains more obvious event specific features which seriously prevents extracting transferable features among different events for the Text model. Thus, the accuracy of Text is the lowest among all the approaches. As for another single modality baseline Vis, its performance is much better than that of Text. The features

Table 5.2. The results of different methods on two datasets.

Dataset	Method	Accuracy	Precision	Recall	F_1
Twitter	Text	0.532	0.598	0.541	0.568
	Vis	0.596	0.695	0.518	0.593
	VQA	0.631	0.765	0.509	0.611
	NeuralTalk	0.610	0.728	0.504	0.595
	att-RNN	0.664	0.749	0.615	0.676
	EANN−	0.648	0.810	0.498	0.617
	EANN	**0.715**	**0.822**	**0.638**	**0.719**
Weibo	Text	0.763	0.827	0.683	0.748
	Vis	0.615	0.615	0.677	0.645
	VQA	0.773	0.780	0.782	0.781
	NeuralTalk	0.717	0.683	**0.843**	0.754
	att-RNN	0.779	0.778	0.799	0.789
	EANN−	0.795	0.806	0.795	0.800
	EANN	**0.827**	**0.847**	0.812	**0.829**

of image are more transferable, and thus reduce the effect of imbalanced posts. With the help of VGG19, a powerful tool for extracting useful features, we can capture the more sharable patterns contained in images to tell the realness of news compared with textual modality.

Though the visual modality is effective for fake news detection, the performance of Vis is still worse than that of the multi-modal approaches. This confirms that integrating multiple modalities is superior for the task of fake news detection. Among multi-modal models, att-RNN performs better than VQA and NeuralTalk, which shows that applying attention mechanism can help improve the performance of the predictive model.

For the variant of the proposed model EANN−, it does not include the event discriminator, and thus tends to capture the event-specific features. This would lead to the failure of learning enough shared features among events. In contrast, with the help of the event discriminator, the complete EANN significantly improves the performance in terms of all the measures. This demonstrates the effectiveness of the event discriminator for performance improvements. Specifically, the accuracy of EANN improves 10.3% compared with the best baseline att-RNN, and F1 scores increases 16.5%.

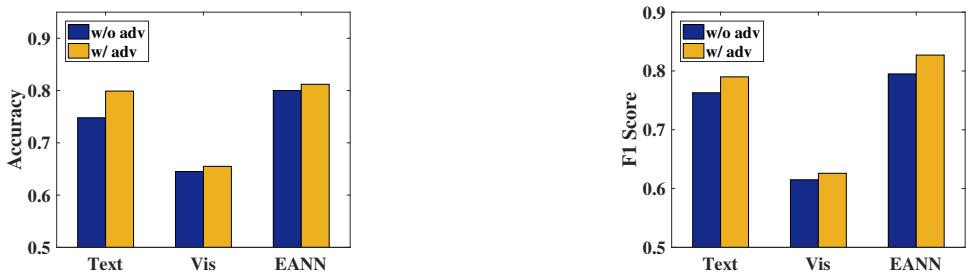


Figure 5.3. The performance comparison for the models w/ and w/o adversary.

On the Weibo dataset, similar results can be observed as those on the Twitter dataset. For single modality approaches, however, contradictory results are observed. From Table 5.2, we can see that the performance of Text is greatly higher than that of Vis. The reason is that the Weibo dataset does not have the same imbalanced issue as the Twitter dataset, and with sufficient data diversity, useful linguistic patterns can be extracted for fake news detection. This leads to learning a discriminable representation on the Weibo dataset for the textual modality. On the other hand, the images in the Weibo dataset are much more complicated in semantic meaning than those in the Twitter dataset. With such challenging image dataset, the baseline Vis cannot learn meaningful representations, though it uses the effective visual extractor VGG19 to generate feature representations.

As can be seen, the variant of the proposed model EANN- outperforms all the multi-modal approaches on the Weibo dataset. When modeling the text information, our model employs convolutional neural networks with multiple filters and different word window sizes. Since the length of each post is relatively short (smaller than 140 characters), CNN may capture more local representative features.

For the proposed EANN, it outperforms all the approaches on accuracy, precision and F1 score. Compared with EANN−, we can conclude that using the event discriminator component indeed improves the performance of fake news detection.

5.3.5 Event Discriminator Analysis

In this subsection, we aim to analyze the importance of the designed event discriminator component from the quantitative and qualitative perspectives.

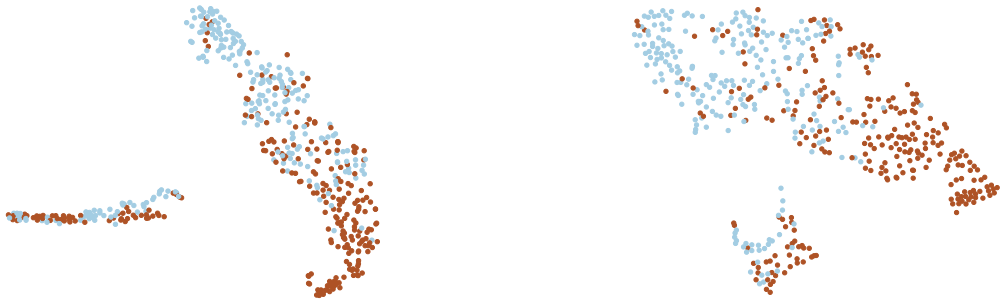
Quantitative Analysis

To intuitively illustrate the importance of employing event discriminator in the proposed model, we conduct the following experiments. For each single modality approach, we design its corresponding adversarial model. Then we run the new designed model on the Weibo dataset. Figure 5.3 shows the results in terms of F1 score and accuracy. In Figure 5.3, "w/ adv" means that we add event discriminator into the corresponding approaches, and "w/o adv" denotes the original approaches. For the sake of simplicity, let Text+ and Vis+ represent the corresponding approaches, Text and Vis, with event discriminator component being added, respectively.

From Figure 5.3, we can observe that both accuracy and F1 score of Text+ and Vis+ are greater than those of Text and Vis respectively. Note that for the proposed approach EANN, its reduced model is EANN−. The comparison between EANN and EANN− has been discussed in Section 5.3.4. Thus, we can draw a conclusion that incorporating event discriminator component is essential and effective for the task of fake news detection.

Qualitative Analysis

To further analyze the effectiveness of event discriminator, we qualitatively visualize the text features R_T learned by EANN− and EANN on the Weibo testing set with t-SNE [110] shown in Figure 5.4. The label for each post is real or fake.



(a) EANN− (b) EANN

Figure 5.4. Visualizations of learned latent text feature representations on the testing data of Weibo.

From Figure 5.4, we can observe that for the approach EANN−, it can learn discriminable features , but the learned features are still twisted together, especially for the left part of Figure 5.4a. In contrast, the feature representations learned by the proposed model EANN are more discriminable,

and there are bigger segregated areas among samples with different labels shown in Figure 5.4b. This is because in the training stage, event discriminator tries to remove the dependencies between feature representations and specific events. With the help of the minimax game, the muli-modal feature extractor can learn invariant feature representations for different events and obtain more powerful transfer ability for detection of fake news on new events. The comparison between EANN− and EANN proves that the proposed approach learns better feature representations with the component of event discriminator, and thus achieves better performance.

5.3.6 Case Studies for Multiple Modalities

In order to illustrate the importance of considering multi-modal features for fake news detection, we compare the results reported by the proposed EANN and single modality feature models (Text and Vis), and report the fake tweets correctly detected by EANN but missed by the single modality feature models.

(a) Five headed snake

(b) Photo: Lenticular clouds over Mount Fuji, Japan. #amazing #earth #clouds #mountains

Figure 5.5. Some fake news detected by EANN but missed by single text modality model on the Twitter dataset.

We first show two top-confident tweets which are successfully detected by the proposed model but missed by single textual modality model in Figure 5.5. The text content do not show evidence to identify that the tweets are fake. For both of the examples in Figure 5.5, they describe the images with common patterns. The textual modality model Text also identifies this news as a real one. Although the experts may be engaged to verify the text content using their domain knowledge, this option may not be available for normal readers. As seen, the two attached images look quite suspicious and are very likely to be forged pictures. By feeding visual content and textual content into the proposed EANN, both tweets are classified as fake with high confidence scores. This

shows that the proposed model EANN obtains some clues from the attached images to make correct classification. The additional visual content provides more information for fake news detection beyond single textual modality.

(a) Want to help these unfortunates? New, Iphones, laptops, jewelry and designer clothing could aid them through this!

(b) Meet The Woman Who Has Given Birth To 14 Children From 14 Different Fathers!

Figure 5.6. Some fake news detected by EANN but missed by single image modality model on the Twitter dataset.

Figure 5.6 shows another two examples missed by image modality model Vis but successfully spotted by the proposed EANN model. For the first example, the complicated semantic meaning is contained in the attached image, which is challenging to be captured by the visual feature extractor. However, the words with strong emotion and inflammatory intention suggest this is a suspicious post. By combining textual and visual content of tweets, the proposed EANN can easily detect that this is fake news with high confidence. The attached image in the second example looks very normal, but the corresponding textual description seems to misrepresent the image and mislead the readers. Without the textual content, the meaning of the tweets would totally change. Only aligned with the corresponding text description, it can be identified as fake news. The visual modality model Vis does not classify this example as false, but with the help of multi-modal features, the proposed EANN model gives the high confidence in detecting this fake news.

5.3.7 Convergence Analysis

In order to explore the training process of the proposed EANN model, the development of training, testing and discrimination loss (adversarial losses) has been shown in Figure 5.7. At the beginning, all of the three losses decrease. Then the discrimination loss increases and stabilizes at a certain level. The decreasing discrimination loss in the beginning represents the event discriminator detecting the event specific information included in the feature representations of multi-modal

feature extractor. As the minimax game between the discriminator and the feature extractor is continuing, the feature representations tend to be event invariant. Thus, the event specific information is removed incrementally, and the discrimination loss increases over the time. During the training process, the three losses smoothly converge, which means that a certain level of equilibrium have been achieved. As the training loss decreases steadily, we can observe that the testing loss also decreases steadily, and a very similar pattern of trend is shown. This observation proves that the feature representations learned by the proposed EANN can capture the general information among all the events, and this representation is also discriminative even on new coming events.

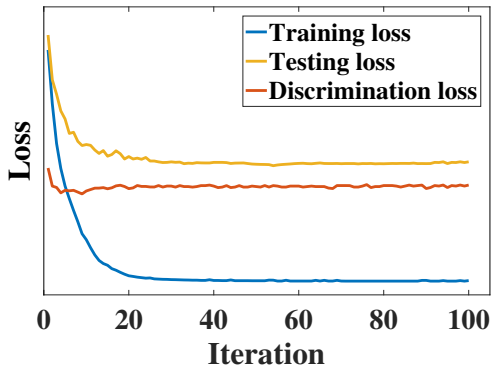


Figure 5.7. The training, testing and event discrimination loss development.

6. WEAKLY-SUPERVISED FAKE NEWS DETECTION

(A version of this chapter has been previously published in AAAI 2020 [8].)

6.1 Introduction

The recent proliferation of social media has significantly changed the way in which people acquire information. According to the 2018 Pew Research Center survey, about two-thirds of American adults (68%) get news on social media at least occasionally. Fake news, which refer to intentionally and verifiably false news stories, can spread virally on social media platforms as people rarely verify the source of the news when sharing a news article that sounds true. The spread of fake news may bring many negative impacts, including social panic and financial loss. Recent years have witnessed a number of high-impact fake news spread regarding terrorist plots and attacks, presidential election, and various natural disasters. In many of these cases, even when correct information later disseminates, the rapid spread of fake news can have devastating consequences. Therefore, there is an urgent need for the development of automatic fake news detection algorithms which can detect fake news as early as possible and help stop the viral spread of such news.

Recently, many approaches are proposed to identify fake news, which can be roughly divided into two categories, i.e., traditional learning [92], [93] and deep learning based models [7], [95], [96], [111]. Traditional learning methods typically extract features from news articles and train classifiers based on the extracted features. Compared with traditional learning methods, deep learning models have achieved an improvement in the performance of fake news detection due to their powerful abilities of learning informative representations automatically. However, training deep learning models usually requires a large amount of hand-labeled data, i.e., news articles that are labeled as real or fake. The creation of such data is expensive and time-consuming. Also, accurate labels can only be obtained when the annotators have sufficient knowledge about the events. Furthermore, the dynamic nature of news articles leads to decaying quality of existing labeled samples. Some of these samples may become outdated quickly and cannot represent the news articles on newly emerged events. To maintain the quality of labeled samples, annotators have to continuously label newly emerging news articles, which is infeasible. To fully unleash the power of

deep learning models in fake news detection, it is essential to tackle the challenge of labeling fake news.

A possible solution is to leverage the feedback provided by users who read the news. Nearly every social medial platform provides a way for users to report their comments about the news, and some of these comments are highly relevant to fake news detection. For example, for a news article published on a WeChat official account[1], a user who reads the article can report whether this news is fake or not with a brief explanation. Such reports from users can be regarded as "weak" annotation for the task of fake news detection. The large collection of user reports can help alleviate the label shortage problem in fake news detection. However, different from expert-labeled samples, these weak annotated samples are unavoidably noisy. Users may report real news as fake ones, and the reasons they provide may not be meaningful. Therefore, how to transform weak annotation to labeled samples in the training set and select high-quality samples is the major issue we need to solve.

In light of the aforementioned challenges, we propose a reinforced weakly-supervised fake news detection framework (WeFEND), which can leverage the crowd users' feedback as weak supervision for fake news detection. The proposed framework WeFEND consists of three main components: the annotator, the fake news detector and the reinforced selector. In particular, given a small set of labeled fake news samples together with users' feedback towards these news articles, we can train an annotator based on the feedback, which can then be used to automatically assign weak labels for those unlabeled news articles simply based on the user feedback they received. The reinforced selector which employs reinforcement learning techniques then selects high-quality samples from weakly labeled samples as the input to the fake news detector. The fake news detector finally assigns a label for each input article based on its content. The three components integrate nicely and their interactions mutually enhance their performance. We conduct extensive experiments on a large collection of news articles published by WeChat official accounts and corresponding feedback reported by users on these articles. Experimental results show that the proposed framework WeFEND outperforms the state-of-the-art approaches on fake news detection. Moreover, we will

[1] ↑WeChat is a Chinese multi-purpose messaging, social media and mobile payment app developed by Tencent. Wechat official accounts push news articles and information for subscribed followers.

publicly release this dataset[2] to the community to encourage further research on fake news detection with user reports.

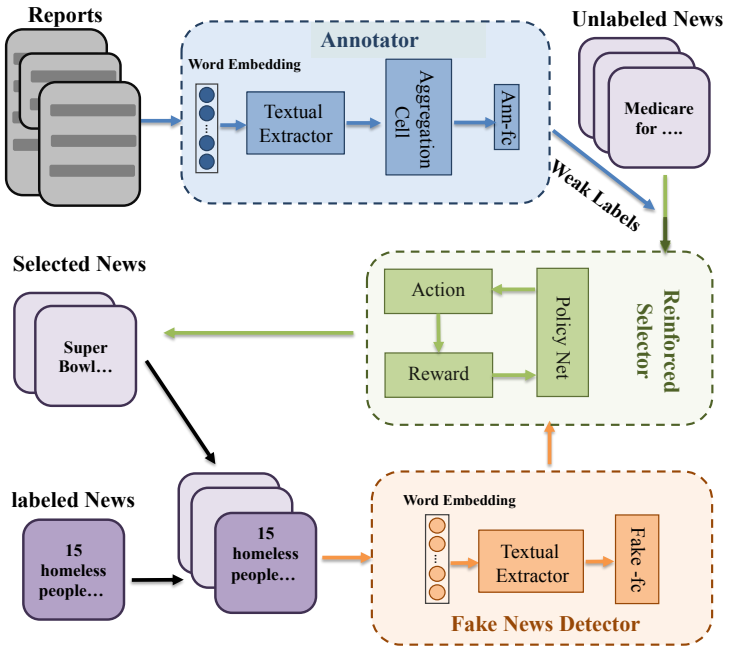


Figure 6.1. The architecture of proposed framework WeFEND.

6.2 Methodology

In this section, we first briefly introduce the overview of the proposed fake news detection framework WeFEND, and then demonstrate each component in detail.

6.2.1 Overview

The problem setting is as follows. Each sample consists of both news articles and user feedback comments. Both are texts, and are transformed into vector representations by word embedding. User feedback comments are referred to as *reports*, which are detailed reasons and evidence provided by users about the credibility of the corresponding news articles. A small set of samples are labeled by experts as fake or real, and our objective is to predict the labels of the unlabeled samples.

[2]↑https://github.com/yaqingwang/WeFEND-AAAI20

Figure 6.1 shows the overview of the proposed framework WeFEND. There are three key components: annotator, data selector and fake news detector. Annotator can be seen as a pretrained model on the reports with their labels. Based on the pretrained model, we can assign weak labels for the unlabeled samples according to the annotator on the reports. However, it is hard to guarantee the quality of weak labels. To automatically choose high-quality samples, we design a data selector by exploiting reinforcement learning techniques on the samples labeled by the annotator. Finally, the selected samples and the original labeled samples are used to train fake news detector. In both annotator and fake news detector, a textual feature extractor is used to extract features from input text. The details of these components are introduced in the following subsections.

6.2.2 Textual Feature Extractor

From Figure 6.1, we can observe that textual feature extractor is a basic module of annotator and fake news detection, as not all words are relevant to the task of fake news detector. In this work, we choose convolutions neural network [101], which is proven effective in the fake news detection [7], as textual feature extractor. The input of the textual feature extractor is news content or a report message, and both can be represented as a sequential list of words. For the t-th word in the sentence, we represent it by the corresponding d dimensional word embedding vector, denoted as $x_t \in \mathbb{R}^d$, which is the input to the convolutions neural network. Details of CNN module [101] are in the Supplemental Material.

The learned representation from textual feature extractor are the input features to annotator and fake news detector. Next, we will introduce how to train an annotator and use it to assign weak labels to the unlabeled samples.

6.2.3 Automatic Annotation based on Reports

One benefit of the proposed framework is that it can automatically assign weak labels to the unlabeled news samples, which helps enlarge the size of the training set with little costs. To train such a model, we propose to use report messages provided by users as weak supervision.

Aggregation Cell. One news article may have reports from multiple users, so we propose to aggregate features obtained from different reports for one sample. Since the report messages

from multiple users for one piece of news are permutation invariant, we design an aggregation cell consisting of a commutative aggregation function and a fully-connected layer. The commutative aggregation function, like sum, mean and max-pooling, can combine the permutation invariant input set. We take the i-th sample as an example, and the j-th report message is represented as $r_{\mathrm{j}}^{(\mathrm{i})}$. The corresponding report message set is denoted as $R^{(\mathrm{i})} = \{r_1^{(\mathrm{i})}, r_2^{(\mathrm{i})}, ..., r_{|R^{(\mathrm{i})}|}^{(\mathrm{i})}\}$, where $|R^{(\mathrm{i})}|$ is the number of report messages of the i-th sample. The report message $r_{\mathrm{j}}^{(\mathrm{i})} \in R^{(\mathrm{i})}$ is first fed into the textual feature extractor to obtain an informative textual feature representation, denoted as $\mathbf{h}_{\mathrm{j}}^{(\mathrm{i})}$. Then we use the aggregation cell to combine the report message set $R^{(\mathrm{i})}$ to learn the hidden feature representation $\mathbf{h}^{(\mathrm{i})}$. In order to stabilize the training procedure, we use average operation as the commutative aggregation function. The procedure of aggregation cell is represented as:

$$\mathbf{h}^{(\mathrm{i})} = \sigma(\mathbf{w}_r \cdot \sum_{\mathrm{j}=1}^{|R^{(\mathrm{i})}|} \frac{\mathbf{h}_{\mathrm{j}}^{(\mathrm{i})}}{|R^{(\mathrm{i})}|}), \tag{6.1}$$

where σ is the ReLU activation function, and $\mathbf{w}_r$ is the weight of the fully-connected layer.

We feed $\mathbf{h}^{(\mathrm{i})}$ into the fully connected layer to output the corresponding probability of the i-th sample being a fake one, which is denoted as $D_r(R^{(\mathrm{i})}, \theta_r)$, where θ_r represents all the parameters of the annotator and corresponding textual feature extractor. The entire report message dataset is represented as $R = \{R^{(1)}, R^{(2)}, ..., R^{(|R|)}\}$, and the corresponding ground truth labels of news are denoted as $Y = \{y^{(1)}, y^{(2)}, ..., y^{(|R|)}\}$, where $|R|$ is the number of report sets. Based on R and Y, the loss function for the proposed annotator is defined by cross entropy as follows:

$$\begin{aligned} L_r(R, Y; \theta_r) = -\frac{1}{|R|} \sum_{\mathrm{i}=1}^{|R|} [y^{(\mathrm{i})} \log D_r(R^{(\mathrm{i})}; \theta_r) \\ + (1 - y^{(\mathrm{i})}) \log(1 - D_r(R^{(\mathrm{i})}; \theta_r))]. \end{aligned} \tag{6.2}$$

Given the unlabeled news set X^u with corresponding report messages, we use the trained annotator to predict their labels, which are denoted as $\hat{Y}^u$. By the annotator, we can obtain a large weakly labeled dataset $\{X^u, \hat{Y}^u\}$. However, the labels in this automatically-annotated dataset are unavoidably noisy and directly adding these samples to the training set may degrade the detection performance. Thus, the challenge here is how to select high-quality samples from this set to guarantee the detection performance. To address this challenge, we propose to employ

reinforcement learning techniques in the design of a data selector. The details of the proposed data selector are introduced in the following subsection.

6.2.4 Data Selection via Reinforcement Learning

The objective of the data selector is to automatically select high-quality samples from those with weak labels obtained from the annotator. The criteria of the selection is based on whether adding the chosen sample can improve the fake news detection performance. According to this criteria, we design a performance-driven data selection method (called reinforced data selector) using reinforcement learning mechanism. Next, we first introduce the input data of the designed data selector, and then present the details of this data selector.

Let $\tilde{X}$ denote all the input data of the proposed reinforced data selector. However, instead of directly putting the entire dataset $\tilde{X}$ into the selector, we divide the whole dataset into K small bags of data samples, i.e., $\tilde{X} = \{\tilde{X}^{(k)}\}_{k=1}^{K}$. A bag of data samples is the input of the selector each time. For the k-th bag of data $\tilde{X}^{(k)}$, it contains B samples, i.e., $\tilde{X}^{(k)} = \{x_1^{(k)}, x_2^{(k)}, ..., x_B^{(k)}\}$. The benefit of using multiple small bags of samples is that this approach can provide more feedback to the selector and this makes the training procedure of reinforcement learning more efficient.

Problem Formulation. In the data selection procedure, the samples in one bag are sequentially fed into the designed reinforced data selector. For every sample, the *action* of reinforced data selector is to retain or remove. The decision of the current sample $x_{\mathrm{i}}^{(k)}$ is based on its *state* vector and all the previous decisions of samples $\{x_1^{(k)}, x_2^{(k)}, ..., x_{\mathrm{i}-1}^{(k)}\}$. Thus, the data selection problem can be naturally cast as a Markov Decision Process (MDP) that is the prerequisite of reinforcement learning. Since the goal of data selection is to improve the performance of fake news detection, we directly use the performance changes of fake news detection as the *reward* for reinforced selector. The performance is evaluated by accuracy. For this sequential decision procedure, the reward is delayed because it can only be obtained after all the decisions are made. To solve the delayed reward problem, we employ the policy-based reinforcement learning mechanism. Since the reinforced selector needs to use the performance changes of fake news detection as reward, we introduce the fake news detector first.

Fake News Detector. Fake news detection model is a neural network, which consists of a textual feature extractor and a fully-connected layer with corresponding activation functions. The input to fake news detector is news content, and the output is the probability of the given news being fake. The detector is denoted as $D_n(\cdot;\theta_n)$, where θ_n represents all the parameters.

After introducing fake news detector, we will introduce the concepts of *state*, *action*, and *reward* used in the proposed reinforced selector in detail as follows.

State. The state vector of the sample $x_{\mathrm{i}}^{(k)}$ is denoted as $s_{\mathrm{i}}^{(k)}$. Since every action is made based on the current sample and the chosen sample, the state vector mainly consists of two components: the representation of the current sample and the average representation of the chosen samples. The representation of the current sample is related to data quality and diversity. We use the output probability from the proposed annotator and the output probability of fake news detector to measure the quality of the data. To represent the data diversity, we first calculate cosine similarity between the current sample and all the chosen samples. Here each sample is represented by a vector obtained from textural feature extractor. We then select the max value of cosine similarity as the diversity. To balance the distribution of classes, the weak label of the current sample is also used as a part of the representation. Therefore, the current state vector contains four elements: 1) the output probability from the annotator, 2) the output probability from fake news detector, 3) the maximum of cosine similarity between the current sample and the chosen samples, and 4) the weak label of the current sample. The representations of all the chosen samples are defined as the average of all the chosen samples' state vectors. The concatenation of the current state vector and the average of previous state vectors is considered as the final state vector $s_{\mathrm{i}}^{(k)}$.

Action. The action value of the reinforced selector for every sample is 1 or 0. 1 represents the action to *retain* the sample, and 0 denotes the action to *remove* it. To determine the action, we train a policy network, denoted as $P(\cdot;\theta_s)$, where θ_s represents the parameters. The policy network includes two fully connected layers with corresponding activation functions. Take the sample $x_{\mathrm{i}}^{(k)}$ as an example. The policy network outputs a probability of retaining, denoted as $p_{\mathrm{i}}^{(k)}$, based on the sample's state vector $s_{\mathrm{i}}^{(k)}$.

$$P(s_{\mathrm{i}}^{(k)};\theta_s) = \psi(\mathbf{w}_{s2} \cdot \sigma(\mathbf{w}_{s1} \cdot s_{\mathrm{i}}^{(k)})), \tag{6.3}$$

where $\mathbf{w}_{s1}$ and $\mathbf{w}_{s2}$ are the weights of two fully-connected layers, σ represents the ReLU activation function, and ψ is the Sigmoid function. Then the action $a_{\mathrm{i}}^{(k)}$ is sampled according to the output probability. The policy $\boldsymbol{\pi}_{\theta_s}(s_{\mathrm{i}}^{(k)}, a_{\mathrm{i}}^{(k)})$ can be represented as

$$\boldsymbol{\pi}_{\theta_s}(s_{\mathrm{i}}^{(k)}, a_{\mathrm{i}}^{(k)}) = \begin{cases} p_{\mathrm{i}}^{(k)} & \text{if } a_{\mathrm{i}}^{(k)} = 1 \\ 1 - p_{\mathrm{i}}^{(k)} & \text{if } a_{\mathrm{i}}^{(k)} = 0 \end{cases}.$$

Reward. Since the goal of the action is to retain the samples that can bring improvement to fake news detection, we use the performance changes of detection model $D_n(\cdot;\theta_n)$ as the reward function. Given the k-th bag of data $\{x_1^{(k)}, x_2^{(k)}, ..., x_B^{(k)}\}$, the actions of retaining or removing are made based on the probability output from the policy network. To evaluate the performance changes, we need to set a baseline accuracy acc. To this end, we first extract a validation dataset from the whole labeled dataset. Note that all the trained model will test on this extracted validation dataset. We then calculate the baseline accuracy acc with the detection model $D_n(\cdot;\theta_n)$. Since the designed data selector can choose some high-quality samples from $\{x_1^{(k)}, x_2^{(k)}, ..., x_B^{(k)}\}$, the fake news detection model will be retrained using the retained data samples. A new accuracy acc_k can be obtained with the retrained model on the validation dataset. Finally, the reward R_k for k-th bag data $\{x_{\mathrm{i}}^{(k)}\}_{\mathrm{i}=1}^{B}$ is represented by the difference of acc_k and acc as follows:

$$R_k = acc_k - acc. \tag{6.4}$$

For the k-th bag of data $\{x_{\mathrm{i}}^{(k)}\}_{\mathrm{i}=1}^{B}$, we aim to maximize the expected total reward. Since the scale of R_k is small, we use the summation of reward to define the objective function in order to make the training procedure more efficient. The objective function is defined as

$$J(\theta_s) = \sum_{\mathrm{i}=1}^{B} \boldsymbol{\pi}_{\theta_s}(s_{\mathrm{i}}^{(k)}, a_{\mathrm{i}}^{(k)}) R_k. \tag{6.5}$$

The derivative of the objective function above is

$$\begin{aligned} \bigtriangledown_{\theta} J(\theta_s) &= \sum_{\mathrm{i}=1}^{B} R_k \bigtriangledown_{\theta_s} \boldsymbol{\pi}_{\theta_s}(s_{\mathrm{i}}^{(k)}, a_{\mathrm{i}}^{(k)}) \\ &= \mathbb{E}_{\pi_{\theta_s}}[\sum_{\mathrm{i}=1}^{B} R_k \bigtriangledown_{\theta_s} \log \boldsymbol{\pi}_{\theta_s}(s_{\mathrm{i}}^{(k)}, a_{\mathrm{i}}^{(k)})] \end{aligned} \tag{6.6}$$

According to the policy-based reinforcement learning algorithm [112], [113], we update the parameters θ of the policy network by stochastic gradient ascent as follows:

$$\theta_s \leftarrow \theta_s + \alpha \sum_{\mathrm{i}=1}^{B} R_k \bigtriangledown_{\theta_s} \log \boldsymbol{\pi}_{\theta_s}(s_{\mathrm{i}}^{(k)}, a_{\mathrm{i}}^{(k)}), \tag{6.7}$$

where α is the learning rate. To improve the exploration and stabilize training, we train a target policy network $P(\cdot, \theta_s{}')$ that updates much slower than the policy network $P(\cdot, \theta_s)$:

$$\theta_s' = \tau\theta_s' + (1-\tau)\theta_s. \tag{6.8}$$

In the training stage, half of the bags are fed into the policy network $P(\cdot; \theta_s)$ and the another half of bags are fed into the target policy network $P(\cdot; \theta_s')$.

6.2.5 Reinforced Weakly-supervised Fake News Detection Framework

In this subsection, we introduce how to integrate the three key components: annotator, fake news detector and reinforced selector. First, we pretrain the annotator using the labeled report data $\{R, Y\}$ and assign weak labels $\hat{Y}^u$ to the unlabeled news set X^u. The proposed reinforced selector will select high-quality samples from the weakly labeled dataset $\{X^u, \hat{Y}^u\}$. Here we set the selected bags as K. Then both the selected data set $\{X_s, Y_s\} = \{X_s^{(k)}, Y_s^{(k)}\}_{k=1}^{K}$ and the original labeled data are fed into the fake news detector for training. Thus, the final loss of fake news detection consists of two sub losses:

$$\begin{aligned} L_n(X, Y, X_s, Y_s; \theta_n) =& \lambda_l \cdot L_n^l(X, Y; \theta_n) \\ &+ \lambda_s \cdot L_n^s(X_s, Y_s; \theta_n), \end{aligned} \tag{6.9}$$

where $L_n^l(X, Y; \theta_n)$ and $L_n^s(X_s, Y_s; \theta_n)$ are the losses on a small amount of manually labeled data and automatically-annotated data set respectively. Here the λ_l and λ_u control the balance between $L_n^l(\theta_n)$ and $L_n^s(\theta_n)$, and we simply set the values of λ_l and λ_u as 1. The two losses are defined by cross entropy respectively as follows:

$$\begin{aligned} L_n^l(X, Y; \theta_n) = &- \mathbb{E}_{(x,y)\sim(X,Y)} \left[y \log(D_n(x; \theta_n)) \right. \\ &\left. + (1 - y) \log(1 - D_n(x; \theta_n)) \right], \end{aligned} \tag{6.10}$$

$$\begin{aligned} L_n^s(X_s, Y_s; \theta_n) = &- \mathbb{E}_{(x_s,y_s)\sim(X_s,Y_s)} \left[y_s \log(D_n(x_s, \theta_n)) \right. \\ &\left. + (1 - y_s) \log(1 - D_n(x_s; \theta_n)) \right]. \end{aligned} \tag{6.11}$$

6.3 Experiments

In this section, we introduce the dataset used in the experiments, present the compared fake news detection models, validate the effectiveness and explore some insights of the proposed framework.

6.3.1 Dataset

To fairly evaluate the performance of the proposed framework, we collect a dataset from WeChat's Official Accounts and conduct comprehensive experiments to analyze the performance. This dataset includes user reports and will be publicly released in future to encourage research on fake news detection.

Table 6.1. The Statistics of the WeChat Datasets.

		# News	# Report	# Avg. Reports/News
Unlabeled	-	22981	31170	1.36
Labeled Training	Fake	1220	2010	1.65
	Real	1220	1740	1.43
Labeled Testing	Fake	870	1640	1.89
	Real	870	1411	1.62

In this dataset, the news are collected from WeChat's Official Accounts, dated from March 2018 to October, 2018. To facilitate the detection fake news, the WeChat's Official Account encourages users to report suspicious articles, and write feedback to explain why they think these articles are

suspicious. To obtain a small set of labeled samples, we first collect the news with reports and then send them to the experts of WeChat team for verification. Thus, the manually labeled fake and real news both have report messages. We split the fake news and real news into training and testing sets according to the post timestamp. The news in the training data were posted from March 2018 to September 2018, and testing dataset is from September 2018 to October 2018. There is no overlapped timestamp of news between these two sets. This design is to evaluate the performance of fake news detection on the fresh news. We also have an unlabeled set containing a large amount of collected news without annotation. The time window of the unlabeled set is from September to October 2018. The detailed statistics are shown in the Table 8.1. Note that the headlines can be seen as the summary of the news content. In the manual annotation process, experts only look at headlines to conduct labeling. Thus, in this work, we use headlines as the input data.

6.3.2 Baselines

To validate the effectiveness of the proposed model, we choose both traditional machine learning algorithms and deep learning models as baseline methods. Previous work on verbal deception detection showed that LIWC [114] is a valuable tool for the deception detection in various contexts [115], [116]. Based on LIWC features, we detect fake news with different traditional machine learning algorithms including Logistic Regression (LIWC-LR), SVM (LIWC-SVM) and Random Forest (LIWC-RF). Besides traditional machine learning algorithms, we also compare the proposed algorithm with the stat-of-the-art deep learning fake news detection models LSTM [95], CNN [7] and EANN [7]. To show effects of automatic annotation, we proposed two semi-supervised models based on CNN and LSTM, which are denoted as LSTM_{semi} and CNN_{semi} respectively, as baselines. Furthermore, the complete WeFEND model consists of three components: annotator, fake news detector and data selector. To show the role of data selector, we design one variant of the proposed model named WeFEND$-$, which does not include data selector.

• **Supervised Setting.** We split the manually labeled training set into two sets in a ratio 8:2. 20% of training set is used as a validation set to select parameters, and the remaining 80% of data is used for training purpose.

• **Semi-supervised Setting.** We still split the data as the supervised setting, but also use the unlabeled data in the semi-supervised setting. The combination of 80% of training set and unlabeled set are used for training. We use the entropy minimization to define the loss on unlabeled data as [117]. Considering the relative data size, we set the ratio between two losses on labeled set and unlabeled set as 1:0.1.

• **Weakly-supervised Setting.** We pretrain the proposed annotator on *reports* in the training set and use the pretrained annotator to automatically annotate the unlabeled set. Based on the annotation, we divide the unlabeled data into two sets: weakly fake data and weakly real data. For each set, we randomly select a subset of data samples. The number of data samples is 10% of the whole unlabeled data. These two subsets consist of a new validation set, which is used for choosing the best parameters. All the remaining data with weak labels are used for training the model.

• **Hybrid Setting.** All the settings for Hybird are the same as those of Weakly-supervised, but in the last step, we use both the labeled training data (80% as the Supervised and Semi-supervised settings) and the remaining data with weak labels to train the model.

6.3.3 Performance Comparison

Table 6.2. The performance comparison of different methods on WeChat dataset.

Category	Method	Accuracy	AUC-ROC	Fake News			Real News		
				Precision	Recall	F_1	Precision	Recall	F_1
Supervised	LIWC-LR	0.528	0.558	0.604	0.160	0.253	0.517	0.896	0.655
	LIWC-SVM	0.568	0.598	0.574	0.521	0.546	0.563	0.614	0.587
	LIWC-RF	0.590	0.616	0.613	0.483	0.541	0.574	0.696	0.629
	LSTM	0.733	0.799	0.876	0.543	0.670	0.669	**0.923**	0.775
	CNN	0.747	0.834	0.869	0.580	0.696	0.685	0.913	0.783
	EANN	0.767	0.803	0.863	0.634	0.731	0.711	0.899	0.794
Semi-supervised	LSTM_{semi}	0.753	0.841	0.854	0.611	0.713	0.697	0.895	0.784
	CNN_{semi}	0.759	0.848	0.850	0.630	0.723	0.706	0.889	0.787
Automatically annotated	WeFEND−	0.807	0.858	0.846	**0.751**	0.795	0.776	0.863	0.817
	WeFEND	**0.824**	**0.873**	**0.880**	**0.751**	**0.810**	**0.783**	0.898	**0.836**

Table 8.2 shows the performance of different approaches on the WeChat dataset. We can observe that that the proposed framework achieves the best results in terms of Accuracy, AUC-ROC, precision, recall and F_1 measurement.

In the supervised setting, LIWC-LR achieves the worst performance. The reason is that LIWC-LR is a linear model and hard to discriminate the complicated distributions of fake and real news content. Compared with LIWC-LR, LIWC-SVM and LIWC-RN improve the performance in terms of most measurements. However, compared with traditional machine learning models, deep learning based models, including LSTM, CNN and EANN, significantly improve the performance. This confirms that deep learning models have superior ability to extract informative features for detection. In particular, compared with the best traditional machine learning baseline LIWC-RF, CNN achieves around 27% and 35% improvement on Accuracy and AUC-ROC respectively. EANN model has the ability to capture the dynamic nature of news by learning the event-invariant feature representations. It leads to the performance improvement and better generalization ability compared with the plain LSTM and CNN.

Along with the setting of supervised learning, in semi-supervised setting, we incorporate external unlabeled news. We run LSTM and CNN models in the semi-supervised setting. Since the number of data largely increases, we can observe the performance improvement in both models. Take LSTM-based models as an example. The Accuracy and AUC-ROC of LSTM$_{semi}$ increases 3% and 5% respectively, compared with supervised LSTM. This illustrates that using unlabeled data enlarges size of training set and achieves performance improvement.

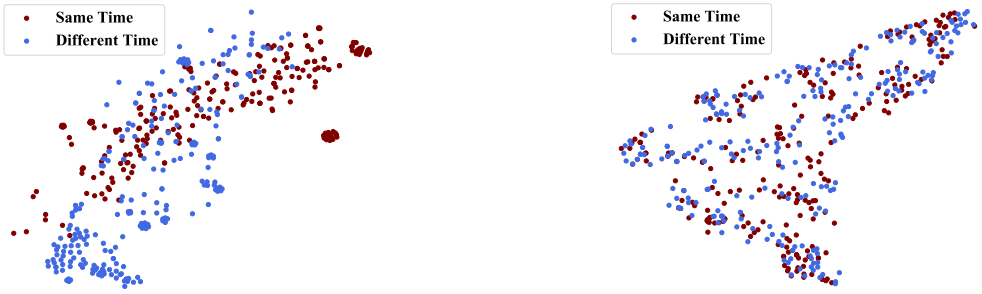
The advantage of the proposed framework is that it can automatically annotate unlabeled news. From the results shown in Table 8.2, we can observe that the performance of WeFEND$-$ is better than this of models in the supervised setting and semi-supervised setting.

Though incorporating automatic annotation as weak supervision helps fake news detection in some aspects, weak supervision is unavoidably noisy. In Table 8.2, the recall values of WeFEND$-$ improve as the coverage is increasing, but their precision values for fake news detection decrease. This shows that incorporating weak supervision may add more false positive examples. For real news, since the majority of unlabeled data with reports is still real news, the precision still improves. To reduce the influence of noisy labels, the proposed framework WeFEND has the data selector component based on reinforcement learning techniques. After incorporating data selector, the precision values of fake news and real news are improved compared with their reduced version in the same hybrid setting. Furthermore, we can observe from Table 8.2 that the proposed WeFEND achieve the best performance compared with all the baselines.

6.3.4 Insight Analysis

Due to the dynamic nature of news, the annotation needs to be timely to cover news articles on newly emerged events. To address this issue, we propose to use reports from users to automatically annotate fresh news. To valid our intuition, we conduct experiments to demonstrate why reports are useful for this purpose.

The experiment is designed as follows. We first split the original training dataset consists of news content and reports into two sets: 80% data as the new training set (denoted as D_t) and the remaining 20% data as the testing set for the same time window setting (denoted as D_s). For the different time window setting, we randomly select a subset samples from original testing dataset, which is denoted as D_d. The number of samples in D_s is similar as that in D_d. The fake news detector and annotator are first trained on the news content of D_t, and then we separately test the models on D_s and D_d. We show the distributions of reports on two time sets. Since the real news is easy to collect, the goal of annotation procedure is to expand the size of fake news samples. Thus, to analyze the distribution of reports, we mainly focus on fake news samples. The distributions of reports on the same and different time set are shown in the Figure 6.2. For clear comparison between the distribution of reports and news content, the feature representations of news content for fake news are also shown in Figure 6.2a.



(a) News Content (b) Reports

Figure 6.2. The Visualization of latent representations for news content and reports of fake news.

From Figure 6.2a, we can observe that although the distributions of news content in the same time set and different time set have overlaps, the samples from two set are separately clustered at the top right and bottom left corner. This shows the distribution of news contents changes with time. In contrast, the feature representations of report messages from two sets are all twisted and cannot be

distinguished as shown in Figure 6.2b. This proves that the distributions of reports is time invariant and further explains why the model trained on report messages achieves a consistent performance. Thus, the annotation based on reports can guarantee consistent quality even for fresh news articles.

6.3.5 Importance of Reinforced Selector

To demonstrate the importance of reinforced selector, we run WeFEND− ("w/o RL") and WeFEND ("w RL") 5 times, and the performance comparison during the first 30 epochs is shown in Figure 6.3. Note that the only difference between two models is whether it has the component of reinforced selector or not. The solid line represents the average accuracy of 5 times, and the line with light color represents the accuracy value of a single time. As the probability output from fake news detection model can provide more information for the reinforced selector, we can observe that the average accuracy of the model with reinforced selector is stably higher than that w/o reinforced selector after 12 epochs from Figure 6.3. The ablation study shows that the designed reinforced selector is effective in improving the performance of fake news detection.

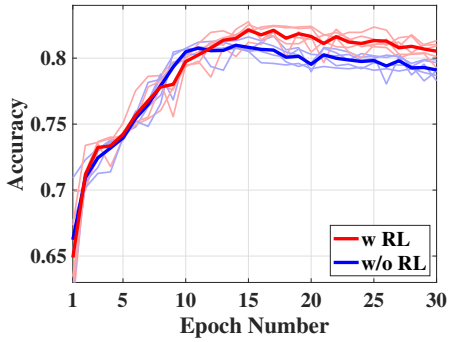


Figure 6.3. The changes of Accuracy in terms of the number of Epochs.

7. FEW-SHOT DOMAIN ADAPTATION FOR FAKE NEWS DETECTION

(A version of this chapter has been previously published in KDD 2021 [118].)

7.1 Introduction

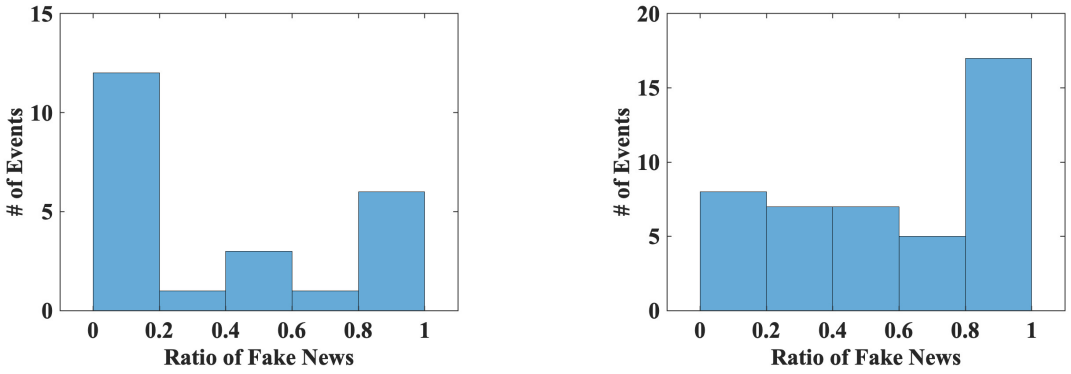
The recent proliferation of social media has significantly changed the way in which people acquire information. According to the 2018 Pew Research Center survey, about two-thirds of American adults (68%) get news on social media at least occasionally. The fake news on social media usually take advantage of multimedia content which contain misrepresented or even forged images, to mislead the readers and get rapid dissemination. The dissemination of fake news may cause large-scale negative effects, and sometimes can affect or even manipulate important public events. Recent years have witnessed a number of high-impact fake news spread regarding terrorist plots and attacks, presidential election and various natural disasters. Therefore, there is an urgent need for the development of automatic detection algorithms, which can detect fake news as early as possible to stop the spread of fake news and mitigate its serious negative effects.

Figure 7.1. Fake news examples on an emergent event Boston Bombing from Twitter.

Task Challenges. Thus far, various fake news detection methods, including both traditional learning [92], [93] and deep learning based models [7], [95], [96], [111], [119], [120] have been exploited to identify fake news. Despite the success of deep learning models with large amounts of labeled datasets, the algorithms still suffer in the cases where fake news detection is needed on emergent events. Due to the domain shift in the news events [8], the model trained on past

events may not achieve satisfactory performance and thus the new knowledge from emergent events are needed to add into fake news detection models. However, adding the knowledge from newly emergent events requires to build a new model from scratch or continue to fine-tune the model on newly collected labeled data, which can be challenging, expensive, and unrealistic for real-world settings. Moreover, fake news usually emerged on newly arrived events where we hardly obtain sufficient posts in a timely manner. In the early stage of emergent events, we usually only have a handful of related verified posts (An example is shown in the Fig. 7.1). How to leverage *a small set of verified posts* to make the model learn quickly to detect fake news on the newly-arrived events is a crucial challenge.

Limitations of Current Techniques. To overcome the challenge above, the few-shot learning, which aims to leverage a small set of data instances for quick learning, is a possible solution. One promising research line of few-shot learning is **meta-learning** [121], [122], whose basic idea is to leverage the global knowledge from previous tasks to facilitate the learning on new task. However, the success of existing meta-learning methods is highly associated with an important assumption: the tasks are from a similar distribution and the shared global knowledge applies to different tasks. This assumption usually does not hold in the fake news detection problem as the writing style, content, vocabularies and even class distributions of news on different events usually tends to differ. As it can be observed from Figure 7.2, the ratios of fake news on events are significantly different.



(a) Twitter (b) Weibo

Figure 7.2. The number of events with respect to different percentages of fake news.

The significant difference across events posts serious challenges on **event heterogeneity**, which cannot be simply handled by globally sharing knowledge [123]. Another research line of few-shot

learning is **neural processes np**, [124], [125], which conduct inference using a small set of data instances as conditioning. Even though neural processes show better generalizablity , they are based on a fixed set of parameters and usually suffer from the limitations like **underfitting** [125], thereby leading to unsatisfactory performance. These two research lines of models are complementary to each other: the parameter adaptation mechanism in meta-learning can provide more parameter flexibility to *alleviate unfitting issues* of the neural process. Correspondingly, the neural processes can help handle *the heterogeneity challenge* for MAML by using a small set of data instances as conditioning instead of encoding all the information into parameter set. Although it is promising to integrate two popular few-shot approaches together, the incompatible operations on the given small set of data instances is the main obstacle for developing the model based on these two.

Our Approach. To address the aforementioned challenges, in this work, we propose a novel meta neural process network (namely MetaFEND) for emergent fake news detection. MetaFEND unifies the incompatible operations from meta-learning and neural process via a simple yet novel simulated learning task, whose goal is to adapt the parameters to better take advantage of given support data points as conditioning. Toward this end, we propose to conduct leave-one-out prediction as shown in the Fig. 7.3, i.e., we repeatedly use one of given data as target data and the rest are used as context set for conditioning on all the data in support set. Therefore, the proposed model can handle heterogeneous events via event adaption parameters and conditioning on event-specific data instances simultaneously. Furthermore, we incorporate two novel components - *label embedding* and *hard attention* - to handle categorical characteristics of label information and extract the most informative instance as conditioning despite imbalanced class distributions of news events. Experimental results on two large real-world datasets show that the proposed model effectively detect fake news on new events with a handful of posts and outperforms the state-of-the-art approaches.

7.2 Background

We define our problem and introduce preliminary works in this section.

7.2.1 Problem Formulation

There are many tasks related to fake news detection, such as rumor detection [107] and spam detection [126]. Following the previous work [95], [97], we specify the definition of fake news as news which is intentionally fabricated and can be verified as false. In this work, we tackle fake news detection on emergent events and make a practical assumption that a few labeled examples are available per event. Our goal is to leverage the knowledge learned from past events to conduct effective fake news detection on newly arrived events with a few examples. More formally, we define the fake news detection following the few-shot problem.

Few-shot Fake News Detection Let $\mathcal{E}$ denote a set of news events. In each news event $\mathrm{e} \sim \mathcal{E}$, we have a few labeled posts on the event e. The core idea of few-shot learning is to use episodic classification paradigm to simulate few-shot settings during model training. In each episode during the training stage, the labeled posts are partitioned into two independent sets, support set and query set. Let $\{\mathbf{X}_{\mathrm{e}}^{s}, \mathbf{Y}_{\mathrm{e}}^{s}\} = \{x_{\mathrm{e,i}}^{s}, y_{\mathrm{e,i}}^{s}\}_{\mathrm{i}=1}^{K}$ represent the support set, and $\{\mathbf{X}_{\mathrm{e}}^{q}, \mathbf{Y}_{\mathrm{e}}^{q}\} = \{x_{\mathrm{e,i}}^{q}, y_{\mathrm{e,i}}^{q}\}_{\mathrm{i}=K+1}^{N}$ be the query set. The model is trained to learn to conduct fake news detection on the query set $\{\mathbf{X}_{\mathrm{e}}^{q}, \mathbf{Y}_{\mathrm{e}}^{q}\}$ given the support set $\{\mathbf{X}_{\mathrm{e}}^{s}, \mathbf{Y}_{\mathrm{e}}^{s}\}$. During the inference stage, K labeled posts are provided per event. For each event e, the model leverages its corresponding K labeled posts as support set $\{\mathbf{X}_{\mathrm{e}}^{s}, \mathbf{Y}_{\mathrm{e}}^{s}\} = \{x_{\mathrm{e,i}}^{s}, y_{\mathrm{e,i}}^{s}\}_{\mathrm{i}=1}^{K}$ to conduct fake news detection on given event e.

7.2.2 Preliminary Work

MAML. We first give an overview of MAML method [121], a representative algorithm of gradient-based meta-learning approaches, and take few-shot fake news detection as an example. The meta-learning procedure is split into two stages: meta-training and meta-testing.

During the *meta-training* stage, the baseline learner f_θ is adapted to specific event e as $f_{\theta_{\mathrm{e}}}$ with the help of the support set $\{\mathbf{X}_{\mathrm{e}}^{s}, \mathbf{Y}_{\mathrm{e}}^{s}\}$. Such an event specific learner $f_{\theta_{\mathrm{e}}}$ is evaluated on the corresponding query set $\{\mathbf{X}_{\mathrm{e}}^{q}, \mathbf{Y}_{\mathrm{e}}^{q}\}$. The loss $\mathcal{L}(f_{\theta_{\mathrm{e}}}, \{\mathbf{X}_{\mathrm{e}}^{q}, \mathbf{Y}_{\mathrm{e}}^{q}\})$ on $\{\mathbf{X}_{\mathrm{e}}^{q}, \mathbf{Y}_{\mathrm{e}}^{q}\}$ is used to update the parameters of baseline learner θ. During the meta-testing stage, the baseline learner f_θ is adapted to the testing event e' using the procedure in meta-training stage to obtain event specific parameters $\theta_{\mathrm{e}'}$, which is employed to make predictions on the query set $\{\mathbf{X}_{\mathrm{e}'}^{q}, \mathbf{Y}_{\mathrm{e}'}^{q}\}$ of event e'.

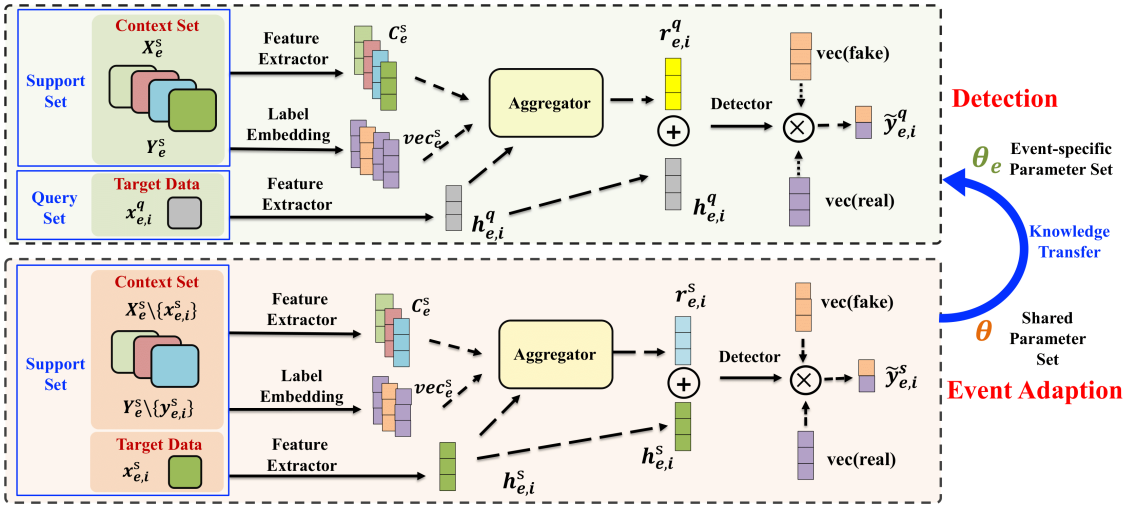


Figure 7.3. The proposed framework MetaFEND. The proposed framework has two stages: event adaption and detection. During the event adaption stage, the model parameter set θ is updated to event-specific parameter set θ_{e}. During the detection stage, the event-specific parameter set θ_{e} is used to detect fake news on event e. $\oplus$ denotes concatenation operation and $\otimes$ means element-wise product.

MAML update parameter vector θ using one or more gradient descent updates on event e. For example, when using one gradient update:

$$\theta_{\mathrm{e}} = M(f_\theta, \{\mathbf{X}_{\mathrm{e}}^s, \mathbf{Y}_{\mathrm{e}}^s\}) = \theta - \alpha \bigtriangledown_\theta \mathcal{L}(f_\theta, \{\mathbf{X}_{\mathrm{e}}^s, \mathbf{Y}_{\mathrm{e}}^s\}).$$

The model parameters are trained by optimizing for the performance of $f_{\theta_{\mathrm{e}}}$ with respect to θ across events sampled from $p(\mathcal{E})$. More concretely, the meta-objective is as follows:

$$\min_\theta \sum_{\mathrm{e}\sim\mathcal{E}} \mathcal{L}(f_{\theta_{\mathrm{i}}}) = \sum_{\mathrm{e}\sim\mathcal{E}} \mathcal{L}(f_{\theta - \alpha\bigtriangledown_\theta \mathcal{L}(f_\theta, \{\mathbf{X}_{\mathrm{e}}^s, \mathbf{Y}_{\mathrm{e}}^s\})}, \{\mathbf{X}_{\mathrm{e}}^q, \mathbf{Y}_{\mathrm{e}}^q\}).$$

Limitations of MAML. The MAML can capture task uncertainty via one or several gradient updates. However, in fake news detection problem, when events are heterogeneous, the event uncertainty is difficult to encode into parameters via one or several gradient steps. Moreover, even if given support data and query data of interest are from the same event, there is no guarantee that they are all highly related to each other. In such a case, the parameter adaption on fake news detection loss on support set may be misleading for some posts.

Conditional Neural Process (CNP). The CNP includes four components: encoder, feature extractor, aggregator and decoder. The basic idea of conditional neural process is to make predictions with the help of support set $\{\mathbf{X}_{\mathrm{e}}^{s}, \mathbf{Y}_{\mathrm{e}}^{s}\} = \{x_{\mathrm{e,i}}^{s}, y_{\mathrm{e,i}}^{s}\}_{\mathrm{i}=1}^{K}$ as context. The dependence of a CNP on the support set is parametrized by a neural network encoder, denoted as $g(\cdot)$. The encoder $g(\cdot)$ embeds each observation in the support set into feature vector, and the aggregator $\mathrm{agg}(\cdot)$ maps these feature vectors into an embedding of fixed dimension. In CNP, the aggregation procedure is a permutation-invariant operator like averaging or summation. The query data of interest $x_{\mathrm{e,i}}^{q}$ is fed into feature extractor $h(\cdot)$ to get the feature vector. Then the decoder $f(\cdot)$ takes the concatenation of aggregated embedding and given target data $x_{\mathrm{e,i}}^{q}$ as input and output the corresponding prediction as follows:

$$p(y_{\mathrm{e,i}}^{q}|\{\mathbf{X}_{\mathrm{e}}^{s}, \mathbf{Y}_{\mathrm{e}}^{s}\}, x_{\mathrm{e,i}}^{q}) = f\Big(\mathrm{agg}(g(\{\mathbf{X}_{\mathrm{e}}^{s}, \mathbf{Y}_{\mathrm{e}}^{s}\})) \oplus h(x_{\mathrm{e,i}}^{q})\Big).$$

where $\oplus$ is concatenation operator.

Limitations of CNP. One widely recognized limitation of CNP is underfitting [125]. For different context data points, their importance is usually different in the prediction. However, the aggregator of CNP treats all the support data equally and cannot achieve query-dependent context information. Moreover, the CNP simply concatenates the input features and numerical label values of posts together as input, ignoring the categorical characteristics of labels.

7.3 Methodology

In this work, we study how to develop an effective model which can identify fake news on emergent events with a small set of labeled data. To this end, we propose a meta neural process framework which can fuse meta-learning and neural process methods together via a simulated task. To tackle the challenges brought by heterogeneous news events, we further propose a label embedding component to handle categorical labels and a hard attention component, which can select the most informative information from the support set with imbalanced class distributions. In the next subsection, we introduce our overall design and architecture.

7.3.1 Meta-learning Neural Process Design

As shown in Figure 7.3, our proposed framework includes two stages: event adaptation and detection. The event adaptation stage is to adapt the model parameters to specific event with the help of the support set. The detection stage is to detect fake news on the given event with the help of the support and the adapted parameter set.

Event adaption. We take the i-th support data $\{x^s_{\mathrm{e,i}}, y^s_{\mathrm{e,i}}\}$ as an example, in the event adaption stage, the $\{x^s_{\mathrm{e,i}}, y^s_{\mathrm{e,i}}\}$ is used as target data and the rest of support set $\{\mathbf{X}^s_{\mathrm{e}}, \mathbf{Y}^s_{\mathrm{e}}\} \setminus \{x^s_{\mathrm{e,i}}, y^s_{\mathrm{e,i}}\}$ are used as context set accordingly. The context set $\{\mathbf{X}^s_{\mathrm{e}}, \mathbf{Y}^s_{\mathrm{e}}\} \setminus \{x^s_{\mathrm{e,i}}, y^s_{\mathrm{e,i}}\}$ and target data $x^s_{\mathrm{e,i}}$ are fed into the proposed model to output the prediction. The loss can be calculated between the prediction $\hat{y}^s_{\mathrm{e,i}}$ and the corresponding label $y^s_{\mathrm{e,i}}$. For simplicity, we use θ to represent all the parameters included in the proposed model. Then, our event adaption objective function on the support set can be represented as follows:

$$\mathcal{L}^s_{\mathrm{e}} = \sum_{\mathrm{i}} \log p_\theta(y^s_{\mathrm{e,i}} | \{\mathbf{X}^s_{\mathrm{e}}, \mathbf{Y}^s_{\mathrm{e}}\} \setminus \{x^s_{\mathrm{e,i}}, y^s_{\mathrm{e,i}}\}, x^s_{\mathrm{e,i}}). \tag{7.1}$$

We then update parameters θ one or more gradient descent updates on $\mathcal{L}^s_{\mathrm{e}}$ for event e. For example, when using one gradient update:

$$\theta_{\mathrm{e}} = \theta - \alpha \bigtriangledown_\theta \mathcal{L}^s_{\mathrm{e}}. \tag{7.2}$$

Detection stage. The proposed model with event-specific parameter set θ_{e} takes query set $\mathbf{X}^q_{\mathrm{e}}$ and entire support set $\{\mathbf{X}^s_{\mathrm{e}}, \mathbf{Y}^s_{\mathrm{e}}\}$ as input and outputs predictions $\hat{\mathbf{Y}}^q_{\mathrm{e}}$ for query set $\mathbf{X}^q_{\mathrm{e}}$. The corresponding loss function in the detection stage can be represented as follows:

$$\mathcal{L}^q_{\mathrm{e}} = \log p_{\theta_{\mathrm{e}}}(Y^q_{\mathrm{e}} | X^s_{\mathrm{e}}, Y^s_{\mathrm{e}}, X^q_{\mathrm{e}}). \tag{7.3}$$

Through this meta neural process, we can learn an initialization parameter set which can rapidly learn to use given context input-outputs as conditioning to detect fake news on newly arrived events.

Neural Network Architecture. From Figure 7.3, we can observe that the network structures used in these two stages are the same, including feature extractor, label embedding, aggregator and detector. The feature extractor is a basic module which can take posts as input and output

corresponding feature vectors. Label embedding component is to capture semantic meanings of labels. Then we use an aggregator to aggregate these information into a fixed dimensional vector, namely context embedding, which is used as reference for fake news detection. Thereafter both the context embedding and target feature vector are fed into detector to output a vector. The final prediction is based on the similarities between this output vector and label embeddings. In the following subsections, we use event adaption to introduce the details of each component in our proposed model. For simplicity, we omitted superscript s and q in the illustrations about components.

7.3.2 Feature Extractor

From Figure 7.3, we can observe that feature extractor is a basic module to process raw input. Following the prior works [7], [8], our feature extractor consists of two parts: textual feature extractor and visual feature extractor. For a minor note, the feature extractor is a plug-in component which can be easily replaced by other state-of-the-art models.

Textual feature extractor. We adopt convolutional neural network [127], which is proven effective in the fake news detection [7], [8], as textual feature extractor. The input of the textual feature extractor is unstructured news content, which can be represented as a sequential list of words. For the t-th word in the sentence, we represent it by the word embedding vector which is the input to the convolutional neural network. After the convolutions neural network, we feed the output into a fully connected layer to adjust the dimension to d_f dimensional textual feature vector.

Visual feature extractor. The attached images of the posts are inputs to the visual feature extractor. In order to efficiently extract visual features, we employ the pretrained VGG19 [128] which is used in the multi-modal fake news works [7], [105]. On top of the last layer of VGG19 network, we add a fully connected layer to adjust the dimension of final visual feature representation to the same dimension of textual feature vector d_f. During the joint training process with the textual feature extractor, we freeze the parameters of pre-trained VGG19 neural network to avoid overfitting.

For a multimedia post, we feed the text and image of the example into textual and visual feature extractor respectively. The output of two feature extractors are concatenated together to form a feature vector. For the target data $x_{\mathrm{e,i}}$, we denote its feature vector as $\mathbf{h}_{\mathrm{e,i}}$. For the context data $x_{\mathrm{e},k}$ where $k \neq \mathrm{i}$, we denote its feature vector as $\mathbf{c}_{\mathrm{e},k} \in \mathbf{C}_{\mathrm{e}}$.

7.3.3 Aggregator

To construct context embedding for target data, we need to design an aggregator which satisfies two properties: permutation-invariant and target-dependent. To satisfy the two properties, we choose to adopt the attention mechanism which can compute weights of each observations in context set with respect to the target and aggregates the values according to their weights to form the new value accordingly.

Attention mechanism. In this work, we use scaled dot-product attention mechanism [25]. This attention function can be described as mapping a query and a set of key-value pairs to an output, where the query $\mathbf{Q}$, keys $\mathbf{K}$, values $\mathbf{V}$, and output are all vectors. In our problem, for the target data $x_{\mathrm{e,i}}$ and the context set $\mathbf{X}_{\mathrm{e}} \setminus \{x_{\mathrm{e,i}}\} = \{x_{\mathrm{e},k}\}_{k=1,k\neq \mathrm{i}}^{K}$ on event e. We use the the target feature vector $\mathbf{h}_{\mathrm{e,i}} \in \mathbb{R}^{1\times d}$ after linear transformation as query vector $\mathbf{Q}_{\mathrm{i}}$, the context feature vector $\mathbf{C}_{\mathrm{e}} = [c_{\mathrm{e},1}, ..., c_{\mathrm{e},K}] \in \mathbb{R}^{K\times d}$ after linear transformation as the Key vector $\mathbf{K}$. For the context set, we represent its label information $\mathbf{Y}_{\mathrm{e}} \setminus \{y_{\mathrm{e,i}}\} = \{y_{\mathrm{e},k}\}_{k=1,k\neq \mathrm{i}}^{K}$ by semantic embeddings as $\mathbf{vec}_{\mathrm{e}} = \{\mathbf{vec}_{\mathrm{e},k}\}_{k=1,k\neq \mathrm{i}}^{K}$. The details of label embedding are introduced in the next subsection. Then we concatenate context feature vector and label embedding as $\mathbf{C}_{\mathbf{e}} \oplus \mathbf{vec}_{\mathrm{e}} = [\mathbf{c}_{\mathrm{e},1} \oplus \mathbf{vec}_{\mathrm{e},1}, ..., \mathbf{c}_{\mathrm{e},K} \oplus \mathbf{vec}_{\mathrm{e},K}] \in \mathbb{R}^{(K-1)\times 2d}$. The concatenated embedding after linear transformation is used as value vector $\mathbf{V}$. We represent $\mathbf{Q}_{\mathrm{i}}, \mathbf{V}, \mathbf{K}$ as follows:

$$\mathbf{Q}_{\mathrm{i}} = \mathbf{W}_q \mathbf{h}_{\mathrm{e,i}},$$

$$\mathbf{K} = \mathbf{W}_k \mathbf{C}_{\mathrm{e}},$$

$$\mathbf{V} = \mathbf{W}_v (\mathbf{C}_{\mathrm{e}} \oplus \mathbf{vec}_{\mathrm{e}}),$$

where $\mathbf{W}_q \in \mathbb{R}^{d\times d}$, $\mathbf{W}_k \in \mathbb{R}^{d\times d}$ and $\mathbf{W}_v \in \mathbb{R}^{2d\times d}$.

The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by dot-product function of the query with the corresponding key. More specifically, attention function can be represented as follows:

$$\mathbf{a}_{\mathrm{i}} = \mathrm{softmax}(\frac{\mathbf{Q}_{\mathrm{i}}\mathbf{K}^T}{\sqrt{d}}) \tag{7.4}$$

$$\text{Attention}(\mathbf{Q}_\text{i}, \mathbf{K}, \mathbf{V}) := a_\text{i}\mathbf{V}. \tag{7.5}$$

Limitation of Soft-Attention. The attention mechanism with soft weight values is categorized into soft-attention. However, soft-attention cannot effectively trim irrelevant data especially when we have a context set with an imbalanced class distribution shown in Fig. 7.2. Moreover, we show a case study in the experimental section for a better illustration.

Hard-Attention. To overcome the limitation of soft-attention, we propose to select the most related context data point instead of using weighted average. To enable argmax operation to be differentiable, we use Straight-Through (ST) Gumbel SoftMax [129] for discretely sampling the context information given target data. We introduce the sampling and $\arg\max$ approximations of ST Gumbel SoftMax procedure next.

The Gumbel-Max trick [130] provides a simple and efficient way to draw samples z from a categorical distribution with class probabilities. In our problem, for the i-th target data point $x_{\text{e,i}}$ with context set $\mathbf{X}_\text{e} \setminus \{x_{\text{e,i}}\} = \{x_{\text{e},k}\}_{k=1,k\neq\text{i}}^{K}$, the class probabilities can be obtained from the weight vector $\mathbf{a}_\text{i} = [a_{\text{i},1}, ..., a_{\text{i},K}]$ from dot-product attention mechanism according to Eq. 7.4. Because $\arg\max$ operation is not differentiable, we use the softmax function as a continuous, differentiable approximation to $\arg\max$, and generate K-dimensional sample vectors $\mathbf{P}_\text{i} = [p_{\text{i},1}, p_{\text{i},2}.., p_{\text{i},K}]$ as follows:

$$p_{\text{i},k} = \frac{\exp((\log(a_{\text{i},k}) + g)/\tau)}{\sum_{k,k\neq\text{i}}^{K} \exp((\log(a_{\text{i},k}) + g)/\tau)} \tag{7.6}$$

where τ is a temperature parameter, $g = -\log(-\log(\mu))$ is the Gumbel noise and μ is generated by a certain noise distribution (e.g., $u \sim \mathcal{N}(0, 1)$). As the softmax temperature τ approaches 0, the Gumbel-Softmax distribution becomes identical to the categorical distribution. Moreover, Straight-Through (ST) gumbel-Softmax takes different paths in the forward and backward propagation, so as to maintain sparsity yet support stochastic gradient descent. Through gumbel-softmax, the hard-attention mechanism is able to draw the most informative sample based on weight vectors from $\mathbf{P}_\text{i}$ for given target sample $x_{\text{e,i}}$.

The hard-attention can trim the irrelevant data points and select the most related data point, denoted as $\mathbf{c}_{\text{e},k} \oplus \mathbf{v}_{\text{e},k} \in \mathbb{R}^{2d}$. Besides the hard-attention mechanism, the aggregator includes an

additional fully connected layer on top of hard-attention to adjust the dimension. The $\mathbf{c}_{e,k} \oplus \mathbf{v}_{e,k}$ is fed into this fully connected layer to output context embedding $\mathbf{r}_{e,i} \in \mathbb{R}^d$.

7.3.4 Detector based on Label Embedding

Categorical characteristic of label information. The context information includes posts and their corresponding labels. The existing works like CNP [124] and ANP [125] usually simply concatenate the input features and numerical label values together as input to learn a context embedding via a neural network. Such operation discards the fact that label variables are categorical. Moreover, this operation tends to underestimate the importance of labels as the dimension of input features is usually significantly larger than that of single dimensional numerical value. To handle categorical characteristic, we propose to embed labels into fixed dimension vectors inspired by word embedding [131]. We define two embeddings $\mathbf{vec}(\mathbf{fake})$ and $\mathbf{vec}(\mathbf{real})$ for the labels of fake news and real news respectively. For example, given the k-th post $x_{e,k}$ on event e, the corresponding label is fake and its label embedding vector is $\mathbf{vec}(\mathbf{fake})$, and we denote the label embedding of $x_{e,k}$ as $\mathbf{vec}_{e,k}$. To ensure that the label embedding can capture the semantic meanings of corresponding labels, we propose to use embeddings $\mathbf{vec}(\mathbf{fake})$ and $\mathbf{vec}(\mathbf{real})$ in the detector as metrics and output predictions are determined based on metric matching.

The detector is a fully-connected layer which takes target feature vector and context embedding as inputs and outputs a vector that has the same dimensionality as that of the label embedding. More specifically, for i-th target data, the context embedding $\mathbf{r}_{e,i}$ and target feature vector $\mathbf{h}_{e,i}$ are concatenated. Then the detector takes $\mathbf{r}_{e,i} \oplus \mathbf{h}_{e,i} \in \mathbb{R}^{2d}$ as input and produces a output vector $\mathbf{o}_{e,i} \in \mathbb{R}^d$. The similarities between output $\mathbf{o}_{e,i}$ from our model and label embeddings $\mathbf{vec}(\mathbf{fake})$ and $\mathbf{vec}(\mathbf{real})$ are calculated as follows:

$$\text{similarity}(\mathbf{o}_{e,i}, \mathbf{vec}(\mathbf{fake})) = \|\mathbf{o}_{e,i} \circ \mathbf{vec}(\mathbf{fake})\|, \tag{7.7}$$

$$\text{similarity}(\mathbf{o}_{e,i}, \mathbf{vec}(\mathbf{real})) = \|\mathbf{o}_{e,i} \circ \mathbf{vec}(\mathbf{real})\|. \tag{7.8}$$

The two similarity scores are then mapped into $[0, 1]$ as probabilities via *softmax*. The trainable label embedding capture semantic meaning of labels and can generalize easily to new events with the help of adaptation step according to Eq. 7.2.

7.3.5 Algorithm Flow

After introducing the meta-learning neural process design, feature extractor, label embedding, aggregator and detector components, we present our algorithm flow.

As it can be observed from Figure 7.3, when tackling an event e, our proposed framework MetaFEND has two stages: event adaption and detection. In more details, our proposed model adapts to the specific event according to Eq. 7.2 and then the event-specific parameter is used in the fake news detection on given event. The algorithm flow is same in the two stages and we use event adaption stage as an example to illustrate this procedure.

Our input includes handful instances as context set $\{\mathbf{X}_{\text{e}}^s, \mathbf{Y}_{\text{e}}^s\} \setminus \{x_{\text{e,i}}^s, y_{\text{e,i}}^s\}$ and $x_{\text{e,i}}^s$ as target data. We first feed $\mathbf{X}_{\text{e}}^s \setminus \{x_{\text{e,i}}^s\}$ into feature extractor and get context feature representations $\mathbf{C}_{\text{e}}^s$. The context feature representations $\mathbf{C}_{\text{e}}^s$ is then concatenated with label embedding $\mathbf{vec}_{\text{e}}^s$ of $\mathbf{Y}_{\text{e}}^s$. In the target side, the target data $x_{\text{e,i}}^s$ is also fed into feature extractor to get representation as $\mathbf{h}_{\text{e,i}}^s$. The aggragator component aggregates $\mathbf{h}_{\text{e,i}}^s$, $\mathbf{C}_{\text{e}}^s$ and $\mathbf{vec}_{\text{e}}^s$ as introduced in section 7.3.3 to output context embedding $\mathbf{r}_{\text{e,i}}^s \in \mathbb{R}^d$. Then we concatenate $\mathbf{r}_{\text{e,i}}^s$ with target feature vector $\mathbf{h}_{\text{e,i}}^s \in \mathbb{R}^d$. The concatenated feature goes through the detector which is consisted of a fully connected layer to output a vector $\mathbf{o}_{\text{e,i}}^s$. The similarity scores between $\mathbf{o}_{\text{e,i}}^s$ and $\mathbf{vec}(\mathbf{fake})$, $\mathbf{vec}(\mathbf{real})$ are calculated according to Eq. 7.7 and Eq. 7.8 respectively. In the end, the similarity scores are mapped to probability values for fake news detection via softamax operation.

7.4 Experiments

In this section, we introduce the datasets used in the experiments, present the compared fake news detection models, validate the effectiveness and explore some insights of the proposed framework.

7.4.1 Datasets

To fairly evaluate the performance of the proposed model, we conduct experiments on datasets collected from two real-world social media datasets, namely Twitter and Weibo. The detailed description of the datasets are given below:

Table 7.1. The Statistics of the Datasets.

	Twitter	Weibo
# of fake News	6,934	4,050
# of real News	5,683	3,558
# of images	514	7,606

Table 7.2. The performance comparison of models for fake news detection on the Twitter and Weibo datasets under 5-shot and 10-shot settings. Accuracy and F1 score of models are followed by standard deviation. The percentage improvement (↑) of MetaFEND over the best baseline per setting is in the last row. EANN, CNP, ANP, MAML, Meta-SGD and MetaFEND share the same feature extractor as the backbone.

Method	Twitter				Weibo			
	5-Shot		10-Shot		5-Shot		10-Shot	
	Accuracy	F1 Score	Accuracy	F1 Score	Accuracy	F1 Score	Accuracy	F1 Score
VQA	73.62 ± 1.83	76.69 ± 1.23	73.49 ± 2.61	74.69 ± 2.97	76.93 ± 0.71	75.88 ± 0.45	77.80 ± 1.43	76.36 ± 1.77
attRNN	63.04 ± 2.09	60.25 ± 4.63	63.14 ± 2.00	56.60 ± 5.25	76.07 ± 1.63	74.36 ± 2.96	78.09 ± 0.58	77.69 ± 0.35
EANN	70.01 ± 3.58	72.95 ± 2.86	70.56 ± 1.00	67.77 ± 0.80	76.43 ± 0.84	74.51 ± 0.56	77.49 ± 1.95	76.56 ± 1.28
CNP	71.42 ± 2.58	72.58 ± 3.57	72.47 ± 3.61	72.11 ± 5.74	77.47 ± 5.19	77.01 ± 4.66	78.81 ± 1.57	78.07 ± 1.98
ANP	77.08 ± 2.92	79.65 ± 3.81	74.25 ± 0.76	75.16 ± 1.27	77.85 ± 1.67	76.00 ± 3.61	76.52 ± 1.84	73.73 ± 2.78
MAML	82.24 ± 1.54	82.97 ± 1.76	85.22 ± 0.64	84.98 ± 1.70	74.68 ± 0.75	74.16 ± 0.33	75.87 ± 0.33	73.41 ± 0.86
Meta-SGD	74.13 ± 2.31	75.35 ± 2.56	74.63 ± 2.46	74.57 ± 2.74	71.73 ± 1.81	69.51 ± 2.28	73.34 ± 2.35	71.42 ± 2.80
MetaFEND	**86.45 ± 1.83**	**86.21 ± 1.32**	**88.79 ± 1.27**	**88.66 ± 1.09**	**81.28 ± 0.75**	**80.19 ± 1.27**	**82.92 ± 0.13**	**82.37 ± 0.28**
(Improvement)	**(↑5.12%)**	**(↑3.91%)**	**(↑4.19%)**	**(↑4.33%)**	**(↑4.41%)**	**(↑4.13%)**	**(↑5.22%)**	**(↑5.51%)**

The **Twitter dataset** is from MediaEval Verifying Multimedia Use benchmark [104], which is used in [7], [105] for detecting fake content on Twitter. The **Weibo dataset**[1] is used in [7], [105], [132] for detecting multi-modal fake news. The news events are included in the Twitter dataset and we follow the previous works [7], [105], [132] to obtain events on Weibo via a single-pass clustering method [107]. In the two datasets above, we only keep the events which are associated with more than 20 posts and randomly split the posts on same event into support and query data. To validate performance of the models on newly emergent events, we ensure that the training and testing sets

[1] ↑https://github.com/yaqingwang/EANN-KDD18

do not contain any common event. We adopt Accuracy and F1 Score as evaluation metrics. These two datasets cover diverse news events and thus can be used as good test-grounds for evaluation of fake news detection on heterogeneous events.

7.4.2 Baselines

To validate the effectiveness of the proposed model, we choose baselines from multi-modal models and the few-shot learning models. For the multi-modal models, we fine-tune them on support set from events in the testing data for a fair comparison. In the experiments, we have the 5-shot and 10-shot settings. In our problem, 5-shot setting refers to that 5 labeled posts are provided as support set.

Fine-tune models. All the multi-modal approaches take the information from multiple modalities into account, including VQA [108], att-RNN [105] and EANN [7]. In the fine-tune setting, the training data including labeled support data and labeled query data is used to train the baselines. In the testing stage, the trained models are first fine-tuned on the labeled support data of given event, and then make predictions for testing query data. (1) **VQA** [108]. Visual Question Answering (**VQA**) model aims to answer the questions based on the given images and is used as a baseline for multimodal fake news in [105]. (2) **att-RNN** [105]. att-RNN is the state-of-the-art model for multi-modal fake news detection. It uses attention mechanism to fuse the textual, visual and social context features. In our experiments, we remove the part dealing with social context information, but the remaining parts are the same. (3) **EANN** [7]. EANN is one of the state-of-the-art models for fake news detection. It consists of three components: feature extractor, event discriminator and fake news detector. It captures shared features across different events of news to improve generlziation ability.

Few-shot learning models. We use CNP [124], ANP [125], MAML [121] and Meta-SGD [122] as few-shot learning baselines. (1) **CNP** [124]. Conditional neural process is the state-of-the-art model for few-shot learning. It combines neural network and gaussian process by using a small set of input-output pairs as context to output predication for given input of data. (2) **ANP** [125]. Attentive neural process belongs to the family of neural process which outputs prediction based on concatenation of learned distribution of context, context features and given input. (3) **MAML** [121].

Model-aganostic Meta-learning is a representative optimization-based meta-learning model. The mechanism of MAML is to learn a set of shared model parameters across different tasks which can rapidly learn novel task with a small set of labeled data. (4) **Meta-SGD** [122]. Meta-SGD is one of the state-of-the-art meta learning method for few-shot learning setting. Besides a shared global initialized parameters as with MAML, it also learns step sizes and update direction during the training procedure.

The proposed model share the same feature extractor backbone with EANN, CNP, ANP, MAML, Meta-SGD to study the effects of other designs in addition to benefits of the feature extractor backbone.

Implementations In the proposed model, the 300 dimensional FastText pre-trained word-embedding weights [133] are used to initialize the parameters of the embedding layer. The window size of filters varies from 1 to 5 for textual CNN extractor. The hidden size d_f of the fully connected layer in textual and visual extractor and dimension d are set as 16 which is searched from options {8, 16, 32, 64}. τ decays from 1 to 0.5 as the suggested way in [129]. The gradient update step is set to 1 an inner learning rate β is set to 0.1 for fine-tune models: MAML, Meta-SGD and our proposed framework MetaFEND. We implement all the deep learning baselines and the proposed framework with PyTorch 1.2 using NVIDIA Titan Xp GPU. For training models, we use Adam [134] in the default setting. The learning rate α is 0.001. We use mini-batch size of 10 and training epochs of 400.

7.4.3 Performance Comparison

Table 7.2 shows the performance of different approaches on the Twitter and Weibo datasets. We can observe that the proposed framework MetaFEND achieves the best results in terms of most of the evaluation metrics in both 5-shot and 10-shot settings.

Twitter. On the Twitter dataset in 5-shot setting, compared with CNP, ANP incorporates the attention mechanism and hence can achieve more informative context information. Due to the heterogeneity of events, it is not easy for Meta-SGD to learn a shareable learning directions and step size across all events. Thus, Meta-SGD's performance is lower than MAML's in terms of accuracy. Compared with all the baselines, MetaFEND achieves the best performance in terms of

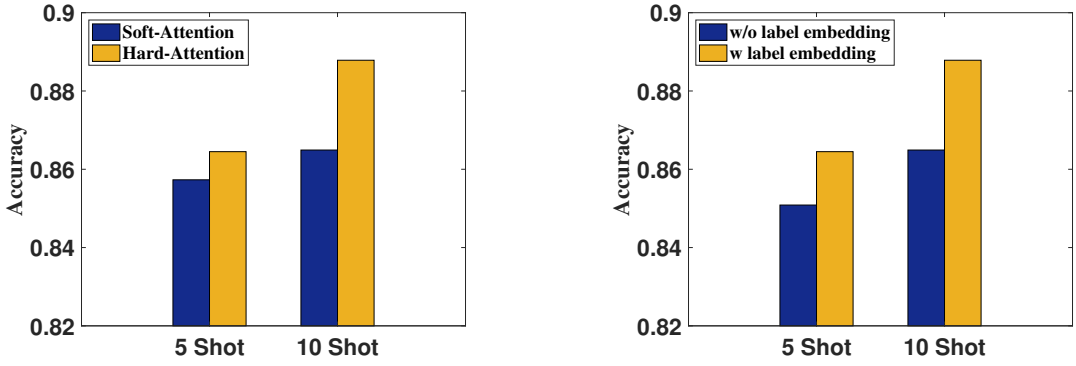
most the metrics. Our proposed model inherits the advantages of MAML to learn a set of parameters which can rapidly learn to detect fake news with a small support set. Moreover, MetaFEND can use the support data as conditioning set explicitly to better capture the uncertainty of events and thus it is able to achieve more than 5% improvement compared with MAML in terms of accuracy. In the 10-shot setting, as the size of give support data increases, the soft attention mechanism of ANP unavoidably incorporates the irrelevant data points. In contrast, the proposed model MetaFEND employs the hard-attention mechanism to trim irrelevant data points from context set and significantly outperforms all the baselines in terms of all the metrics.

Weibo. Compared with the Twitter data, the Weibo dataset has different characteristics. On the Weibo dataset, most of the posts are associated with different images. Thus, we can evaluate the performance of models under the circumstance where support datasets do not include direct clues with query set. As EANN tends to ignore event-specific features, it achieves the lowest accuracy among fine-tune models in 10-shot setting. For the few-shot models, ANP and CNP achieves better performance compared with gradient-based meta-learning methods MAML and Meta-SGD. This is because the parameter adaptation may not be effective when support data set and query set do not share the same patterns. Compared with ANP in 5-shot setting, our proposed method MetaFEND achieves 4.39% improvement in terms of accuracy and 5.51% improvement in terms of F1 score. The reason is that our MetaFEND can learn a base parameter which can rapidly learn to use a few examples as reference information for fake news detection. Thus, our proposed model enjoys the benefits of neural process and meta-learning model families.

7.4.4 Ablation Study

We show ablation study to analyze the role of Hard-Attention and label embedding components.

Soft-Attention v.s. Hard-Attention. To intuitively illustrate the role of hard-attention mechanism in the proposed model, we show ablation study by replacing hard-attention with soft-attention. Then we repeatedly run the new designed model on the Twitter dataset five times in 5-shot and 10-shot settings respectively and report the average of accuracy values. The results are show in the Figure 7.4. From Figure 7.4a, we can observe that accuracy scores of "Hard-Attention" in 5-shot and 10-shot settings are greater than those of "Soft-Attention" respectively. As the number



(a) Attention (b) Label Embedding

Figure 7.4. The ablation study about (a) Soft-Attention and Hard-Attention and (b) Label Embedding.

of support set increases, hard-attention mechanism does not have the limitation of soft-attention mechanism which unavoidably incorporates unrelated data points and significantly outperforms the soft-attention in terms of accuracy score. Thus, we can conclude that hard-attention mechanism can take effectively advantage of support set, and the superiority is more significant as we enlarge size of support set.

w/o Label Embedding v.s. w/ Label Embedding. To analyze the role of label embedding in the proposed model, we design MetaFEND's corresponding reduced model by replacing label embedding with label value 0 or 1. Accordingly, we change the multiplication between output with label embedding to a binary-class fully connected layer to directly output the probabilities. Figure 7.4b shows the results in terms of accuracy score. In Figure 7.4b, "w/o label embedding" denotes that we remove the label embedding, and "w label embedding" denotes the original approach. We can observe that the accuracy score of "w label embedding" is greater than "w/o label embedding" in 5-shot and 10-shot settings, demonstrating the effectiveness of label embedding

7.4.5 Case Study

In order to illustrate the challenges of emergent fake news detection and how our model handles challenges, we show one example in 5-shot learning setting as case study in Fig. 7.5. As it can be observed, the four of five news examples in the support set are real news. Due to imbalanced class condition in the support set, it is difficult for Soft-Attention to provide correct prediction for

news of interest in the query set. More specifically, Fig. 7.5 shows the attention score values (red color) between examples in support set and query set based on multi-modal features. Although the first example with largest attention score value is most similar to news example in the query set, the majority of context information is from the other four examples due to imbalanced class distribution. Such an imbalanced class distribution leads to incorrect prediction for Soft-Attention. The Hard-Attention mechanism can achieve correct result by focusing on the most similar sample in the support set. Through this example, we can also observe the necessity of event adaption stage. The posts and images for the same event are very similar and difficult to distinguish. Without event adaption stage, the model cannot capture informative clues to make correct predictions.

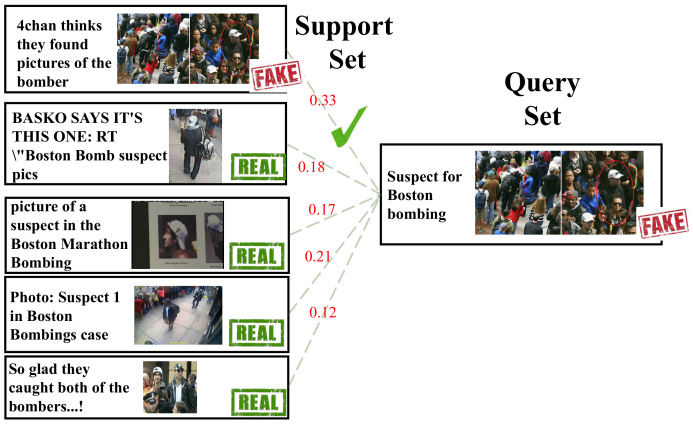


Figure 7.5. Fake news examples missed by Soft-Attention but spotted by Hard-Attention

8. ADAPTATION WITH UNLABELED DATA FOR TEXTUAL ATTRIBUTE VALIDATION

(A version of this chapter has been previously published in KDD 2020 [9].)

8.1 Introduction

Product catalogs are valuable resources for eCommerce website for the organization, standardization and publishing of product information. Because the majority of product catalogs on eCommerce websites (e.g., Amazon, Ebay, and Walmart) are contributed by individual retailers, the catalog information unavoidably contains noisy facts [135], [136]. The existence of such errors results in misleading information delivered to consumers and significantly downgrades the performance of downstream applications, such as product recommendation. As the magnitude of product catalogs does not allow for manual validation, there is an urgent need for the development of automatic yet effective validation algorithms.

In a product catalog, a product is typically associated with multiple textual attributes, such as name, brand, functionality and flavor, whose values are short texts. Therefore, in this work, we focus on the important task of validating the correctness of a textual attribute value given a product. A real example is "Ben & Jerry's - Vermont's Finest Ice Cream, Non-GMO - Fairtrade - Cage-Free Eggs - Caring Dairy - Responsibly Sourced Packaging, Americone Dream, Pint (8 Count)", which is the product title of an icecream on Amazon. The attribute "flavor" is a textual attribute, and for this particular icecream, "Americone Dream" is its flavor attribute value. The objective is to automatically output whether this value is correct or not for this product.

One may consider to model this task as anomaly detection based on the values of the target textual attribute, so that anomalies correspond to wrong values. However, this solution is not applicable to the validation task because: 1) As individual retailers self-report these attribute values, the set of possible values cannot be predetermined, and thus traditional anomaly detection approaches cannot work. 2) Textual anomaly detection has been studied and many methods have been proposed to identify anomalies by extracting distinguishing features from the texts. However, in the validation task, the correctness of a value is highly dependent on the product. For example,

“Americone dream" may not be a common piece of textual value, but it is a correct flavor name for Ben&Jerry icecream.

Motivated by this observation, we propose to verify the correctness of textual attribute value against the text description of the corresponding product. A detailed description of a product can be found from the product webpage, which contains rich information about many attributes of the product. For example, in our example, the title itself already covers the values of several attributes, such as flavor and ingredients. By cross-checking the textual attribute value “Americone-dream" for flavor against this description, we can easily verify that this value is correct. However, this cross-checking cannot be completed by a simple matching of the keywords. We found that a certain amount of errors are because the retailers often abuse the attribute by filling a real value of another attribute. Such errors cannot be detected by simply matching the value with product description text as they indeed can be found there. For example, for value “Non-GMO", it is a wrong value as of flavor, but could be labeled as correct by a simple matching against the product title of this icecream.

Therefore, we propose to model the validation problem as the task of automatic correctness inference based on an input of a textual attribute value and the description of the corresponding product. This setting is related to the natural language inference (NLI) task, which automatically determines if a hypothesis is true or false based on a text statement. Recently, powerful neural network based models, such as Transformer [137] and BERT [1] have shown promising performance towards NLI task. However, their success relies on sufficient high-quality labeled data, which requires the annotation of correctness on a large number of hypothesis-statement pairs. This requirement cannot be satisfied in the textual attribute validation task. There are thousands to millions product categories on eCommerce website, and thus annotating sufficient labeled data for all the categories is impossible. If only limited categories are annotated, such labeled data cannot be applied to other categories. For products in different categories, the product attributes and the vocabularies of the attributes could vary significantly. For example, even for the same attribute “flavor”, there is no overlapping values when describing the flavor of *seasoning*, *ice cream* and *coffee*.

To tackle the aforementioned challenges, we propose a novel meta-learning latent variable approach, namely MetaBridge, for textual attribute validation. The proposed approach effectively

leverages a small set of labeled data in limited categories for training category-agnostic models, and utilizes unlabeled data to capture category-specific information. More specifically, the proposed objective function is directly derived from the textual attribute validation task based evidence lower bound, and it seamlessly integrates meta-learning principle and latent variable modeling. We then propose to solve this problem via a stochastic neural network which has the sampling and parameter adaptation steps. The benefits of the proposed approach include the following. First, the parameter adaptation step allows more parameter flexibility to capture category-specific information. Second, we enforce the distribution consistences between unlabeled and labeled data via KL Divergence, which makes best use of limited labeled information while extracts most useful information from unlabeled data. Third, the proposed model is a stochastic neural network where sampling step is beneficial to the prevention of overfitting. The insights behind our objective function are explored in our experiments. Experimental results on two large real-world datasets show that proposed model can effectively generalize to new product categories and outperforms the state-of-the-art approaches.

8.2 Problem Setting and Preliminary

In this section, we first introduce our problem and the few-shot learning setting, then we present the representative algorithm of meta-learning, its limitations and our intuitions.

8.2.1 Problem Setting

Given a set of product profiles presented as unstructured text data like titles and their corresponding textual attribute values, our objective is to identify incorrect attribute values based on corresponding product profiles. Note that we have open world assumption thus we cannot construct a golden list to filter out never-seen attribute values. As the the categories of product are from thousands to millions and annotation job requires corresponding knowledge, we can only obtain a small set of annotated data about a subset of product categories. But for each category, unlabeled data are easily collected. We next formally define the problem we are solving.

Definition 8.2.1. *Given a set of product categories C and corresponding products $I = \{I_c : c \in C\}$, product profiles $P = \{p_{\mathrm{i}} : \mathrm{i} \in I\}$, attribute values as $V = \{v_{\mathrm{i}} : \mathrm{i} \in I\}$, we aim to identify $X = (P, V)$ pair that are incorrect for product I.*

After defining our problem, we introduce our learning setting. Following the *few-shot learning* setting [138], in each category $c \sim C$, we have a few unlabeled examples $x_c^s = \{x_{c,\mathrm{i}}^s\}_{\mathrm{i}=1}^N$ to constitute the support set $\mathcal{D}_c^s$ and have a small set of labeled examples $\{x_c^q, y_c^q\} = \{x_{c,\mathrm{i}}^q, y_{c,\mathrm{i}}^q\}_{\mathrm{i}=N+1}^{N+K}$ as the query set $\mathcal{D}_c^q$. We need to learn from a subset of categories a well-generalized model which can facilitate training in a new category based on unlabeled support set $\mathcal{D}_c^s$ and infer the correctness of attribute values for corresponding products I_c in the same category c.

8.2.2 MAML

We give an overview of Model-Agnostic Meta-Learning method [121] which is a representative algorithm of optimization-based meta-learning approaches. First, we use our problem as an example to introduce the general learning setting of meta-learning methods. The learning of meta-learning are split into two stages: meta-training and meta-testing. During the meta-training stage, the baseline learner f_θ with parameter set θ will be adapted to specific category c as f_{θ_c} with the help of meta-learner $M(\cdot)$ on support set $\mathcal{D}_c^s$, i.e., $\theta_c = M(\theta, \mathcal{D}_c^s)$. Such category specific learner f_{θ_c} is evaluated on the corresponding query set $\mathcal{D}_c^q$. During the meta-testing stage, the baseline learner f_θ will be adapted to testing category c on $\mathcal{D}_c^s$ using the same procedure with meta-training stage, i.e., $\theta_c = M(\theta, \mathcal{D}_c^s)$, and make predictions for the $\mathcal{D}_c^q$.

In the MAML, it updates parameter vector θ using one or more gradient descent updates on the category c. For example, when using one gradient update:

$$\theta_c = M(f_\theta, \mathcal{D}_c^s) = \theta - \beta \bigtriangledown_\theta \mathcal{L}(f_\theta, \mathcal{D}_c^s),$$

where β is inner step size and $\mathcal{D}_c^s$ is a support set for given category c. The model parameters are trained by optimizing for the performance of f_{θ_c} with respect to θ across categories. More concretely, the meta-objective is as follows:

$$\min_\theta \mathcal{L}(f_\theta) = \sum_{c \in C} \mathcal{L}(f_{\theta - \beta \bigtriangledown_\theta \mathcal{L}(f_\theta, \mathcal{D}_c^s)}, \mathcal{D}_c^q),$$

where $\mathcal{D}_c^q$ is a query set for given category c.

Limitations: MAML captures category uncertainty with the help of a few labeled data. Such mechanism brings expensive and continuous annotation costs. Although we can change the supervised loss on support set to unsupervised loss like entropy minimization, the adaptation on unlabeled data will undoubtedly increase the difficulty of capturing category uncertainty and further degrade the performance. Moreover, meta-learning methods suffer from overfitting problem especially when only a small set of labeled data is available.

Key ideas of our solution: To avoid continuous annotation cost, we expect our model to capture the category-uncertainty via unlabeled data. Thus, how we take advantage of unlabeled data to benefit our method is a key problem. A simple intuition is that we need to bridge unlabeled data and labeled data together to stabilize adaptation step. To achieve such goal, we propose a new approach which can integrate latent variable model with meta-learning framework. The latent variable model can capture the category distribution via a latent variable which can construct a connection between unlabeled and labeled data and prevents overfitting with the inherent sampling procedure.

8.3 Methodology

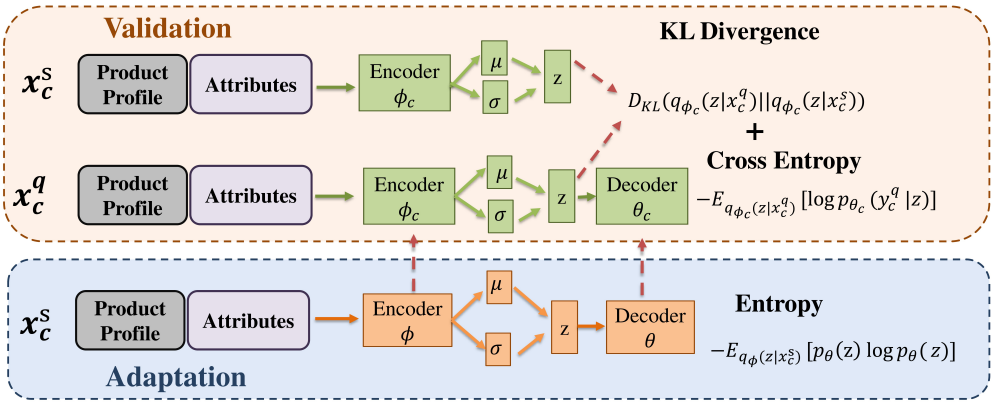


Figure 8.1. The proposed approach MetaBridge. The proposed approach mainly includes two stages: adaptation and Validation. During the adaptation stage, the model parameter Θ is updated to Θ_c accordingly to capture the uncertainty of category c. During the validation stage, the adapted model Θ_c is used to validate textual attributes for products on the category c.

In this section, we first introduce how we derive our meta-learning latent variable objective function, then we present our model architecture and the algorithm flow.

8.3.1 Overview

As shown in Figure 8.1, the proposed MetaBridge mainly includes two stages: adaptation and validation. During the adaptation stage, the model parameter is updated on unlabeled support data from given product category; during the validation stage, the category-specific model is used to make textual validation for products from same product category. To capture uncertainty on unlabeled data and prevent overfitting, we propose a meta learning latent variable objective function which includes two terms: inference loss and bridging regularizer. By jointly minimizing both objectives, we enforce the model i) to learn direct signal from labeled data, and ii) internally harmonizes the latent structures of the new category and existing category from unlabeled data. More specifically, the proposed approach is a stochastic neural network which includes sampling and parameter adaptation steps. Furthermore, the proposed model can enforce the distribution consistency between unlabeled and labeled data via KL Divergence. Thus, we are able to train a complicated meta learning Transformer-based model which can jointly processes signals from textual product description and attribute values to conduct effective inference.

8.3.2 Latent Variable Model

The goal of the proposed algorithm is to learn to infer on various categories even unseen category with a handful unlabeled training instances. More specifically, for the c-th category, the corresponding support set x_c^s is given, we aim to infer y_c^q based on x_c^q. Here We denote $x_c = \{x_c^s, x_c^q\}, y_c = \{y_c^q\}$ for simplicity and hence our objective function can be represented as follows:

$$\log p_\Theta(y|x) = \sum_{c \in C} \log p_\Theta(y_c|x_c), \tag{8.1}$$

where Θ represents the parameter set of the proposed model. For each category c, we only have a very limited number of labeled data points. To capture category uncertainty, we include a latent variable z that captures category distribution. This latent variable is of particular interest because

it can capture the category uncertainty and allows us to sample data for the learned category to prevent overfitting.

To be clear, we take c-th category as an example. Let $p(z, y_c|x_c)$ be a joint distribution over a set of latent variables $z \sim Z$ and observed variables $y_c \in Y$ and $x_c \in X$ for category c. An inference query involves computing posterior beliefs after incorporating evidence into the prior: $p(z|y_c, x_c) = p(z, y_c|x_c)/p(y_c|x_c)$. This quantity is often intractable to compute as the marginal likelihood $p(y_c|x_c) = \int_z p(z, y_c|x_c)dz$ requires integrating or summing over a potentially exponential number of configurations for z. As with variational autoencoders [139], we approximate the objective function using the evidence lower bound (ELBO) on the log likelihood. For the purpose of calculating ELBO, let us introduce an encoder model $q_\phi(z|x_c, y_c)$: an approximation to the intractable true posterior $p(z|x_c, y_c)$ with a parameter set ϕ. In a similar vein, we use a decoder model $p_\theta(y_c|x_c, z)$ to approximate the intractable true posterior $p(y_c|x_c, z)$ with a parameter set θ. Thus, the parameter set Θ includes $\{\phi, \theta\}$. After introducing the encoder and decoder, we present how to derive our objective function based on ELBO.

Evidence Lower Bound (ELBO) The ELBO can be shown to decompose into

$$\begin{aligned} &\log p_\Theta(y_c|x_c) \\ \geq &\mathbb{E}_{q_\phi(z|x_c,y_c)}[\log p_\theta(y_c|z, x_c)] - D_{KL}(q_\phi(z|x_c, y_c) \,||\, p(z)). \end{aligned} \tag{8.2}$$

To better reflect the desired model behavior at test time, i.e., we have a handful training instances as a support set x_c^s for each category, we explicitly split x_c into support and query sets. Our goal is to model the conditional of the query set given the support set. Thus, instead of using prior $p(z)$ in Eq. 8.2, we propose to use a more informative conditional prior distribution $p(z|x_c^s)$ as with [140] and further rewrite our objective function as follows:

$$\begin{aligned} &\log p_\Theta(y_c|x_c) \\ =&\log p_\Theta(y_c^q|x_c^s, x_c^q) \\ \geq &\mathbb{E}_{q_\phi(z|x_c^s,x_c^q,y_c^q)}[\log p_\theta(y_c^q|z, x_c^s, x_c^q)] \\ &- D_{KL}(q_\phi(z|x_c^q, x_c^s, y_c^q) \,||\, p(z|x_c^s)) \end{aligned} \tag{8.3}$$

For the encoder $q_\phi(z|x_c^s, x_c^q, y_c^q)$, since x_c^q is given and y_c^q is implicitly encoded into parameter set ϕ, we assume z is conditional independent with y_c^q given x_c^q and ϕ. Thus, our objective function can be simplified as follows:

$$\begin{aligned} &\log p_\Theta(y_c^q|x_c^s, x_c^q) \\ \geq &\mathbb{E}_{q_\phi(z|x_c^s, x_c^q)}[\log p_\theta(y_c^q|z, x_c^s, x_c^q)] \\ &- D_{KL}(q_\phi(z|x_c^s, x_c^q) \,||\, p(z|x_c^s)) \end{aligned} \tag{8.4}$$

The support set x_c^s is used to help the proposed model to quickly adapt to new category. Thus, how we take advantage of this set to benefit our framework is a key problem. To tackle this problem, we propose to encode the information from support set into our parameter inspired by MAML [121] and further we can obtain a category-specific model to accelerate unseen category adaptation. We will introduce how to incorporate information from support set into our framework via parameter adaptation in the next subsection.

8.3.3 Parameter Adaptation

As introduced in the subsection 8.2.2, MAML obtains a category specific parameter set using one or more gradient descent updates based on loss from support set x_c^s. Considering the support set in our problem is unlabeled, we redefine the loss function on unlabeled support set by entropy minimization. Entropy minimization encourages the confidence of predictions and is commonly used in the semi-supervised learning [117], [141], [142] and domain adaptation [143]–[145]. More concretely, the loss function $\mathcal{L}_s^c$ on the support set x_c^s is defined by entropy as follows:

$$\mathcal{L}_s^c(\theta, \phi, x_c^s) = -\mathbb{E}_{q_\phi(z|x_c^s)}[p_\theta(z) \log p_\theta(z)] \tag{8.5}$$

and the parameter adaptation step via one step of gradient descent is defined accordingly as follows:

$$\{\theta_c, \phi_c\} = \{\theta, \phi\} - \beta \bigtriangledown_{\theta,\phi} \mathcal{L}_s^c(\theta, \phi, x_c^s). \tag{8.6}$$

Here we assume the information of support set is encoded into parameter via gradient descent and then exclude the x_c^s from conditionals. Moreover, for the decoder $p_\theta(y_c^q|z, x_c^s, x_c^q)$, y_c^q is conditional

independent with x_c^q given z since z is the feature representation of x_c^q. Thus, we can have simpler equations as follows:

$$\textbf{Encoder:} \quad q_\phi(z|x_c^s, x_c^q) \rightarrow q_{\phi_c}(z|x_c^q) \tag{8.7}$$

$$\textbf{Decoder:} \quad p_\theta(y_c^q|z, x_c^s, x_c^q) \rightarrow p_{\theta_c}(y_c^q|z) \tag{8.8}$$

8.3.4 Objective Function

To optimize our objective function, we still need to approximate conditional prior $p_\theta(z|x_c^s)$ which is intractable. As the parameter adaptation step can encode support set into the model and captures category specific information, hence we propose to use $q_{\phi_c}(z|x_c^s)$ as a approximation to $p(z|x_c^s)$ and then we have our final objective function as follows:

$$\begin{aligned}
&\log p_\Theta(y_c^q|x_c^s, x_c^q) \\
\geq &\mathbb{E}_{q_{\phi_c}(z|x_c^q)}[\log p_{\theta_c}(y_c^q|z)] \\
&- D_{KL}(q_{\phi_c}(z|x_c^q) \mid\mid p(z|x_c^s)) \\
\simeq &\mathbb{E}_{q_{\phi_c}(z|x_c^q)}[\log p_{\theta_c}(y_c^q|z)] \\
&- D_{KL}(q_{\phi_c}(z|x_c^q) \mid\mid q_{\phi_c}(z|x_c^s))
\end{aligned} \tag{8.9}$$

The objective function includes two terms: the first term is our supervised inference loss on query samples and the second term is to enforce conditional category distribution $q_{\phi_c}(z|x_c^q)$ consistent with conditional distribution $q_{\phi_c}(z|x_c^s)$, i.e., distributions of unlabeled and labeled data from same category. The second term can be treated as a explicit bridge between support set and query set. λ is a hyper-parameter that needs to be set. We explore the impact of λ in the experiment section 8.4.5.

$$\mathcal{L}_q^c = \underbrace{-\mathbb{E}_{q_{\phi_c}(z|x_c^q)}[\log p_{\theta_c}(y_c^q|z)]}_{\textbf{Inference Loss}} + \lambda \underbrace{D_{KL}(q_{\phi_c}(z|x_c^q) \mid\mid q_{\phi_c}(z|x_c^s))}_{\textbf{Bridging Regularizer}} \tag{8.10}$$

In this work, we assume $q_{\phi_c}(z|x_c^q)$ and $q_{\phi_c}(z|x_c^s)$ follow multivariate normal distributions $\mathcal{N}(\mu(x_c^q), \sigma^2(x_c^q)\mathbf{I})$ and $\mathcal{N}(\mu(x_c^s), \sigma^2(x_c^s)\mathbf{I})$ respectively. The KL Divergence $D_{KL}(q_{\phi_c}(z|x_c^q) \,||\, q_\phi(z|x_c^s))$ in Eq. 8.10 can be analytically integrated:

$$\begin{aligned} &D_{KL}(q_{\phi_c}(z|x_c^q) \,||\, q_{\phi_c}(z|x_c^s)) \\ =&\sum_{\mathrm{j}=1}^{d}\left(\log\frac{\sigma_{\mathrm{j}}(x_c^s)}{\sigma_{\mathrm{j}}(x_c^q)} + \frac{\sigma_{\mathrm{j}}^2(x_c^q) + (\mu_{\mathrm{j}}(x_c^q) - \mu_{\mathrm{j}}(x_c^s))^2}{2\sigma_{\mathrm{j}}^2(x_c^s)} - \frac{1}{2}\right), \end{aligned} \tag{8.11}$$

where d is the dimension of z. Thus, we only need to calculate category loss term. To enable distribution $q_{\phi_c}(z|x_c^q)$ differentiable, we follow previous work [139], [146], [147] to use reparameterization trick to parameterize z.

Reparameterization Trick Instead of directly sampling from a complex distribution, we can reparametrize the random variable as a deterministic transformation of an auxiliary noise variable ϵ. In our case, to sample from $q_{\phi_c}(z|x_c^q)$, since $q_{\phi_c}(z|x_c^q) = \mathcal{N}(\mu(x_c^q), \sigma^2(x_c^q)\mathbf{I})$, one can draw samples by computing $z = \mu(x_c^q) + \sigma(x_c^q) \odot \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $\odot$ signify an element-wise product. By passing in auxiliary noise, our proposed model is stochastic and if we do not pass in any auxiliary noise, then the model is deterministic.

After introducing our final objective function, we will present the detailed architecture and algorithm flow in the next subsections.

8.3.5 Model Architecture

Our model mainly includes two components: encoder and decoder.

Encoder The encoder in use is Transformer [137], which is a context-aware model and has been proven powerful in textual classification. The transformer takes a sequence of word tokens as input. In our problem, the input includes two parts: unstructured product profiles and the corresponding product textual attribute values. As the length of two parts are usually very different, we use two Transformers to take two parts separately to obtain fixed-dimensional features. Following [1], the first token of every sequence is always a special classification token ([CLS]). Accordingly, the final hidden state corresponding to this token is used as the aggregate sequence representation. We concatenate the two final hidden states from Transformers and then feed them into two fully

connected layers with weight matrix $\mathbf{W}_{\mu}^{2d\times d}$ and $\mathbf{W}_{\sigma}^{2d\times d}$ to output mean μ and $\log(\sigma)$ as suggested in [139].

Decoder The decoder is a fully connected layer with weight matrix $\mathbf{W}_{o}^{d\times 2}$ to take samples from inferred normal distribution and output the probability of given attribute values being incorrect.

8.3.6 Training and inference procedures

We first sample a batch of categories and get corresponding support set and query set for each category. Given the support set , we first update the parameter of encoder and decoder to get category-specific parameter set θ_c, ϕ_c according to Eq. 8.5 and Eq. 8.6. The category-specific encoder takes query set x_c^q and support set x_c^s to output the parameters for the distribution $p(z|x_c^q)$ and $p(z|x_c^s)$ respectively. Then we can calculate the Bridging Regularizer in the Eq. 8.10. We then sample $z's$ from the posterior $p(z|x_{t,\mathrm{i}}^q)$ and the category-specific decoder takes $z's$ as input to infer the correctness of attribute values. Thus, our model is stochastic during the training stage. During the testing stage, the inference procedure is similar with it in the training procedure, the only difference is that for any data query data $x_{c,\mathrm{i}}^q$, its inferred latent code is set to be the conditional mean $\mu(x_{c,\mathrm{i}}^q) = \mathbb{E}_{q_\phi(z|x_{c,\mathrm{i}}^q)}[z]$ and the category-specific decoder takes $u(x_{c,\mathrm{i}}^q)$ as input. In other words, we use the deterministic model in the testing stage to obtain stable inference results without sampling step.

8.4 Experiments

In this section, we introduce the dataset used in the experiments, present the compared state-of-the-art baseline models, validate the effectiveness and explore insights of the proposed approach.

8.4.1 Datasets

To fairly evaluate the performance of the proposed approach, we use two internal Amazon datasets on attributes *Flavor* and *Ingredient* respectively. The products in the two datatset are from thousands of product categories across different domains. When preprocessing the datasets, we first exclude the products which do not have the attribute of interest. Then we randomly select 100 products as support set and randomly select 10 products from the rest as query set in each category.

We send query set to ask Amazon Mturkers to identify the correctness of attribute values based on corresponding product profiles. Each data point is annotated by 3 Amazon Mturkers and the final label is decided by majority voting. To evaluate the performance of attribute validation models for never-seen product categories, we split the datasets into the training, validation, testing sets according to their product categories. Thus, we ensure that they do not contain any common product category. To evaluate the performance of models under a small data setting, we only use a small portion of product categories for training purpose and the number of product category in training, validation and testing are in a 3:1:6 ratio. The detailed statistics are shown in Table 8.1.

Table 8.1. The Statistics of the Amazon Datasets.

Dataset	# of Product Categories	# of unlabeled Data	# of labeled Data
Flavor	321	32,100	3,210
Ingredient	658	65,800	6,580

8.4.2 Experimental Setup

Metric. We use Precision-Recall AUC (PR AUC) and Recall@Precision (R@P) to evaluate the performance of the models. PR AUC is defined as the area under the precision-recall curve. Such a metric is a useful measurement of prediction when the classes are imbalanced. R@P is defined as the recall value at a given precision. Such a measure is widely used to evaluate the model performance when a specific precision requirement need to be satisfied.

Baselines. To validate the effectiveness of the proposed model, we choose baselines from the following three categories: supervised learning, fine-tune and meta-learning settings.

•**Supervised Learning** We use Logistic Regression (LR), Support Vector Machine (SVM) and Random Forest (RF) as baselines. The supervised learning models are only trained with labeled query data and are not updated when testing. The feature vectors are formed by concatenation of counting the frequencies of specific attribute value in the product textual description, the position of first appearance of attribute value in the description and the average of attribute value word embeddings.

•**Fine-tune** Attribute validation is related to natural language inference (NLI) problem. We select three state-of-the-art models ESIM [148], Transformer [137], BERT [1] as baselines. All

sublayers of ESIM produce the output with dimension $d = 16$ except the last output layer. For the BERT model, we use the output from BERT-base's last second layer and feed the output into a fully connected layer with weight matrix $\mathbf{W}^{768\times16}$ with ReLU activation function. Then the output goes through a fully connected layer to output inference results. In the fine-tune setting, the training data include unlabeled support data and labeled query data. We use the entropy minimization to define the loss on unlabeled data as [117] and use the cross-entropy to define the loss on labeled data. The ratio of labeled loss and unlabeled loss is set as 10:1. In the testing stage, the pre-trained model is first fine-funed on the unlabeled support data of given task with entropy minimization, and then conduct inference on testing query data.

•**Meta-Learning** We select two state-of-the-art meta learning models MAML [121] and Meta-SGD [122] as baselines. The model architectures of two baselines are identical with Transformers in fine-tune setting. The meta learning setting is that we use entropy minimization loss on unlabeled support data to adapt the parameter of models to given tasks, the task-specific parameters will be evaluated on the query data from same task during training stage. In the testing stage, the baselines is first fine-funed on the unlabeled support data with fixed steps of gradient updates and then conduct inference on the testing query data.

8.4.3 Performance Comparison

Table 8.2 shows the performance of different approaches on the *Flavor* and *Ingredient* datasets. We use 100 unlabeled data as support set and 5 labeled data as query set per product category. We can observe that that the proposed framework achieves the best results in terms of all the evaluation metrics on both datasets.

Table 8.2. The performance comparison of different methods in the Flavor and Ingredient data.

Setting	Method	Flavor					Ingredient				
		PR AUC	R@P=0.7	R@P=0.8	R@P=0.9	R@P=0.95	PR AUC	R@P=0.7	R@P=0.8	R@P=0.9	R@P=0.95
Supervised Learning	LR	0.6830 ± 0.0000	48.67 ± 0.00	23.24 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.4520 ± 0.0000	18.71 ± 0.00	14.08 ± 0.00	11.67 ± 0.00	11.47 ± 0.00
	SVM	0.6408 ± 0.0000	42.37 ± 0.00	13.56 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.3863 ± 0.0000	19.72 ± 0.00	3.22 ± 0.00	3.22 ± 0.00	3.22 ± 0.00
	RF	0.6986 ± 0.0095	43.78 ± 1.53	15.81 ± 5.88	4.43 ± 2.81	2.45 ± 2.18	0.4683 ± 0.0137	20.72 ± 1.33	16.15 ± 1.49	14.69 ± 1.06	11.07 ± 1.28
Fine-tune	RNN	0.7092 ± 0.0155	51.09 ± 5.68	34.14 ± 2.85	15.93 ± 4.09	8.35 ± 2.36	0.4388 ± 0.0134	25.88 ± 2.29	20.68 ± 2.49	14.69 ± 2.98	7.69 ± 4.61
	ESIM	0.7160 ± 0.0192	54.90 ± 5.26	38.32 ± 5.09	22.22 ± 5.92	7.69 ± 6.62	0.4412 ± 0.0199	23.30 ± 6.42	16.46 ± 6.95	8.89 ± 5.45	5.07 ± 3.88
	Transformer	0.7210 ± 0.0434	54.19 ± 10.97	34.21 ± 10.27	19.39 ± 6.72	12.86 ± 3.91	0.4890 ± 0.0203	31.47 ± 2.46	28.05 ± 2.94	22.90 ± 2.94	11.31 ± 8.77
	BERT	0.7599 ± 0.0054	63.72 ± 1.27	45.56 ± 3.86	27.76 ± 2.34	18.52 ± 2.76	0.5292 ± 0.0111	34.00 ± 1.21	28.17 ± 1.61	17.00 ± 3.92	13.08 ± 6.04
Meta-Learning	MAML	0.7486 ± 0.0128	61.07 ± 2.55	39.66 ± 3.48	22.62 ± 4.19	15.57 ± 3.71	0.5289 ± 0.0247	34.46 ± 2.43	29.73 ± 3.44	22.48 ± 6.41	16.05 ± 6.16
	Meta-SGD	0.7575 ± 0.0126	64.19 ± 3.51	42.10 ± 4.62	25.06 ± 2.83	15.01 ± 4.64	0.5312 ± 0.0141	32.80 ± 3.43	24.95 ± 1.18	22.40 ± 1.19	20.59 ± 1.34
	MetaBridge	**0.7852 ± 0.0027**	**69.49 ± 0.99**	**50.00 ± 1.86**	**30.77 ± 1.52**	**22.64 ± 2.37**	**0.5658 ± 0.0077**	**39.24 ± 1.60**	**34.57 ± 2.22**	**27.00 ± 0.82**	**21.97 ± 3.52**

On the Flavor dataset, the LR, SVM and RF achieves the similar performance compared with RNN. The results show that the traditional models can achieve comparable performance with deep learning models when a small set of labeled data is given. Among the fine-tune models, we can observe that BERT achieves the better performance compared with RNN, ESIM and Transformer. The main difference between BERT and other baselines lies in the embedding. The improvement suggests the pre-trained embedding of BERT is informative. The RNN, ESIM and Transformer use the same pre-trained fasttext word embedding layer. The comparison between the three baselines indicate that Transformer architecture can take advantage of training data effectively compared with other two baselines. For the meta-learning setting, we can observe that MAML achieves more than 2% improvement in terms of PR AUC compared with Transformer with identical structure. The reason is that MAML can achieve a base parameter which can easily adapt to new task compared with semi-supervised loss learning. Besides a good base parameter, Meta-SGD also learns update directions and learning rates during training procedure. Thus, Meta-SGD achieves better performance compared with vanilla MAML. It is worth noting that the Meta-SGD achieves comparable performance with the best baseline BERT but uses much less parameters. The proposed approach MetaBridge achieves 3.66% improvement over Meta-SGD and 3.33% compared with BERT respectively in terms of PR AUC. The improvement can also be observed from recall at given precision. Since R@P=0.8 is similar with annotators' precision, we also compare the approaches in terms of this metric. The proposed framework achieves more than 10% improvement compared with best baseline BERT in terms of R@P=0.8.

On the Ingredient dataset, the RF achieves better performance compared with deep learning models RNN and ESIM. This further reveals the challenges of deep learning model in the small data learning setting. Among fine-tuned models, similar results can be observed as those in the Flavor dataset. BERT achieves the best performance compared with other fine-tuned models. This result confirms the effectiveness of pre-trained procedure in the small data learning setting. However, a contradict result with Flavor dataset can be observed from comparison between BERT and Meta-learning models. The MAML and Meta-SGD achieves the comparable and even better performance with BERT. The reason is that the vocabularies of ingredients are rarely used in other contexts hence the information is difficult to be captured without training on the given task dataset. This improvement shows the potentials of meta-learning models for the downstream tasks, which needs

models to rapidly learn with a small set of data. Accordingly, the proposed framework achieves 6.98% improvement in terms of PR AUC compared with BERT. Compared with best baseline Meta-SGD, the proposed framework achieves 6.51% in terms of PR AUC. The similar improvement can be also observed from performance comparison on R@P=0.8, the proposed framework improves more than 16% compared with the second best result. Furthermore, we can observe that the proposed MetaBridge achieves the best performance compared with all the baselines.

8.4.4 Ablation Study

Compared with MAML, our derived objective function has two main differences: stochastic characteristic and KL Divergence between support and query data. Thus, we are interested in their roles in the performance improvements. As introduced in the Section 8.3.4, we cannot simply remove one of them considering the KL Divergence and sampling are tightly coupled with each other. Instead, we propose two variants of MAML as baselines to explore the role of stochastic and KL Divergence respectively. To explore the role of stochastic characteristic, we add random noise into the input to last layer of MAML and denote it as stochastic variant. To explore the role of KL Divergence, we reduce sampling step and assume that the posterior distributions of support and query data are from normal distributions with fixed variances $\mathcal{N}(\mu(x_t^s), 1)$ and $\mathcal{N}(\mu(x_t^q), 1)$. The proposed variant is denotes as KL variant.

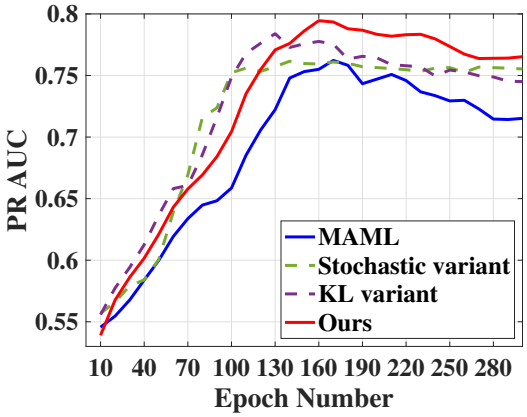


Figure 8.2. The changes of PR AUC for the models in term of the number of Epochs.

We use Flavor dataset as an example. As can be seen from Fig. 8.2, the highest PR AUC score of stochastic variant is similar with that of MAML. However, unlike MAML, the stochastic variant remains highest value without dropping. This shows that the stochastic characteristic can help prevent overfitting issue. By the comparison between KL variant and MAML, we can observe the KL variant can achieve a better PR AUC during the all training epochs. This shows the KL Divergence can construct an effective information flow between support and query data to further improve the performance. However, the KL variant simply assumes that posterior distributions are from normal distribution with fixed variances, and the over-simplistic assumption limits the potential of KL Divergence. By incorporating variances estimation, our proposed framework avoids the over-simplistic distribution assumption and can achieve better performance compared with KL variant. In overall, our proposed framework enjoys the benefits of stochastic characteristic and KL Divergence simultaneously.

8.4.5 Hyperparameter Analysis

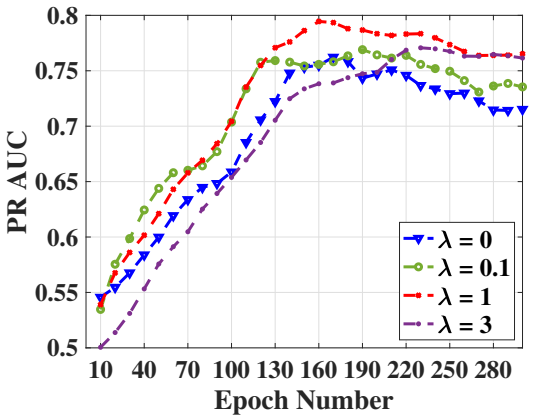


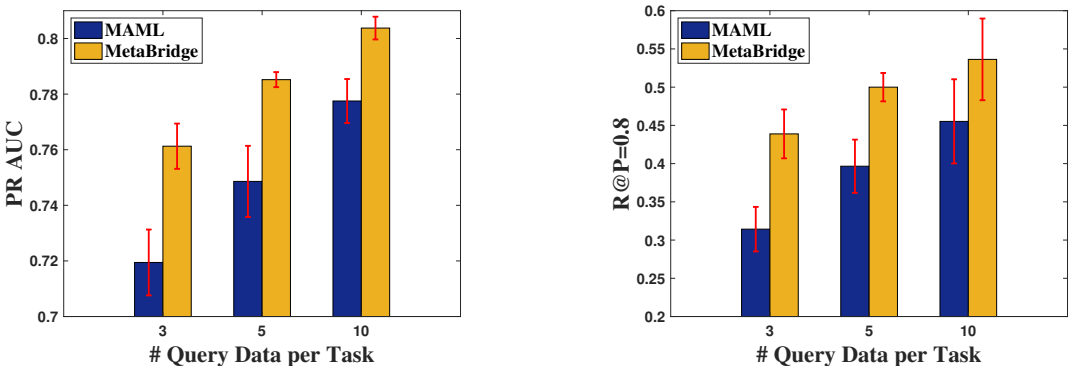
Figure 8.3. The changes of PR AUC with different λ's.

In our objective function, we use hyperparameter λ to control the strength between Inference loss and KL Divergence. In this study, we aim to explore the impact of λ in the proposed framework. We train the proposed framework using different hyperparameter λ on the Flavor Dataset. Fig. 8.3 shows the PR AUC changes of the proposed model with respect to different λ's. When λ is set to 0, the sampling procedure is removed and the model is equivalent to MAML. We can observe that

such a variant cannot effectively take advantage of unlabelled support data and the best PR AUC score is lower than that of other approach variants. And, such a variant suffers from the overfitting issue and converges to worst PR AUC value compared with other models. After changing the λ from 0 to 0.1, we can observe that the PR AUC values are stably higher than that of the variant with $\lambda = 0$. As the λ value further increases from 0.1 to 1, the proposed framework achieves significant improvement around 4% in terms of PR AUC compared with the variant $\lambda = 0$. This illustrates that our objective function can take advantage of unlabeled and small labeled data effectively and improves the generalize ability of the model. When we change value of λ to 3, the PR AUC of model increases slowly in the first 220 epochs compared with other models. But after 220 epochs, the model can archive a high PR AUC value. This further confirms the effectiveness of KL Divergence.

8.4.6 Varying Size of Labels

To analyze the impact of the query data size per product category, we train the proposed approach with different number of query data as 3, 5, 10 per category. The procedure is repeated five times and we report average performance with corresponding standard deviation. To be simple, we denote model variant by its name and number of query data. For example, the MAML which is trained with 3 query data per category is denoted as MAML_3. Figure 8.4 shows the performance comparison of the models with different number of query data in terms of PR AUC (Fig. 8.4a) and R@P=0.8 (Fig. 8.4b). When query data number is 3, our proposed framework achieves around 5.5% improvement compared with MAML_3 in terms of PR AUC. This demonstrates the effectiveness of our model with a smaller set of labeled data available. The reason is that our proposed framework can caputre category uncertainty via unlabeled data and enforce distribution consistence between unlabeled support and labeled query data. Thus, the improvement of our proposed framework over MAML is larger when the number of query data is smaller. As the number of query data increases, the performance values of MAML and our proposed framework improve significantly. This shows that meta-learning models can effectively take advantage of labeled data. For all three settings, our proposed framework shows significant improvement compared with MAML. The improvement further confirms the superiority of our proposed framework.



(a) PR AUC (b) R@P=0.8

Figure 8.4. The performance comparison of models with different numbers of query data per product category.

The similar results can be observed from Fig. 8.4b. The R@P is an important metric when we evaluate our model in the real setting. Our model achieves around 40% and 30% improvement respectively over MAML in terms of R@P=0.8 when the number of query data is set to 3 and 5. When the number of query data is set to 10, the R@P of our model is 53.6% which is higher than that of our proposed framework with 5 query data more than 6%. This reveals the potential of our model if more labeled data is available.

9. RELATED WORK

Minimally-supervised Learning. Recent works have explored unsupervised learning [1]–[3], [139], [149]–[152] and semi-supervised methods, including data augmentation [153]–[156], self-training [6], [36], [50], [87], [157], [158] and contrastive learning [159]. To alleviate the label shortage issues in real-world setting, many works explore how to incorporate external knowledge into model design especially in medical applications [24], [160]–[164]. The knowledge are usually referred to as well-curated information in structured [165]–[167] or unstructured types [23], [168]–[170]. GPT-3 [64] leverages massive scale with 175 billion parameters to obtain remarkable few-shot performance on several NLU tasks given *natural language prompt* and a few *demonstrations* for the task. Recent works [66], [88] extend this idea of *prompting* to language models like BERT [1] and RoBERTa [2].

Light-weight Learning. The standard approach to fine-tuning operate by tuning all of the trainable model parameters for every task. Recent efforts have focused on tuning large PLMs in a lightweight manner by updating a small set of parameters while keeping most of parameters in PLMs frozen, including prefix tuning [171], prompt token tuning [74] and Adapter tuning [72], [75]. All of the above works focus on fully supervised settings with thousands of labeled examples using classic fine-tuning methods. In contrast, in this work, we focus on few-shot learning settings leveraging prompts for model tuning. In the process, we make several observations regarding the design and placement of adapters in few-shot settings in contrast to its resource-rich counterpart. A contemporary work [172] pre-trains adapters with full supervision and demonstrates applications in few-shot settings.

Meta-learning has long been proposed as a form of learning that would allow systems to systematically build up and re-use knowledge across different but related tasks [5], [9], [173]. MAML [121] is to learn model initialization parameters that are used to rapidly learn novel tasks with a small set of labeled data. Following this direction, besides initialization parameters, Meta-SGD [122] learns step sizes and updates directions automatically in the training procedure. As tasks usually are different in the real setting, to handle task heterogeneity, HSML [123] customizes the global shared initialization to each cluster using a hierarchical clustering structure. The event

heterogeneity is widely observed for fake news detection, where nonexistence of hierarchical relationship in news events makes this task more challenging.

Neural process approaches [124], [125], [174] combine stochastic process and neural network to handling task heterogeneity by conditioning on a context set. Conditional Neural Process (CNP) [124] and Neural Process (NP) [174] use neural networks to take input-output pairs of support set as conditioning for inference, incorporating task specific information. However, these two works aggregate the context set by average or sum, ignoring different importance among context data samples and thereby leading to unsatisfactory performance. Attentive Neural Process (ANP) [125] incorporates attention mechanism into Neural Process to alleviate such a issue. However, ANP still suffers from underfitting issue due to fixing parameters for different tasks. Additionally, ANP directly concatenates the label numeric values with feature representation, discarding the categorical characteristics of label information.

Few-shot NER aims to build a model that can recognize a new class with a small number of labeled examples quickly. Recent works [15]–[19] exploit prototype-based methods to conduct NER tasks. Since tokens or entities belonging to the same entity class are not necessarily close to each other, prototype-based methods usually end up learning noisy prototypes and may not achieve satisfactory performance. To further improve few-shot performance, [15], [175] explores different pre-training strategies for few-shot NER, and [5], [15] propose to leverage self-training to take advantage of additional unlabelled in-domain data. Although aforementioned few-shot NER works show the potential of additional data in improving performance of few-shot NER, they still suffer from the limitations of prototype or one-hot representations of labels in transferring knowledge. Moreover, the aforementioned models cannot be applied in the zero-shot learning setting due to either reliance on labeled support set or the adoption of one-hot label representation.

Zero-shot NER is to build a model that can recognize new classes without using corresponding labeled data. This setting is rarely studied in NER task. Zero-shot NER is important and practical in the real scenario since the annotations may not be accessible due to privacy and compliance restrictions for some sensitive user applications. [176] has worked on zero-shot sequence labeling task by using attention to infer binary token-level labels. However, their token level predictions are constrained to being binary and has to rely on sentence labels. These limitations prohibit the use of this method for NER task. MRC-NER [22] formulates NER task as a machine reading

comprehension task and enables zero-shot NER. However, the inference of MRC-NER for the single sentence needs to be conducted multiple times to collect results corresponding to all the entity types of interest, incurring expensive inference cost. Moreover, the reading comprehension framework needs to be trained with large-scale dataset and is not effective in the few-shot setting. [177], [178] propose to incorporate entity description for zero-shot entity-linking task. Other zero-shot problems studied in NLP involve text classification [26], entity typing [179], word sense disambiguation [180] and relation extraction [181]. These problems have different settings and challenges compared to zero-shot NER.

Fake News Detection. Many fake news detection algorithms try to distinguish news according to their features, which can be extracted from social context and news content. (1) *Social context features* represent the user engagements of news on social media [97] such as the number of followers, hash-tag (#), propagation patterns [106] and retweets. However, social context features are very noisy, unstructured and labor intensive to collect. Especially, it cannot provide sufficient information for newly emerged events. (2) *Textual features* are statistical or semantic features extracted from text content of posts, which have been explored in many literatures of fake news detection [97], [182], [183]. Unfortunately, linguistic patterns are not yet well understood, since they are highly dependent on specific events and corresponding domain knowledge [95]. To overcome this limitation, approaches like [96], [111], [119], [120], [184] propose to use deep learning models to identify fake news and have shown the significant improvements. (3) *Visual features* have been shown to be an important indicator for fake news detection [94], [97]. The basic features of attached images in the posts are explored in the work [94], [185], [186]. we consider multi-modal features when identifying fake news on social media. To tackle *multi-modal fake news detection*, in [105], the authors propose a deep learning based fake news detection model, which extracts the multi-modal and social context features and fuses them by attention mechanism. To detect fake news on never-seen events, Wang et al. [7] propose an event-adversarial neural network (EANN) which can capture event-invariant features for fake news detection.

Attribute validation. Attribute validation task is related to anomaly detection which aims to find patterns in data that do not conform to expected behavior [187]. In the anomaly detection, the most related line of research is log anomaly detection which aims to find text, which can indicate the reasons and the nature of the failure of a system [188]. The traditional methods typically extract

features from unstructured texts and then detect anomalies based on hand-craft features. Compared with traditional learning, deep learning models have achieved an improvement in the performance of anomaly detection due to their powerful abilities [188]. The deep learning anomaly detection (DAD) approaches [189], [190] model the log data as a natural language sequence and apply RNN and CNN to detect anomalies. Attribute validation task is also related to natural language inference (NLI). NLI is a classification task where a system is asked to classify the relationship between a pair of premise and hypothesis as either entailment, contradiction or neutral. Large annotated datasets such as the Stanford Natural Language Inference [191] (SNLI) and Multi-Genre Natural Language Inference [192] (MultiNLI) corpus have promoted the development of many different neural NLI models [1], [137], [148], [193] that achieve promising performance. However, NLI task usually requires large annotated datasets for training purpose. While pre-training is beneficial, it is not optimized to allow fine-tuning with limited supervision and such models can still require large amounts of task-specific data for fine-tuning [194], [195].

10. CONCLUSIONS

Large-scale deep learning models have become the standard starting point and reaches previously unattainable performance for various tasks. To further scale deep learning, we pursue to develop systems with human learning abilities. Humans are able to learn new concepts with few examples based accumulated knowledge and adapt to unforeseen circumstances quickly. Specifically, we achieve this goal from two perspectives: few-shot learning and domain adaptation.

- **Few-shot Learning.** Conventional machine learning paradigms usually treat each class as a one-hot vector (represented by a class label), which does not carry semantic information of classes and cannot form effective supervision for model training in low resource scenarios. Meanwhile, the trained model could be highly associated with known classes and is difficult to transfer learned knowledge to novel classes. Towards this, we propose a method which could learn from semantic natural language supervision. Such a design provides a flexible and precise way to capture the semantics of entity classes and brings substantial improvement in low-resource scenarios. Even though semantic supervision signals largely improve the few-shot learning abilities, the gap between few-shot learning and full-supervised learning still exists. Then, we proposed a meta self-training framework which leverages very few manually annotated labels and a large amount of unlabeled data for model training. While self-training serves as an effective mechanism to learn from large amounts of unlabeled data via iterative knowledge exchange – meta-learning helps in adaptive sample re-weighting to mitigate error propagation from noisy pseudo-labels. Pre-trained language models (PLM) have been steadily increasing in size in terms of trainable parameters ranging from millions to billions of parameters, increasing both the computational cost and the serving cost in terms of the storage, where every task requires its customized copy of the large model parameters. We present a new fine-tuning method `LiST` that improves *few-shot learning ability* and *parameter-efficiency* over existing fine-tuning strategies.

- **Domain Adaptation.** One typical paradigm to handle domain shift is to repeat the procedures of collecting data from novel domains, re-training the model and re-deploying trained models. Such a paradigm is not only computationally expensive but time consuming. To address

those issues, we elaborate on four models which address domain adaptation with different data scenarios in target domains of interest. (1) No data in target domains. which uses event discriminator to measure the dissimilarities among different events, and further learns the event invariant features which can generalize well for the newly emerged events. (2) Weak supervision. We develop a framework, namely `WeFEND`, which can leverage users reports as weak supervision to learn new patterns from novel events for fake news detection and significantly reduce data annotation efforts and costs. Furthermore, a data selector based on reinforcement learning techniques is integrated to choose high-quality samples from the weakly labeled data and filter out those low-quality ones that may degrade the detectors performance. (3) Few labels. We propose a quick adaption model design, namely `MetaFEND`, which is able to learn new knowledge within few labels. More specifically, as the writing style, content, vocabularies and even class distributions of news on different events usually tends to differ, `MetaFEND` learns to leverage labeled data instances as conditioning, addressing limitations of meta-learning in handling heterogeneous domain distributions. (4) Unlabeled data. we propose a meta-learning latent variable approach, namely MetaBridge, for product attribute validation task. The proposed approach effectively leverages a small set of labeled data in limited product types for training and enables quick adaptation to more than thousands of types with unlabeled data.

REFERENCES

[1] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "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)*, 2019, pp. 4171–4186.

[2] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized BERT pretraining approach," *CoRR*, vol. abs/1907.11692, 2019. arXiv: 1907.11692.

[3] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," 2019.

[4] Y. Wang, H. Chu, C. Zhang, and J. Gao, "Learning from language description: Low-shot named entity recognition via decomposed framework," *arXiv preprint arXiv:2109.05357*, 2021.

[5] Y. Wang, S. Mukherjee, H. Chu, Y. Tu, M. Wu, J. Gao, and A. H. Awadallah, "Meta self-training for few-shot neural sequence labeling," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021, pp. 1737–1747.

[6] Y. Wang, S. Mukherjee, X. Liu, J. Gao, A. H. Awadallah, and J. Gao, "List: Lite self-training makes efficient few-shot learners," *arXiv preprint arXiv:2110.06274*, 2021.

[7] Y. Wang, F. Ma, Z. Jin, Y. Yuan, G. Xun, K. Jha, L. Su, and J. Gao, "Eann: Event adversarial neural networks for multi-modal fake news detection," in *Proceedings of the 24th acm sigkdd international conference on knowledge discovery & data mining*, 2018, pp. 849–857.

[8] Y. Wang, W. Yang, F. Ma, J. Xu, B. Zhong, Q. Deng, and J. Gao, "Weak supervision for fake news detection via reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

[9] Y. Wang, Y. E. Xu, X. Li, X. L. Dong, and J. Gao, "Automatic validation of textual attribute values in e-commerce catalog by learning with limited labeled data," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 2533–2541.

[10] D. Mollá, M. Van Zaanen, D. Smith, *et al.*, "Named entity recognition for question answering," 2006.

[11] J. Guo, G. Xu, X. Cheng, and H. Li, "Named entity recognition in query," in *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, 2009, pp. 267–274.

[12] D. Nadeau and S. Sekine, "A survey of named entity recognition and classification," *Lingvisticae Investigationes*, vol. 30, no. 1, pp. 3–26, 2007.

[13] A. Ritter, O. Etzioni, and S. Clark, "Open domain event extraction from twitter," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2012, pp. 1104–1112.

[14] B. Peng, C. Li, J. Li, S. Shayandeh, L. Liden, and J. Gao, "Soloist: Few-shot task-oriented dialog with a single pre-trained auto-regressive model," *arXiv preprint arXiv:2005.05298*, 2020.

[15] J. Huang, C. Li, K. Subudhi, D. Jose, S. Balakrishnan, W. Chen, B. Peng, J. Gao, and J. Han, "Few-shot named entity recognition: A comprehensive study," *arXiv preprint arXiv:2012.14978*, 2020.

[16] Y. Yang and A. Katiyar, "Simple and effective few-shot named entity recognition with structured nearest neighbor learning," *arXiv preprint arXiv:2010.02405*, 2020.

[17] Y. Hou, W. Che, Y. Lai, Z. Zhou, Y. Liu, H. Liu, and T. Liu, "Few-shot slot tagging with collapsed dependency transfer and label-enhanced task-adaptive projection network," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020, pp. 1381–1393.

[18] M. Ziyadi, Y. Sun, A. Goswami, J. Huang, and W. Chen, "Example-based named entity recognition," *ArXiv*, vol. abs/2008.10570, 2020.

[19] S. Wiseman and K. Stratos, "Label-agnostic sequence labeling by copying nearest neighbors," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 5363–5369.

[20] E. F. T. K. Sang and F. D. Meulder, "Introduction to the conll-2003 shared task: Language-independent named entity recognition," in *Seventh Conference on Natural Language Learning at HLT-NAACL 2003*.

[21] R. Weischedel, S. Pradhan, L. Ramshaw, J. Kaufman, M. Franchini, M. El-Bachouti, N. Xue, M. Palmer, J. D. Hwang, C. Bonial, *et al.*, "Ontonotes release 5.0," 2012.

[22] X. Li, J. Feng, Y. Meng, Q. Han, F. Wu, and J. Li, "A unified mrc framework for named entity recognition," *arXiv preprint arXiv:1910.11476*, 2019.

[23] D. Chen, A. Fisch, J. Weston, and A. Bordes, "Reading wikipedia to answer open-domain questions," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2017, pp. 1870–1879.

[24] M. Seo, A. Kembhavi, A. Farhadi, and H. Hajishirzi, "Bidirectional attention flow for machine comprehension," *arXiv preprint arXiv:1611.01603*, 2016.

[25] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, . Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[26] W. Yin, J. Hay, and D. Roth, "Benchmarking zero-shot text classification: Datasets, evaluation and entailment approach," 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)*, 2019, pp. 3905–3914.

[27] L. Derczynski, E. Nichols, M. van Erp, and N. Limsopatham, "Results of the wnut2017 shared task on novel and emerging entity recognition," in *Proceedings of the 3rd Workshop on Noisy User-generated Text*, 2017, pp. 140–147.

[28] J. Liu, P. Pasupat, D. Cyphers, and J. R. Glass, "Asgard: A portable architecture for multilingual dialogue systems," *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 8386–8390, 2013.

[29] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 4080–4090.

[30] A. Fritzler, V. Logacheva, and M. Kretov, "Few-shot classification in named entity recognition task," in *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, 2019, pp. 993–1000.

[31] O. Chapelle, B. Schlkopf, and A. Zien, "Semi-supervised learning," 2010.

[32] H. J. S. III, "Probability of error of some adaptive pattern-recognition machines," *IEEE Trans. Inf. Theory*, vol. 11, no. 3, pp. 363–371, 1965. DOI: 10.1109/TIT.1965.1053799.

[33] X. Li, Q. Sun, Y. Liu, Q. Zhou, S. Zheng, T.-S. Chua, and B. Schiele, "Learning to self-train for semi-supervised few-shot classification," in *Advances in Neural Information Processing Systems 32*, 2019.

[34] Q. Xie, M.-T. Luong, E. Hovy, and Q. V. Le, "Self-training with noisy student improves imagenet classification," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2020.

[35] G. Karamanolakis, S. Mukherjee, G. Zheng, and A. H. Awadallah, "Self-training with weak supervision," *arXiv preprint arXiv:2104.05514*, 2021.

[36] S. Mukherjee and A. Awadallah, "Uncertainty-aware self-training for few-shot text classification," *Advances in Neural Information Processing Systems*, 2020.

[37] S. Ruder and B. Plank, "Strong baselines for neural semi-supervised learning under domain shift," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, 2018, pp. 1044–1054.

[38] S. Petrov and R. McDonald, "Overview of the 2012 shared task on parsing the web," 2012.

[39] C. Liang, Y. Yu, H. Jiang, S. Er, R. Wang, T. Zhao, and C. Zhang, "Bond: Bert-assisted open-domain named entity recognition with distant supervision," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 1054–1064.

[40] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," in *5th International Conference on Learning Representations, ICLR 2017*, 2017.

[41] A. Shrivastava, A. Gupta, and R. Girshick, "Training region-based object detectors with online hard example mining," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 761–769.

[42] M. P. Kumar, B. Packer, and D. Koller, "Self-paced learning for latent variable models," in *Advances in Neural Information Processing Systems 23*, Curran Associates, Inc., 2010, pp. 1189–1197.

[43] M. Ren, W. Zeng, B. Yang, and R. Urtasun, "Learning to reweight examples for robust deep learning," in *International Conference on Machine Learning*, 2018, pp. 4334–4343.

[44] L. Jiang, Z. Zhou, T. Leung, L.-J. Li, and L. Fei-Fei, "Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels," in *International Conference on Machine Learning*, 2018, pp. 2304–2313.

[45] X. Pan, B. Zhang, J. May, J. Nothman, K. Knight, and H. Ji, "Cross-lingual name tagging and linking for 282 languages," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vancouver, Canada, Jul. 2017, pp. 1946–1958.

[46] S. Thrun and L. Pratt, "Learning to learn: Introduction and overview," in *Learning to learn*, 1998, pp. 3–17.

[47] B. M. Lake, T. D. Ullman, J. B. Tenenbaum, and S. J. Gershman, "Building machines that learn and think like people," *Behavioral and brain sciences*, vol. 40, 2017.

[48] M. Andrychowicz, M. Denil, S. G. Colmenarejo, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. de Freitas, "Learning to learn by gradient descent by gradient descent," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, 2016.

[49] E. F. Tjong, K. Sang, and J. Veenstra, "Representing text chunks," in *Ninth Conference of the European Chapter of the Association for Computational Linguistics*, 1999.

[50] J. He, J. Gu, J. Shen, and M. Ranzato, *Revisiting self-training for neural sequence generation*, 2019. arXiv: 1909.13788 [cs.LG].

[51] H.-S. Chang, E. G. Learned-Miller, and A. McCallum, "Active bias: Training more accurate neural networks by emphasizing high variance samples," in *Advances in Neural Information Processing Systems 30, 2017*, 2017.

[52] Y. Gal, R. Islam, and Z. Ghahramani, "Deep bayesian active learning with image data," in *Proceedings of the 34th International Conference on Machine Learning, ICML 2017*, vol. 70, PMLR, 2017, pp. 1183–1192.

[53] K. Konyushkova, R. Sznitman, and P. Fua, "Learning active learning from data," in *Advances in Neural Information Processing Systems*, 2017.

[54] P. W. Koh and P. Liang, "Understanding black-box predictions via influence functions," *arXiv preprint arXiv:1703.04730*, 2017.

[55] A. Coucke, A. Saade, A. Ball, T. Bluche, A. Caulier, D. Leroy, C. Doumouro, T. Gisselbrecht, F. Caltagirone, T. Lavril, M. Primet, and J. Dureau, "Snips voice platform: An embedded spoken language understanding system for private-by-design voice interfaces," in *Privacy in Machine Learning and Artificial Intelligence workshop, ICML2018*, 2018.

[56] K. Clark, M.-T. Luong, C. D. Manning, and Q. V. Le, "Semi-supervised sequence modeling with cross-view training," *arXiv preprint arXiv:1809.08370*, 2018.

[57] L. Chen, W. Ruan, X. Liu, and J. Lu, "Seqvat: Virtual adversarial training for semi-supervised sequence labeling," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020, pp. 8801–8811.

[58] A. Tarvainen and H. Valpola, "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results," in *5th International Conference on Learning Representations, ICLR 2017*, 2017.

[59] T. Miyato, S.-i. Maeda, M. Koyama, and S. Ishii, "Virtual adversarial training: A regularization method for supervised and semi-supervised learning," *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, 2018.

[60] A. Kumar, T. Ma, and P. Liang, "Understanding self-training for gradual domain adaptation," *arXiv preprint arXiv:2002.11361*, 2020.

[61] S. Gururangan, A. Marasovi, S. Swayamdipta, K. Lo, I. Beltagy, D. Downey, and N. A. Smith, "Don't stop pretraining: Adapt language models to domains and tasks," *arXiv preprint arXiv:2004.10964*, 2020.

[62] B. Zoph, G. Ghiasi, T.-Y. Lin, Y. Cui, H. Liu, E. D. Cubuk, and Q. Le, "Rethinking pre-training and self-training," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[63] K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning, "ELECTRA: pre-training text encoders as discriminators rather than generators," in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, OpenReview.net, 2020. [Online]. Available: https://openreview.net/forum?id=r1xMH1BtvB.

[64] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, *et al.*, "Language models are few-shot learners," *arXiv preprint arXiv:2005.14165*, 2020.

[65] P. He, X. Liu, J. Gao, and W. Chen, "Deberta: Decoding-enhanced bert with disentangled attention," in *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, OpenReview.net, 2021. [Online]. Available: https://openreview.net/forum?id=XPZIaotutsD.

[66] T. Gao, A. Fisch, and D. Chen, "Making pre-trained language models better few-shot learners," in *Association for Computational Linguistics (ACL)*, 2021.

[67] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[68] Y. LeCun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *Advances in neural information processing systems*, 1990, pp. 598–605.

[69] A. Williams, N. Nangia, and S. Bowman, "A broad-coverage challenge corpus for sentence understanding through inference," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, New Orleans, Louisiana: Association for Computational Linguistics, 2018, pp. 1112–1122. [Online]. Available: http://aclweb.org/anthology/N18-1101.

[70] T. Schick and H. Schütze, "It's not just size that matters: Small language models are also few-shot learners," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Online: Association for Computational Linguistics, Jun. 2021, pp. 2339–2352. DOI: 10.18653/v1/2021.naacl-main.185. [Online]. Available: https://aclanthology.org/2021.naacl-main.185.

[71] A. Aghajanyan, L. Zettlemoyer, and S. Gupta, "Intrinsic dimensionality explains the effectiveness of language model fine-tuning," *arXiv preprint arXiv:2012.13255*, 2020.

[72] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly, "Parameter-efficient transfer learning for nlp," in *International Conference on Machine Learning*, PMLR, 2019, pp. 2790–2799.

[73] H. Cai, C. Gan, L. Zhu, and S. Han, "Tinytl: Reduce memory, not parameters for efficient on-device learning," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[74] B. Lester, R. Al-Rfou, and N. Constant, "The power of scale for parameter-efficient prompt tuning," *CoRR*, vol. abs/2104.08691, 2021. arXiv: 2104.08691. [Online]. Available: https://arxiv.org/abs/2104.08691.

[75] J. Pfeiffer, A. Rücklé, C. Poth, A. Kamath, I. Vuli, S. Ruder, K. Cho, and I. Gurevych, "Adapterhub: A framework for adapting transformers," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP 2020): Systems Demonstrations*, Online: Association for Computational Linguistics, 2020, pp. 46–54. [Online]. Available: https://www.aclweb.org/anthology/2020.emnlp-demos.7.

[76] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, "GLUE: A multi-task benchmark and analysis platform for natural language understanding," 2019.

[77] A. Williams, N. Nangia, and S. Bowman, "A broad-coverage challenge corpus for sentence understanding through inference," 2018.

[78] I. Dagan, O. Glickman, and B. Magnini, "The PASCAL recognising textual entailment challenge," in *the First International Conference on Machine Learning Challenges: Evaluating Predictive Uncertainty Visual Object Classification, and Recognizing Textual Entailment*, 2005.

[79] R. Bar Haim, I. Dagan, B. Dolan, L. Ferro, D. Giampiccolo, B. Magnini, and I. Szpektor, "The second PASCAL recognising textual entailment challenge," 2006.

[80] D. Giampiccolo, B. Magnini, I. Dagan, and B. Dolan, "The third PASCAL recognizing textual entailment challenge," in *the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, 2007.

[81] L. Bentivogli, P. Clark, I. Dagan, and D. Giampiccolo, "The fifth PASCAL recognizing textual entailment challenge.," in *TAC*, 2009.

[82] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank."

[83] T. Zhang, F. Wu, A. Katiyar, K. Q. Weinberger, and Y. Artzi, "Revisiting few-sample BERT fine-tuning," 2021.

[84] J. Wiebe, T. Wilson, and C. Cardie, "Annotating expressions of opinions and emotions in language," *Language resources and evaluation*, vol. 39, no. 2, pp. 165–210, 2005.

[85] B. Pang and L. Lee, "A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts," 2004.

[86] E. Perez, D. Kiela, and K. Cho, "True few-shot learning with language models," *arXiv preprint arXiv:2105.11447*, 2021.

[87] Y. Wang, S. Mukherjee, H. Chu, Y. Tu, M. Wu, J. Gao, and A. H. Awadallah, "Meta self-training for few-shot neural sequence labeling," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021, pp. 1737–1747.

[88] T. Schick and H. Schütze, "Exploiting cloze-questions for few-shot text classification and natural language inference," in *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, 2021, pp. 255–269.

[89] B. Lester, R. Al-Rfou, and N. Constant, "The power of scale for parameter-efficient prompt tuning," *arXiv preprint arXiv:2104.08691*, 2021.

[90] H. Allcott and M. Gentzkow, "Social media and fake news in the 2016 election," *Journal of Economic Perspectives*, vol. 31, no. 2, pp. 211–36, 2017.

[91] M. Farajtabar, J. Yang, X. Ye, H. Xu, R. Trivedi, E. Khalil, S. Li, L. Song, and H. Zha, "Fake news mitigation via point process based intervention," *arXiv preprint arXiv:1703.07823*, 2017.

[92] N. J. Conroy, V. L. Rubin, and Y. Chen, "Automatic deception detection: Methods for finding fake news," *Proceedings of the Association for Information Science and Technology*, vol. 52, no. 1, pp. 1–4, 2015.

[93] E. Tacchini, G. Ballarin, M. L. Della Vedova, S. Moret, and L. de Alfaro, "Some like it hoax: Automated fake news detection in social networks," *arXiv preprint arXiv:1704.07506*, 2017.

[94] Z. Jin, J. Cao, Y. Zhang, J. Zhou, and Q. Tian, "Novel visual and statistical image features for microblogs news verification," *IEEE transactions on multimedia*, vol. 19, no. 3, pp. 598–608, 2017.

[95] N. Ruchansky, S. Seo, and Y. Liu, "Csi: A hybrid deep model for fake news detection," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, ACM, 2017, pp. 797–806.

[96] J. Ma, W. Gao, P. Mitra, S. Kwon, B. J. Jansen, K.-F. Wong, and M. Cha, "Detecting rumors from microblogs with recurrent neural networks.," in *IJCAI*, 2016, pp. 3818–3824.

[97] K. Shu, A. Sliva, S. Wang, J. Tang, and H. Liu, "Fake news detection on social media: A data mining perspective," *ACM SIGKDD Explorations Newsletter*, vol. 19, no. 1, pp. 22–36, 2017.

[98] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[99] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *Journal of Machine Learning Research*, vol. 12, no. Aug, pp. 2493–2537, 2011.

[100] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A convolutional neural network for modelling sentences," *arXiv preprint arXiv:1404.2188*, 2014.

[101] Y. Kim, "Convolutional neural networks for sentence classification," *arXiv preprint arXiv:1408.5882*, 2014.

[102] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[103] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," in *International Conference on Machine Learning*, 2015, pp. 1180–1189.

[104] C. Boididou, K. Andreadou, S. Papadopoulos, D.-T. Dang-Nguyen, G. Boato, M. Riegler, Y. Kompatsiaris, *et al.*, "Verifying multimedia use at mediaeval 2015.," in *MediaEval*, 2015.

[105] Z. Jin, J. Cao, H. Guo, Y. Zhang, and J. Luo, "Multimodal fusion with recurrent neural networks for rumor detection on microblogs," in *Proceedings of the 2017 ACM on Multimedia Conference*, ACM, 2017, pp. 795–816.

[106] K. Wu, S. Yang, and K. Q. Zhu, "False rumors detection on sina weibo by propagation structures," in *Data Engineering (ICDE), 2015 IEEE 31st International Conference on*, IEEE, 2015, pp. 651–662.

[107] Z. Jin, J. Cao, Y.-G. Jiang, and Y. Zhang, "News credibility evaluation on microblog with a hierarchical propagation model," in *2014 IEEE International Conference on Data Mining*, IEEE, 2014, pp. 230–239.

[108] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. Lawrence Zitnick, and D. Parikh, "Vqa: Visual question answering," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 2425–2433.

[109] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," in *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, IEEE, 2015, pp. 3156–3164.

[110] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[111] K. Popat, S. Mukherjee, A. Yates, and G. Weikum, "Declare: Debunking fake news and false claims using evidence-aware deep learning," *arXiv preprint arXiv:1809.06416*, 2018.

[112] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Proceedings of NIPS*, 2000, pp. 1057–1063.

[113] R. S. Sutton and A. G. Barto, *Introduction to reinforcement learning*. MIT press Cambridge, 1998, vol. 135.

[114] J. W. Pennebaker, R. L. Boyd, K. Jordan, and K. Blackburn, "The development and psychometric properties of liwc2015," Tech. Rep., 2015.

[115] M. Ott, Y. Choi, C. Cardie, and J. T. Hancock, "Finding deceptive opinion spam by any stretch of the imagination," in *Proceedings of ACL*, 2011, pp. 309–319.

[116] V. Pérez-Rosas, B. Kleinberg, A. Lefevre, and R. Mihalcea, "Automatic detection of fake news," *arXiv preprint arXiv:1708.07104*, 2017.

[117] Y. Grandvalet and Y. Bengio, "Semi-supervised learning by entropy minimization," in *Proceedings of NIPS*, 2005, pp. 529–536.

[118] Y. Wang, F. Ma, H. Wang, K. Jha, and J. Gao, "Multimodal emergent fake news detection via meta neural process networks," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021, pp. 3708–3716.

[119] J. Ma, W. Gao, and K.-F. Wong, "Detect rumor and stance jointly by neural multi-task learning," in *Companion Proceedings of the The Web Conference 2018*, 2018, pp. 585–593.

[120] J. Ma, W. Gao, and K.-F. Wong, "Detect rumors on twitter by promoting information campaigns with generative adversarial learning," in *The World Wide Web Conference*, 2019, pp. 3049–3055.

[121] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, JMLR. org, 2017, pp. 1126–1135.

[122] Z. Li, F. Zhou, F. Chen, and H. Li, "Meta-sgd: Learning to learn quickly for few-shot learning," *arXiv preprint arXiv:1707.09835*, 2017.

[123] H. Yao, Y. Wei, J. Huang, and Z. Li, "Hierarchically structured meta-learning," *arXiv preprint arXiv:1905.05301*, 2019.

[124] M. Garnelo, D. Rosenbaum, C. Maddison, T. Ramalho, D. Saxton, M. Shanahan, Y. W. Teh, D. Rezende, and S. A. Eslami, "Conditional neural processes," in *International Conference on Machine Learning*, 2018, pp. 1704–1713.

[125] H. Kim, A. Mnih, J. Schwarz, M. Garnelo, A. Eslami, D. Rosenbaum, O. Vinyals, and Y. W. Teh, "Attentive neural processes," *arXiv preprint arXiv:1901.05761*, 2019.

[126] H. Shen, F. Ma, X. Zhang, L. Zong, X. Liu, and W. Liang, "Discovering social spammers from multiple views," *Neurocomputing*, vol. 225, pp. 49–57, 2017.

[127] Y. Kim, "Convolutional neural networks for sentence classification," *arXiv preprint arXiv:1408.5882*, 2014.

[128] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[129] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," *arXiv preprint arXiv:1611.01144*, 2016.

[130] E. J. Gumbel, *Statistical theory of extreme values and some practical applications: a series of lectures*. US Government Printing Office, 1948, vol. 33.

[131] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[132] P. Qi, J. Cao, T. Yang, J. Guo, and J. Li, "Exploiting multi-domain visual information for fake news detection," *arXiv preprint arXiv:1908.04472*, 2019.

[133] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 2017, ISSN: 2307-387X.

[134] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[135] G. Zheng, S. Mukherjee, X. L. Dong, and F. Li, "Opentag: Open attribute value extraction from product profiles," in *KDD 2018*, 2018.

[136] G. Karamanolakis, J. Ma, and X. L. Dong, "Txtract: Taxonomy-aware knowledge extraction for thousands of product categories," *arXiv preprint arXiv:2004.13852*, 2020.

[137] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, . Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[138] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, *et al.*, "Matching networks for one shot learning," in *Advances in neural information processing systems*, 2016, pp. 3630–3638.

[139] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[140] M. Garnelo, J. Schwarz, D. Rosenbaum, F. Viola, D. J. Rezende, S. Eslami, and Y. W. Teh, "Neural processes," *arXiv preprint arXiv:1807.01622*, 2018.

[141] D.-H. Lee, "Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks," in *Workshop on Challenges in Representation Learning, ICML*, vol. 3, 2013, p. 2.

[142] D. Berthelot, N. Carlini, I. Goodfellow, N. Papernot, A. Oliver, and C. A. Raffel, "Mixmatch: A holistic approach to semi-supervised learning," in *Advances in Neural Information Processing Systems*, 2019, pp. 5050–5060.

[143] P. Morerio, J. Cavazza, and V. Murino, "Minimal-entropy correlation alignment for unsupervised deep domain adaptation," *arXiv preprint arXiv:1711.10288*, 2017.

[144] W. Jiang, C. Miao, F. Ma, S. Yao, Y. Wang, Y. Yuan, H. Xue, C. Song, X. Ma, D. Koutsonikolas, *et al.*, "Towards environment independent device free human activity recognition," in *MobiCom*, ACM, 2018, pp. 289–304.

[145] T.-H. Vu, H. Jain, M. Bucher, M. Cord, and P. Pérez, "Advent: Adversarial entropy minimization for domain adaptation in semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2517–2526.

[146] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *arXiv preprint arXiv:1308.3432*, 2013.

[147] Y. Bengio, E. Laufer, G. Alain, and J. Yosinski, "Deep generative stochastic networks trainable by backprop," in *International Conference on Machine Learning*, 2014, pp. 226–234.

[148] Q. Chen, X. Zhu, Z.-H. Ling, S. Wei, H. Jiang, and D. Inkpen, "Enhanced lstm for natural language inference," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2017, pp. 1657–1668.

[149] Y. Wang, F. Ma, L. Su, and J. Gao, "Discovering truths from distributed data," in *2017 ieee international conference on data mining (icdm)*, IEEE, 2017, pp. 505–514.

[150] Y. Li, J. Gao, C. Meng, Q. Li, L. Su, B. Zhao, W. Fan, and J. Han, "A survey on truth discovery," *ACM Sigkdd Explorations Newsletter*, vol. 17, no. 2, pp. 1–16, 2016.

[151] H. Xiao, J. Gao, Q. Li, F. Ma, L. Su, Y. Feng, and A. Zhang, "Towards confidence interval estimation in truth discovery," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 3, pp. 575–588, 2018.

[152] F. Ma, Y. Li, Q. Li, M. Qiu, J. Gao, S. Zhi, L. Su, B. Zhao, H. Ji, and J. Han, "Faitcrowd: Fine grained truth discovery for crowdsourced data aggregation," in *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, 2015, pp. 745–754.

[153] Q. Xie, Z. Dai, E. Hovy, M.-T. Luong, and Q. V. Le, *Unsupervised data augmentation for consistency training*, 2019. arXiv: 1904.12848 [cs.LG].

[154] J. Du, E. Grave, B. Gunel, V. Chaudhary, O. Celebi, M. Auli, V. Stoyanov, and A. Conneau, "Self-training improves pre-training for natural language understanding," *arXiv preprint arXiv:2010.02194*, 2020.

[155] T. Vu, M.-T. Luong, Q. V. Le, G. Simon, and M. Iyyer, "Strata: Self-training with task augmentation for better few-shot learning," *arXiv preprint arXiv:2109.06270*, 2021.

[156] F. Ma, Y. Wang, J. Gao, H. Xiao, and J. Zhou, "Rare disease prediction by generating quality-assured electronic health records," in *Proceedings of the 2020 SIAM International Conference on Data Mining*, SIAM, 2020, pp. 514–522.

[157] Z. Long, L. Che, Y. Wang, M. Ye, J. Luo, J. Wu, H. Xiao, and F. Ma, "Fedsemi: An adaptive federated semi-supervised learning framework," *arXiv e-prints*, arXiv–2012, 2020.

[158] X. L. Dong, X. He, A. Kan, X. Li, Y. Liang, J. Ma, Y. E. Xu, C. Zhang, T. Zhao, G. Blanco Saldana, *et al.*, "Autoknow: Self-driving knowledge collection for products of thousands of types," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 2724–2734.

[159] B. Gunel, J. Du, A. Conneau, and V. Stoyanov, "Supervised contrastive learning for pre-trained language model fine-tuning," in *International Conference on Learning Representations*, 2020.

[160] G. Xun, K. Jha, Y. Yuan, Y. Wang, and A. Zhang, "Meshprobenet: A self-attentive probe net for mesh indexing," *Bioinformatics*, vol. 35, no. 19, pp. 3794–3802, 2019.

[161] K. Jha, Y. Wang, G. Xun, and A. Zhang, "Interpretable word embeddings for medical domain," in *2018 IEEE international conference on data mining (ICDM)*, IEEE, 2018, pp. 1061–1066.

[162] Y. Yuan, K. Jia, F. Ma, G. Xun, Y. Wang, L. Su, and A. Zhang, "Multivariate sleep stage classification using hybrid self-attentive deep learning networks," in *2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, IEEE, 2018, pp. 963–968.

[163] Y. Yuan, G. Xun, F. Ma, Y. Wang, N. Du, K. Jia, L. Su, and A. Zhang, "Muvan: A multi-view attention network for multivariate temporal data," in *2018 IEEE International Conference on Data Mining (ICDM)*, IEEE, 2018, pp. 717–726.

[164] K. Jha, G. Xun, Y. Wang, V. Gopalakrishnan, and A. Zhang, "Concepts-bridges: Uncovering conceptual bridges based on biomedical concept evolution," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 1599–1607.

[165] H. Wang, F. Ma, Y. Wang, and J. Gao, "Knowledge-guided paraphrase identification," in *Findings of the Association for Computational Linguistics: EMNLP 2021*, 2021, pp. 843–853.

[166] H. Wang, Y. Wang, D. Lian, and J. Gao, "A lightweight knowledge graph embedding framework for efficient inference and storage," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021, pp. 1909–1918.

[167] M. Ye, S. Cui, Y. Wang, J. Luo, C. Xiao, and F. Ma, "Medpath: Augmenting health risk prediction via medical knowledge paths," in *Proceedings of the Web Conference 2021*, 2021, pp. 1397–1409.

[168] M. Ye, S. Cui, Y. Wang, J. Luo, C. Xiao, and F. Ma, "Medretriever: Target-driven interpretable health risk prediction via retrieving unstructured medical text," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021, pp. 2414–2423.

[169] K. Jha, G. Xun, Y. Wang, and A. Zhang, "Hypothesis generation from text based on co-evolution of biomedical concepts," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 843–851.

[170] F. Ma, Y. Wang, H. Xiao, Y. Yuan, R. Chitta, J. Zhou, and J. Gao, "A general framework for diagnosis prediction via incorporating medical code descriptions," in *2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, IEEE, 2018, pp. 1070–1075.

[171] X. L. Li and P. Liang, "Prefix-tuning: Optimizing continuous prompts for generation," *CoRR*, vol. abs/2101.00190, 2021. arXiv: 2101.00190. [Online]. Available: https://arxiv.org/abs/2101.00190.

[172] T. Beck, B. Bohlender, C. Viehmann, V. Hane, Y. Adamson, J. Khuri, J. Brossmann, J. Pfeiffer, and I. Gurevych, "Adapterhub playground: Simple and flexible few-shot learning with adapters," *arXiv preprint arXiv:2108.08103*, 2021.

[173] R. Vilalta and Y. Drissi, "A perspective view and survey of meta-learning," *Artificial intelligence review*, vol. 18, no. 2, pp. 77–95, 2002.

[174] M. Garnelo, J. Schwarz, D. Rosenbaum, F. Viola, D. J. Rezende, S. Eslami, and Y. W. Teh, "Neural processes," *arXiv preprint arXiv:1807.01622*, 2018.

[175] M. Hofer, A. Kormilitzin, P. Goldberg, and A. Nevado-Holgado, "Few-shot learning for named entity recognition in medical text," *arXiv preprint arXiv:1811.05468*, 2018.

[176] M. Rei and A. Søgaard, "Zero-shot sequence labeling: Transferring knowledge from sentences to tokens," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 2018, pp. 293–302.

[177] L. Logeswaran, M.-W. Chang, K. Lee, K. Toutanova, J. Devlin, and H. Lee, "Zero-shot entity linking by reading entity descriptions," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 3449–3460.

[178] L. Wu, F. Petroni, M. Josifoski, S. Riedel, and L. Zettlemoyer, "Scalable zero-shot entity linking with dense entity retrieval," *arXiv preprint arXiv:1911.03814*, 2019.

[179] B. Zhou, D. Khashabi, C.-T. Tsai, and D. Roth, "Zero-shot open entity typing as type-compatible grounding," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018, pp. 2065–2076.

[180] S. Kumar, S. Jat, K. Saxena, and P. Talukdar, "Zero-shot word sense disambiguation using sense definition embeddings," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 5670–5681.

[181] O. Levy, M. Seo, E. Choi, and L. Zettlemoyer, "Zero-shot relation extraction via reading comprehension," *arXiv preprint arXiv:1706.04115*, 2017.

[182] A. Gupta, P. Kumaraguru, C. Castillo, and P. Meier, "Tweetcred: Real-time credibility assessment of content on twitter," in *International Conference on Social Informatics*, Springer, 2014, pp. 228–243.

[183] C. Castillo, M. Mendoza, and B. Poblete, "Information credibility on twitter," in *Proceedings of the 20th international conference on World wide web*, ACM, 2011, pp. 675–684.

[184] Y.-J. Lu and C.-T. Li, "Gcan: Graph-aware co-attention networks for explainable fake news detection on social media," *arXiv preprint arXiv:2004.11648*, 2020.

[185] M. Gupta, P. Zhao, and J. Han, "Evaluating event credibility on twitter," in *Proceedings of the 2012 SIAM International Conference on Data Mining*, SIAM, 2012, pp. 153–164.

[186] D. ping Tian *et al.*, "A review on image feature extraction and representation techniques," *International Journal of Multimedia and Ubiquitous Engineering*, vol. 8, no. 4, pp. 385–396, 2013.

[187] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR)*, vol. 41, no. 3, p. 15, 2009.

[188] R. Chalapathy and S. Chawla, "Deep learning for anomaly detection: A survey," *arXiv preprint arXiv:1901.03407*, 2019.

[189] B. Zhang, Q. Zhao, W. Feng, and S. Lyu, "Alphamex: A smarter global pooling method for convolutional neural networks," *Neurocomputing*, vol. 321, pp. 36–48, 2018.

[190] A. Brown, A. Tuor, B. Hutchinson, and N. Nichols, "Recurrent neural network attention mechanisms for interpretable system log anomaly detection," in *Proceedings of the First Workshop on Machine Learning for Computing Systems*, 2018, pp. 1–8.

[191] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning, "A large annotated corpus for learning natural language inference," *arXiv preprint arXiv:1508.05326*, 2015.

[192] A. Williams, N. Nangia, and S. R. Bowman, "A broad-coverage challenge corpus for sentence understanding through inference," *arXiv preprint arXiv:1704.05426*, 2017.

[193] R. Ghaeini, S. A. Hasan, V. Datla, J. Liu, K. Lee, A. Qadir, Y. Ling, A. Prakash, X. Fern, and O. Farri, "Dr-bilstm: Dependent reading bidirectional lstm for natural language inference," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 2018, pp. 1460–1469.

[194] T. Bansal, R. Jha, and A. McCallum, "Learning to few-shot learn across diverse natural language classification tasks," *arXiv preprint arXiv:1911.03863*, 2019.

[195] D. Yogatama, C. d. M. d'Autume, J. Connor, T. Kocisky, M. Chrzanowski, L. Kong, A. Lazaridou, W. Ling, L. Yu, C. Dyer, *et al.*, "Learning and evaluating general linguistic intelligence," *arXiv preprint arXiv:1901.11373*, 2019.