Tackling Social Value Tasks with Multilingual NLP

A thesis submitted to the
College of Graduate and Postdoctoral Studies
in partial fulfillment of the requirements
for the degree of Master of Science
in the Department of Computer Science
University of Saskatchewan
Saskatoon

By
Md Rabiul Awal

Permission to Use

In presenting this thesis in partial fulfillment of the requirements for a Postgraduate degree from the University of Saskatchewan, I agree that the Libraries of this University may make it freely available for inspection. I further agree that permission for copying of this thesis in any manner, in whole or in part, for scholarly purposes may be granted by the professor or professors who supervised my thesis work or, in their absence, by the Head of the Department or the Dean of the College in which my thesis work was done. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of Saskatchewan in any scholarly use which may be made of any material in my thesis.

Disclaimer

Reference in this thesis to any specific commercial products, process, or service by trade name, trademark, manufacturer, or otherwise, does not constitute or imply its endorsement, recommendation, or favoring by the University of Saskatchewan. The views and opinions of the author expressed herein do not state or reflect those of the University of Saskatchewan, and shall not be used for advertising or product endorsement purposes.

Requests for permission to copy or to make other uses of materials in this thesis in whole or part should be addressed to:

Head of the Department of Computer Science
176 Thorvaldson Building, 110 Science Place
University of Saskatchewan
Saskatoon, Saskatchewan S7N 5C9 Canada

OR

Dean
College of Graduate and Postdoctoral Studies
University of Saskatchewan
116 Thorvaldson Building, 110 Science Place
Saskatoon, Saskatchewan S7N 5C9 Canada

Abstract

In recent years, deep learning applications have shown promise in tackling social value tasks such as hate speech and misinformation in social media. Neural networks provide an efficient automated solution that has replaced hand-engineered systems. Existing studies that have explored building resources, e.g. datasets, models, and NLP solutions, have yielded significant performance. However, most of these systems are limited to providing solutions in only English, neglecting the bulk of hateful and misinformation content that is generated in other languages, particularly so-called low-resource languages that have a low amount of labeled or unlabeled language data for training machine learning models (e.g. Turkish). This limitation is due to the lack of a large collection of labeled or unlabeled corpora or manually crafted linguistic resources sufficient for building NLP systems in these languages.

In this thesis, we set out to explore solutions for low-resource languages to mitigate the language gap in NLP systems for social value tasks. This thesis studies two tasks. First, we show that developing an automated classifier that captures hate speech and nuances in a low-resource language variety with limited data is extremely challenging. To tackle this, we propose HateMAML, a model-agnostic meta-learning-based framework that effectively performs hate speech detection in low-resource languages. The proposed method uses a self-supervision strategy to overcome the limitation of data scarcity and produces a better pre-trained model for fast adaptation to an unseen target language. Second, this thesis aims to address the research gaps in rumour detection by proposing a modification over the standard Transformer and building on a multilingual pre-trained language model to perform rumour detection in multiple languages. Specifically, our proposed model MUSCAT prioritizes the source claims in multilingual conversation threads with co-attention transformers. Both of these methods can be seen as the incorporation of efficient transfer learning methods to mitigate issues in model training with small data.

The findings yield accurate and efficient transfer learning models for low-resource languages. The results show that our proposed approaches outperform the state-of-the-art baselines in the cross-domain multilingual transfer setting. We also conduct ablation studies to analyze the characteristics of proposed solutions and provided empirical analysis outlining the challenges of data collection to performing detection tasks in multiple languages.

Acknowledgements

I want to first thank my mentor and supervisor, professor Roy Ka-Wei Lee, for taking me as a student and allowing me to research the field that I find fascinating. Even though Roy spent the entire time in Singapore while I was in Saskatchewan, and that at times was challenging, our time together is always enjoyable and lovely. Roy has supported me in countless ways throughout my research journey, and I am grateful for his mentorship. Following Roy's departure from the school, professor Michael Horsch was kind enough to take over as my supervisor. Mike went above and beyond to support me in my study, even after his retirement. Mike has helped me see things from new perspectives, and talking to him whenever I was unsure of something always made me feel nice and at ease.. I will miss our weekly conversations. Professor Ian Staveness selflessly volunteered to assist me in completing my thesis after Mike's unexpected retirement. I am grateful to Ian for his critical advice and guidance towards the final steps. I must be lucky to have such nice supervisors and I know it is rare. I am also grateful for the feedback and advice I received from my thesis committee throughout this process.

All of the research presented in this thesis is a result of multiple fruitful collaborations with inspiring, amazing individuals with whom I had the pleasure to interact and learn during my time as a master's student.

Amid the global lockdown, I began the program. During those trying times, my friends were a huge comfort. I must express my gratitude to my Saskatoon classmates who were a huge help to me with daily necessities. I am fortunate to meet new people here that I can cherish for years to come. I also want to thank my friends and lab mates in Singapore for their support. My friends and family back home in Dhaka have always been there for me when I needed them. A special thanks to Resat, who brings meaning and joy into my life.

I want to say out loud that I enjoyed my program at UofS. Throughout the course of my two years of study, I became genuinely interested in the field of deep learning.

Contents

Permission to Use . . . . . **i**

Abstract . . . . . **ii**

Acknowledgements . . . . . **iii**

Contents . . . . . **iv**

List of Tables . . . . . **vi**

List of Figures . . . . . **vii**

List of Abbreviations . . . . . **viii**

1 Introduction **2**
1.1 Social Value Tasks in NLP 3
1.2 Deep Learning for NLP 4
1.3 NLP for World Languages 5
1.4 Solving Value Tasks in Multiple Languages 5
1.5 Contributions 6
1.6 Outline 7

2 Background **8**
2.1 Neural Networks for NLP 8
2.1.1 Deep Feedforward Neural Networks 9
2.1.2 Recurrent Neural Nets 11
2.1.3 Long Short Term Memory Networks 12
2.1.4 Bidirectional RNNs 13
2.1.5 Seq2seq models 14
2.1.6 Attention Mechanism 14
2.1.7 Attention Variants 16
2.2 Transformers 17
2.2.1 Transformers Architecture 18
2.2.2 Transformer Usage, Analysis and Comparison to Other Network Types 20
2.3 Neural Language Models 21
2.3.1 Language modeling 21
2.3.2 Pretrained language models 22
2.3.3 BERT: Bidirectional Encoder Representations from Transformers 23
2.3.4 Multilingual Pre-trained Language Models 24
2.4 Transfer learning in NLP 25
2.4.1 Word embeddings and contexualized representations 25
2.4.2 Recipes for Successful Fine-tuning of a Pre-trained Language Model 25
2.4.3 Cross-lingual Transfer Learning 26
2.4.4 Transfer Learning Approaches on Limited Data 26
2.4.5 Multi-task Learning 28

3 Low-resource Hate Speech Detection **30**
3.1 Introduction 30
3.2 Related Work 32
3.3 Multilingual Hate Speech Detection 34

3.3.1 Model Agnostic Meta-Learning (MAML) . . . 35
3.3.2 Proposed Model: HateMAML . . . 35
3.3.3 HateMAML Algorithm . . . 37
3.3.4 Self-refinement and Zero-shot Learning . . . 37
3.4 Experiments . . . 38
3.4.1 Evaluation . . . 40
3.4.2 Baselines . . . 40
3.4.3 Variations of HateMAML . . . 41
3.4.4 Implementation Details . . . 41
3.5 Results and analysis . . . 41
3.5.1 Zero-shot Experiments . . . 41
3.5.2 Domain Adaptation Experiments . . . 43
3.5.3 Full Fine-tuning vs Meta-training . . . 44
3.6 Major takeaways . . . 45

4 Multilingual Rumour Detection . . . 46
4.1 Introduction . . . 46
4.2 Related Work . . . 47
4.2.1 Rumour Verification . . . 47
4.2.2 Multilingual Pre-Trained Language Models . . . 48
4.2.3 Handling Long Sequences and Hierarchical Transformers . . . 48
4.3 Task Formulation . . . 48
4.4 Methodology . . . 49
4.4.1 Input Representation . . . 50
4.4.2 Local Context Encoder . . . 50
4.4.3 Global Context Encoder . . . 52
4.5 Dataset Construction . . . 52
4.5.1 Translating Existing Rumour Datasets . . . 52
4.5.2 Collecting Multilingual Rumour Dataset . . . 53
4.6 Experiments . . . 54
4.6.1 Baselines . . . 54
4.6.2 Experimental Settings . . . 55
4.6.3 Evaluation Settings . . . 55
4.6.4 Implementation Details . . . 56
4.7 Results & Analysis . . . 56

5 Conclusion . . . 60
5.1 Summary . . . 60
5.2 Future work . . . 61
5.2.1 Constructing datasets for social value tasks in low-resource languages . . . 61
5.2.2 Exploring meta-learning for fine-grained and implicit hate speech detection . . . 61
5.2.3 Detecting rumour with fact-checking resources in multiple languages . . . 61

References . . . 63

List of Tables

2.1 Transformer comparison to other network types. 21

3.1 A summary of the multilingual hate speech datasets. 38
3.2 A summary of the multilingual hate speech datasets. 39
3.3 List of languages and their ISO codes used in our experiments. 39
3.4 Zero-shot evaluation on target low-resource languages 42
3.5 Fine-tuning and meta-training evaluation on multilingual hate speech datasets 42
3.6 Zero-shot experiments and results using auxiliary languages on SemEval2020 dataset. 43
3.7 Domain Adaptation experiments on both fine-tuning and meta-training. 44

4.1 Statistical summary of PHEME and Twitter16. 53
4.2 Statistical summary of SEAR dataset. 54
4.3 Monolingual results on Twitter16 and PHEME 57
4.4 Monolingual results on SEAR. 57
4.5 Multilingual results on Twitter16 and PHEME. 58
4.6 Multilingual results on SEAR. 59

List of Figures

2.1 Recurrent neural network model, viewed as a "unrolled" computation graph. 11
2.2 Long short term memory network. 13
2.3 Sequence-to-sequence model [1]. 14
2.4 Attention mechanism [2]. 15
2.5 The Transformers architecture [3]. 18
2.6 The Multi-head self-attention [3]. 19

3.1 Proposed HateMAML model sketch. 36

4.1 Examples of two rumour conversations. 47
4.2 An overview of the MUSCAT model. 50
4.3 Source Co-Attention Module. 51
4.4 Mono-lingual performance on (a) Twitter16, (b) PHEME and (c) SEAR. 58
4.5 Multi-lingual performance on (a) Twitter16, (b) PHEME and (c) SEAR 59
4.6 MUSCAT's cross-lingual performance on (a) Twitter16 and (b) SEAR. 59

List of Abbreviations

AI	Artificial Intelligence
BART	Bidirectional and Auto-Regressive Transformers
BERT	Bidirectional Encoder Representations from Transformers
BiGCN	Bidirectional Graph Convolutional Neural Networks
BPTT	Back-Propagation Through Time
BRNN	Bidirectional RNN
CHT	Coupled Hierarchial Transformer
DNN	Deep Neural Networks
ELMo	Embeddings from Language Model
FSL	Few-Shot Learning
FFN	Fully-connected Feedforward Network
GCE	Global Context Encoder
GLUE	General Language Understanding Evaluation
GPU	Graphics Processing Unit
LM	Language Model
LCE	Local Context Encoder
LSTM	Long Short-Term Memory
HateMAML	Hate-speech Model-Agnostic Meta Learning
GPT	Generative Pre-trained Transformer
ML	Machine Learning
MAML	Model-Agnostic Meta Learning
MHSA	Multi-Head Self-Attention
MLM	Masked Language Modeling
MLP	Multi-Layer Perceptrons
MPLM	Multilingual Pre-trained Model
MTL	Multi-Task Learning
MT-DNN	Multi-Task Deep Neural Networks
MUSCAT	Multilingual Source Co-Attention Transformer
mBERT	multilingual Bidirectional Encoder Representations from Transformers
NLP	Natural Language Processing
NLI	Natural Language inference
NLU	Natural Language Understanding
NMT	Neural Machine Translating

NSP	Next Sentence Prediction
OSP	Online Social Platform
POS	Parts Of Speech
PLM	Pre-trained Language Model
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Networks
RoBERTa	Robustly optimized BERT approach
RvNN	Recursive Neural Networks
SAN	Stacked Attention Network
SEA	South East Asian
SGD	Stochastic Gradient Descent
SVM	Support Vector Machines
TiDyQA	Question Answering in Typologically Diverse Languages
T5	Text-To-Text Transfer Transformer
ULMFiT	Universal Language Model Fine-tuning for Text Classification
VQA	Visual Question Answering
WSD	Word Sense disambiguation
XLM	Cross-lingual Language Model
XLM-R	XLM-RoBERTa

Statement of Co-Authorship

The chapters in this thesis describe work that has been published in the following conferences or under review:

- Md Rabiul Awal, Minh Dang Nguyen, Roy Ka-Wei Lee and Kenny Tsu Wei Choo. MUSCAT: Multilingual Rumor Verification in Social Media Conversations. 2021 IEEE International Conference on Big Data (Big Data), 2022.

- Md Rabiul Awal, Eshaan Tanwar, Roy Ka-Wei Lee and Tanmay Chakrabarty. Model-Agnostic Meta-Learning for Multilingual Hate Speech Detection. IEEE Transactions on Computational Social Systems, 2022. **under review**

I hereby declare that this thesis incorporates material that is result of joint research, as follows:

- Chapters 3 of the thesis include the outcome of publication which have the following other co-authors: Minh Dang Nguyen, Roy Ka-Wei Lee and Kenny Tsu Wei Choo. In all cases only my primary contributions towards these publications are included in this thesis, and the contribution of co-authors Minh Dang Nguyen, Roy Ka-Wei Lee and Kenny Tsu Wei Choo was primarily through – Minh Dang Nguye provided assistance in data collection and experimentation; Roy Ka-Wei Lee and Kenny Tsu Wei contributed feedback on refinement of ideas and editing of the manuscript.

- Chapter 4 incorporates unpublished material co-authored with Eshaan Tanwar under the supervision of professor Roy Ka-Wei Lee and Tanmay Chakrabarty. In all cases the key ideas, primary contributions, experimental designs, data analysis, interpretation, and writing were performed by myself; Eshaan Tanwar contributed to the processing datasets, running baseline experiments and graphing results; Roy Ka-Wei Lee and Tanmoy Chakrabarty contributed feedback on refinement of ideas and editing of the manuscript.

1 Introduction

Social media websites like Twitter and Reddit, where people communicate online, are a part of our daily lives. The opportunity to get connected from anywhere anytime and freely has changed our ways of communication and is now deeply rooted in our personal and social lives. For example, Facebook is the most popular social media platform, with roughly 2.93 billion monthly active users as of the first quarter of 2022.[1] With such integrated and widespread use of online communications, the significance and role of social media in society have become unprecedented. A 2016 study shows that a majority of U.S. adults (62%) turn to social media for news, with 18% doing so often [4]. A recent headline in BBC noted that TikTok (another popular social media platform) had become a major political battleground for Colombia's presidential election candidates.[2] Not only recent events, but the role of fake news in social media in the 2016 US election has also led to the rethinking of social media influence in political events [5].

Online social platforms (OSPs) allow users to post freely, express whatever they want, and share with anyone even anonymously. The misuse of these platforms are also typical, i.e. offensive attacks on other users or proliferation of false news. There are reports that social media now plays a larger role in hate-related attacks, radicalization, and coordinated spread of misinformation.[3] Fake news in social media can have potential realworld consequences on a large scale since fake news is more widespread in those than mainstream news media.[4] Vast online interactions without proper safeguard can challenge meaningful and healthy communication on both personal and social levels, leading to more critical societal problems. Two of the major challenges in OSPs are: tackling hate speech and preventing misinformation. Offensive language troubles sound online communication, hate speech and stereotypes attack and disparage individual or minority groups, and misinformation could create discord in society and lead to violence. Due to the societal concern and how widespread the problems are on the Internet, there is a need for automatic detection[6, 7].

Most online interaction between users is in the form of natural language such as speech, text, etc. Natural language data are now widely processed by Natural Language Processing (NLP) systems. An NLP system like content filtering at Facebook responds to rapid live interactions in a fraction of a second and at a billion-user scale. It can provide an efficient and intelligent solution to problems such as hate speech at scale. Recent claims that NLP has revolutionized the field of Artificial Intelligence(AI) by leveraging the abundance of web data and modern computing (GPUs) [8]. For instance, Facebook is now using super efficient AI models to

[1] https://www.statista.com/statistics/264810/number-of-monthly-active-facebook-users-worldwide/
[2] https://www.bbc.com/news/av/world-latin-america-61634206
[3] https://www.cnn.com/2021/12/07/tech/facebook-myanmar-rohingya-muslims-intl-hnk/index.html
[4] https://www.buzzfeednews.com/article/craigsilverman/viral-fake-election-news-outperformed-real-news-on-facebook

filter 94% of online content before anyone reports it, whereas early systems were heavily reliant on reports from users.[5] AI and big data-driven approaches is an active area of research for understanding harms in OSPs and developing systems for tackling online misuse.[6] When it comes to the accessibility of NLP systems for global languages, there is, nevertheless, a significant gap. Current systems are brittle and a lot of progress is yet to be done to make them safe and trustworthy, but for some multilingual communities, the algorithms can reportedly contribute to critical harms such as "human rights violations".[7] The study of NLP for societal benefits comes within the broad category of "AI for social good," an emerging area that examines the aforementioned issues, as well as justice and ethical issues in AI [9].

1.1 Social Value Tasks in NLP

Machine learning is an approach that teaches computer to learn from experience. The experience in our case is an instance of text data (e.g. Tweets). We explored text-only NLP systems that are powered by deep learning, a modern approach to machine learning (ML). In this chapter and subsequent discussions, we will consider machine learning tasks that are closely related to online abuse, misinformation, and harm as social value tasks. There are several social value tasks involving text data, for example, tackling hate speech in short texts and early detection of rumour in online conversations. They can be categorized as text classification tasks in NLP, where a machine learning model predicts the class label from a set of discrete labels.

Hate speech or offensive language is intended to offend or attack a person or group based on race, identity, ethnicity, gender, religion, etc [10]. The machine learning challenge is detecting offense and hatred of diverse forms, ones with lexical cues, e.g. slurs, or the others that are indirect forms such as stereotypes. There are many existing works on detecting hate speech in social media (e.g. Twitter) [10, 11], commonly studied in text classification of short texts. One of the key challenges in hate speech detection is the adaptation of machine learning models to diverse domains and forms of hate speech with limited available data.

Detecting misinformation in social media is another challenging task. Social media platforms may contain both misinformation and disinformation. False or erroneous information is referred to as misinformation. Rumours, insults, and practical jokes are a few examples. Disinformation is malicious content that is purposefully spread, and it includes hoaxes, spear phishing, and propaganda. Both of them could spread fear and suspicion among the population. We focus on misinformation detection, specifically detecting rumours in social media. The goal of automated rumour detection is to detect rumours accurately to curb their spread and debunk rumours early before they reach a wider audience. For example, the task of rumour detection on Twitter detects the veracity of an input conversation. Given a conversation thread containing a source post and corresponding replies is input to a classifier that predicts the veracity labels. A conversation on Twitter typically has a long context length and contains multi-level user interactions which can form a tree

[5] https://ai.facebook.com/blog/how-facebook-uses-super-efficient-ai-models-to-detect-hate-speech/
[6] https://ec.europa.eu/research-and-innovation/en/horizon-magazine/can-artificial-intelligence-help-end-fake-news
[7] https://time.com/6217730/myanmar-meta-rohingya-facebook/

structure [12, 13]. The key aspect to this problem is processing long-range interactions of users and making predictions accordingly [14]. This is a relevantly new research interest and the problem is studied mostly by investigating breaking news events in US domains.

1.2 Deep Learning for NLP

Deep learning has become ubiquitous in NLP for many tasks including text classification, machine translation, question answering, and text summarization. Deep learning models leverage rich representations of neural networks to produce high-quality features and can learn a task well, provided sufficient training data is available [15]. Recent work in deep neural networks has demonstrated substantial gains on many NLP tasks. The core to this substantial progress in NLP comes from a technique called language modeling [16, 17]. Powerful pre-trained language models such as Bidirectional Encoder Representations from Transformers (BERT), Robustly optimized BERT approach (RoBERTa) have already reached near-human performance on General Language Understanding Evaluation (SuperGLUE) leaderboard[18], a suite of 11 benchmark NLP tasks. Large-scale pre-training on internet scale data can lead to "super-human" performance that was thought impossible a few years ago. One notable example is Generative Pre-trained Transformer (GPT3) [19], a language model with 175 billion parameters that can solve many language tasks with very little or no task-specific supervision. It is important to keep in mind that a significant portion of these language models are created using internet-scale data and computing resources that are far larger than the typical university budget. Large-scale modeling as it is now requires a lot of resources, and many of them are impossible to fit for regular usage with standard compute such as personal computers and GPUs.

Where early research in text classification mostly focused on identifying useful features in a corpus [20] (i.e. lexical cues for hate speech detection), the focus is now shifted to the use of learned neural representations as automated feature vectors. A de-facto standard choice of neural networks for NLP tasks is pre-trained language models. The current paradigm in many sentence classification tasks involves creating downstream classifiers on top of the pre-trained knowledge, perhaps most famously BERT [21] fine-tuning on the task-specific training data.

Research in hate speech and misinformation are relatively recent interests among researchers. Existing work investigates hate speech with a focus on specific types of hatred or target groups, identifying bias in hate speech models, and a recent interest in multilingual methods [10, 11, 22]. State-of-the-art methods are based on deep neural networks; most of them leverage pre-trained language model's fine-tuning [23]. However, they are limited to a few high-resource languages and domains, which limits their wider use under limited resources. Moreover, hate speech can be expressed in several nuanced forms, such as stereotypes which neural nets struggle to understand [24]. Detecting hate speech of diverse and changing forms poses a major challenge for automated hate speech classifiers. Work in rumour detection is rather limited to a smaller domain, for example, breaking news events in a particular region [25]. One bottleneck is the lack of available datasets

due to the nature of the task. Dataset collection for rumour detection heavily relies on fact-checking websites and information on the real-time proliferation of rumours on social media. For both the task of hate speech and rumour detection existing work is limited to a few high-resource languages, warranting a recent interest in the democratization of resources i.e. datasets and models to many world languages.

1.3 NLP for World Languages

Multilingual NLP systems can comprehend a variety of languages. The economy, society, and international information exchange can all benefit from systems with multiple natural language processing (NLP) capabilities. The development of NLP for world languages faces two main challenges. i) Linguistic diversity: there are more than 6500 spoken or signed languages in the world, which belong to over 14 different language families. ii) Uneven distribution of linguistic resources: labeled and unlabeled resources vary across languages for training machine learning models. Based on their available resources, we can divide the world's languages into two categories. Languages having large collections of labeled or unlabeled corpora or manually crafted linguistic resources sufficient for building statistical NLP solutions referred to as high-resource languages. English, for instance, or Chinese. Low-resource languages lack these resources and are underrepresented both in academia and industry. Languages with limited resources include Turkish and Swahili. In this thesis, we aim to develop deep learning systems detect hate speech and rumours for underrepresented languages.

1.4 Solving Value Tasks in Multiple Languages

There is much community interest in NLP was given to high-resource languages (e.g. English). Decades of work led to the creation of machine learning tasks, text corpus, training data, evaluation benchmarks, software packages, and other resources for a few languages only. The diversity of language systems is little, thus discriminating against many languages and leaving them under-explored and resource-poor [26]. The advances in deep neural language modeling should also be made available to the people of the low-resource community. There is untapped potential and more to be done to make deep learning systems available in low-resource languages (e.g. Hindi). Recent deep multilingual models take a viable step towards multilingual development of NLP systems. Recently researchers have released datasets and methods for tackling hate speech in many low-resource languages [23, 27, 28, 29]. Leveraging pre-trained knowledge and supervised datasets can help improve the hate and rumour detection tasks. However, an important dimension of the real-world challenge remains to make these resources accessible to the overlooked multilingual community, for example, the global south. The studies in this thesis explored developing NLP systems for social value tasks with a focus on the low-resource community. We mainly investigated hate speech in short texts and misinformation in conversation threads. The remainder of this chapter summarizes our contributions.

Research objectives. This thesis is about developing a deep learning model that can detect either hate speech or rumours in multiple languages using limited training data. Our goal is to support underrepresented

languages in tackling social value tasks. Our primary research interests are hate speech and rumour detection, two challenging OSP problems. We investigated multilingual hate speech detection while faced with diverse domains and forms of hate speech due to language variety and model development in small or no training data. We investigate an alternative approach to limited data fine-tuning to improve transfer learning in multilingual modeling. We propose a meta-training-based low-resource adaptation framework and show that meta-learning has better cross-lingual transfer performance over standard fine-tuning. Our research indicates that meta-training, as opposed to limited data fine-tuning, could be a more effective approach for multilingual hate speech. Research in multilingual rumour detection is very limited, with machine learning models and datasets only available in English and Chinese. Our goal is to build the first multilingual rumour detection system for South East Asian languages. We start by building a dataset for four SEA languages to support the development of multilingual NLP systems for rumour detection. Towards multilingual modeling, we made efforts to adopt English-only baselines in multilingual training and provided a starter evaluation tool. Existing systems are inefficient in processing a long conversation thread. We propose a new multilingual model that deals with prioritizing the source claim while processing long-range replies, which helps to jointly focus on salient parts of both the source claim and replies for rumour verification. Future work in this direction can benefit from our contribution to datasets, modeling, and evaluation.

1.5 Contributions

The contributions of this thesis are summarized as follows:

- We were among the first to investigate efficient knowledge transfer in limited-data multilingual hate speech detection. In particular, we propose HateMAML, a domain-adaptive model-agnostic meta learning (MAML) algorithm for hate speech detection in low-resource languages. One of the novelties of HateMAML lies in the efficient utilization of available training samples in the source and auxiliary languages.

- We made effort to understand better low-resource hate speech detection in data scarce and domain adaptation scenarios. We investigate efficacy of transfer in multiple languages in varying configurations and implementation of pre-trained models for hate speech detection. We concluded that meta-training over all available language training data is promising and meta-learning could be an alternative to standard fine-tuning that yields superior performance.

- Another focus in this thesis was tackling misinformation in social media i.e. detecting rumours in Twitter conversations in low-resource languages. We set out to develop rumour detection resources for low-resource languages, for example, South East Asian (SEA) languages. In particular, we construct first multilingual rumour detection corpus in two languages from both real-world and synthetic data. We believe that this dataset is a first but important step to building rumour verification systems in

low-resource languages.

- Finally, we implemented several existing approaches to rumour detection in English, making efforts to adapt them to detect rumours in multiple languages using a single model. We observed that parts of a conversation thread need to be attended well along with the source post in a thread to verify a rumour event. To tackle this, we propose multilingual source co-attention transformer (MUSCAT), a transformer-based model that uses co-attention Transformer to focus on both the source post and replies in order to verify a rumour event.

1.6 Outline

This chapter provides an introduction to the research problems explored. In Chapter 2, I will present related backgrounds on neural networks and transfer learning for multilingual NLP. Chapter 3 will outline effective transfer learning for low-resource hate speech detection. Chapter 4 will detail our investigation on multilingual rumour detection with a focus on South East Asian languages. Chapter 5 will provide a conclusion on the findings and some notes on future work.

2 Background

Neural networks are powerful tools for modeling complex data, and have been successfully applied to a variety of tasks in NLP [16, 30]. In this chapter, we will start by giving an overview of neural networks. Following that, we will discuss how neural networks can be used to model sequences of data, such as in recurrent and long short-term memory networks [31, 32]. The development of general-purpose language representations using Transformers, a self-attention based model [3], has revolutionised recent NLP advancements [21, 33]. Transfer learning is an effective technique for efficient learning by using information from a general-purpose pre-trained model. However, knowledge transfer learning can be challenging in low-resource environments. We also discuss current strategies that help overcome these challenges.

2.1 Neural Networks for NLP

NLP tasks have been traditionally solved using machine learning classifiers such as Support Vector Machines (SVM) and logistic regression [34]. Given a task of interest, task-specific features are extracted manually from expert knowledge about language that seems to be useful for the given task. Those features are used for training a machine learning classifier using a suitable learning algorithm. Take for example a classification task $p(y|x)$ where goal is to classify input x into a target y. The traditional statistical approaches in NLP often design hand-crafted features to represent x and then apply a linear statistical model to learn the classification function. These traditional systems have some major challenges: not only is hand-crafted feature engineering labor intensive and requires expert domain knowledge, but the systems making use of them have not been able to perform well on most challenging language tasks [35].

Deep learning models learn high-dimensional latent feature vector representations via a deep neural network [36] and then learn the classification function on top of the latent features. Recently, the re-emergence of deep neural networks (DNN) provides a compelling but simple way to learn hierarchical representations to handle complex tasks. In pioneering work in NLP, Collobert *et al.* [35] showed deep neural networks can learn various natural language processing tasks almost from scratch including part-of-speech tagging, chunking, named entity recognition, and semantic role labeling. Researchers have shown that neural methods outperform traditional statistical learning techniques [37]. The learning algorithm for neural networks avoids hand-crafted feature engineering leading to the versatility of learnable tasks and therefore disregards the need for a lot of prior knowledge. Specialized networks such as recurrent neural networks are commonly used for sequence modeling tasks. Recurrent networks and their variants such as long-short term memory

networks are powerful learners but limited for successful training of modern deep learning models [38] due to lack of parralelization in the network. Recent alternative architectures such as attention mechanisms are proposed to power state-of-the-art NLP systems. Most importantly successful learning in deep networks is heavily dependent on supervised training using a large dataset.

Note that the latent representations for each new NLP task need to be learned from scratch. In many cases, this becomes problematic since, for many tasks, supervised training data is limited, and the quality of the latent feature representation deteriorates. A common approach to mitigate this problem is to find a generic task and learn transferable feature representations that can be shared across all NLP tasks. One way to build such a model is language modeling, where the task is predicting the next word given previous words is one such task that takes advantage of semi-supervised learning and an abundance of web corpus for training. In fact, the latest paradigm shift in NLP is based on this simple idea of pre-trained language models where pre-trained features can be leveraged to learn any new NLP task with limited task-specific training data. Today's state-of-the-art NLP systems are mostly based on pre-trained models. For instance, pre-trained BERT [21] and GPT [19] have surpassed human-reported accuracy on challenging natural language understanding (NLU) benchmarks.

A deep learning model is generally a neural network with many stacked layers. In this section, we will briefly introduce several neural network models that are commonly used in NLP and that are essential to the work done in this thesis. Recent advancements in sequence modeling, for example, self-attention based Transformers will be discussed in detail which provide a basis for pre-trained language models and their transfer learning to be discussed in subsequent sections.

2.1.1 Deep Feedforward Neural Networks

A neural network is made of artificial neurons. A neuron is a computational unit which has much of the same mathematics as logistic regression. A single layer neural network has multiple neurons. A Deep Feedforward Neural Network (DNN) consists of multiple layers. Core to a neural network is the additional layers in the middle, called hidden layers. The stack of hidden layers maps the relationship between the input and output, where each hidden layer receives input from its previous layer. The multilayer feedforward networks are also called multilayer perceptrons (MLP).

A Deep Feedforward Network defines a mapping $y = f(x;\theta)$. During training, the parameters θ are learned so that y results in the best approximation of the original function $f^*(x)$. Information flows from the input x through some intermediate steps, all the way to the output y. There are no Feedback connections. Any feedforward neural network with L layers, can be seen as a composition of functions f^l corresponding to each layer l:

$$f(x) = f^L(f^{L-1}(...f^1(x)...)). \tag{2.1}$$

The parameters of a single hidden unit are a bias and a weight vector. We represent parameters in a layer

as a single matrix W and bias vector b by combining weights vectors and biases of each units in the layer. By this choice, we can make the required computation efficient in a single layer with matrix operations. The output of the hidden layer h is computed as follows:

$$h = \sigma(Wx + b). \tag{2.2}$$

Each hidden layer applies a linear transformation to the input data, followed by a non-linearity (e.g. sigmoid) function σ. For units in the hidden layer, this translates to each hidden unit taking a weighted sum of its input data from the previous layer and then applying an activation function. The non-linearity is critical to neural networks' representation learning [39].

Deep neural networks are the foundation of deep learning models that have achieved state-of-the-art results in many supervised and unsupervised NLP tasks. They are powerful learners compared to logistic regression classifiers. The universal approximation theorem states that a feedforward neural network can be shown to represent any function with a single hidden layer and a finite number of neurons [40].

Training Neural Networks

Deep neural networks are trained using two steps: forward propagation and backward propagation. Forward propagation takes an input x and produces an output $\hat{y}$; information flows forward through the hidden layers. During training, forward propagation terminates in the final output layer and produces a scalar loss value by comparing labeled target y and predicted $\hat{y}$. The backpropagation algorithm [41], also known as *backprop*, propagates the loss through the network to the input layer in order to compute the gradient. While backprop is responsible for computing the gradient, another learning algorithm is used to update the weights of neurons to the margin of error, most famously the stochastic gradient descent (SGD) algorithm.

The weights of neurons are randomly initialized at first [42]. The training process continues until a stopping criterion is met. This essentially performs learning in neural networks by finding a set of weights in neurons that worked best.

Deep learning algorithms take advantage of optimization in many contexts. The goal of machine learning algorithm is to reduce the expected generalization error, which means how well a model performs on unseen data. This quantity is known as the risk [43]. Towards this goal, a machine learning problem can be cast as an optimization problem - minimize the expected loss over a training set. In practice, neural networks are trained using many optimizers including Adam, RMsProp, Adadelta, etc. Variants of Adam optimizer[44] are commonly used in well-known NLP tasks [18]. Another important training technique is regularization, which mitigates overfitting in neural networks and allows to generalizae well to unseen data even when training on a finite training set.

2.1.2 Recurrent Neural Nets

Most natural language has a sequential form i.e. words in a sentence form a sequence, and audio has a temporal sequence. Recurrent neural networks are designed to process natural language data in a sequential manner, for example, by reading one word after another in a sentence. A recurrent neural network (RNN) is a feedforward neural network, that unfolds a recurrent computation unit across time t [45]. The input to an RNN is a sequence denoted as $(x_1, x_2, ..., x_t)$ where each data point is a real-valued vector. Similarly, the target or output sequence is $(y_1, y_2, .., y_t)$. An RNN can process a sequence of arbitrary length by recursively applying a transition function to its *internal hidden state vector* h_t of the input sequence.

$$h_t = \begin{cases} 0 & if\ t = 0 \\ f(h_{t-1}, x_t) & otherwise \end{cases} \tag{2.3}$$

The state-to-state transition function f is commonly composed of an element-wise nonlinearity with an affine transformation of both x_t and h_{t-1}. The output hidden state activation of the previous input sends feedback to its next state, thus creating a feedback loop in the chain of input sequences.

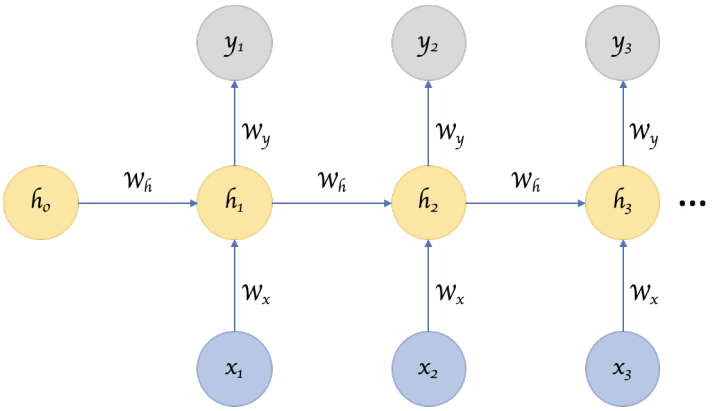


Figure 2.1: Recurrent neural network model, viewed as a "unrolled" computation graph.

At each time step t, an RNN takes two inputs, the input is x_t and the hidden previous state h_{t-1}. The forward propagation begins with the initial state h_0. Then, for each time step from $t = 1$ to $t = N$, we update following equations:

$$\begin{aligned} h_t &= \tanh(W^{hx}x_t + W^{hh}h_{t-1} + b_h), and \\ \hat{y_t} &= softmax(W^{yh}h_t + b_y), \end{aligned} \tag{2.4}$$

where the parameters are the bias vectors b_h and b_y and the weight matrices W^{hx}, W^{hh} and W^{yh}, respectively, for input-to-hidden, hidden-to-hidden and hidden-to-output.

Figure 2.1 is an example of a recurrent network that maps an input sequence to an output sequence of the same length. The last hidden state stores the information of the complete sequence. Traditionally, a simple

strategy for modeling a sequence is to map the input sequence to a fixed-sized vector, and then feed the vector to a softmax layer for classification or other tasks [31]. The total loss is computed by summing over all the time steps for a given sequence x paired with y. The backpropagation algorithm applied to RNNs is called backpropagation through time (BPTT). It iterates gradients backward in time to back-propagate gradients through time, from $t = N - 1$ down to 1 [46]. Similar to other neural networks, RNNs use their internal representation to perform a high-dimensional interpolation between training examples [47]. The result is that RNNs can successfully be applied to many complex sequential tasks such as machine translation, text classification, name entity recognition, etc.

However, RNNs are unable to store long-range dependency information and are prone to instability during training [48]. In practice, vanilla RNNs struggle to learn dependencies between more distant input tokens. As long-term dependency arises, they suffer from a well-known problem called vanishing gradients problem [32, 49] – the gradients shrink during the backpropagation for a long sequence length. As the gradients become smaller, the learning becomes difficult since the long-range structure is being ignored. However, for many NLP tasks, a long-range dependency is desired to solve the task coreference resolution. This limitation is addressed by altering architecture that prevents vanishing gradients during training.

2.1.3 Long Short Term Memory Networks

The long short-term memory (LSTM) network is a variation of the recurrent neural network, designed to overcome the problems of traditional RNNs, such as the vanishing gradients problem. The LSTM network has the same structure as the RNN, only the repeating module in the hidden layer is replaced by a *memory cell*, which can store information for long periods. The memory cell is a recurrently-connected unit that has an input, output, and forget gate. This cell is updated at each time step. The forget gate decides what information to keep or forget from the previous time step. The input gate controls what new information to add to the cell. The output gate decides what information from the memory cell will be output and flows out into the rest of the network.

The LSTM memory cell has many variations. Figure 2.2 is an example of the most commonly adopted version. It has four components, consisting of three gates and one cell state. Each gate updates the information in the cell state by the following composite function:

$$\begin{aligned} i_t &= \sigma(x_t W^{xi} + h_{t-1} W^{hi}), \\ f_t &= \sigma(x_t W^{xf} + h_{t-1} W^{hf}), \\ o_t &= \sigma(x_t W^{xo} + h_{t-1} W^{ho}), \\ \tilde{C}_t &= \tanh(x_t W^{xC} + h_{t-1} W^{hC}), \\ C_t &= f_t \otimes C_{t-1} + i_t \otimes \tilde{C}_t, and \\ h_t &= o_t \otimes \tanh(C_t), \end{aligned} \tag{2.5}$$

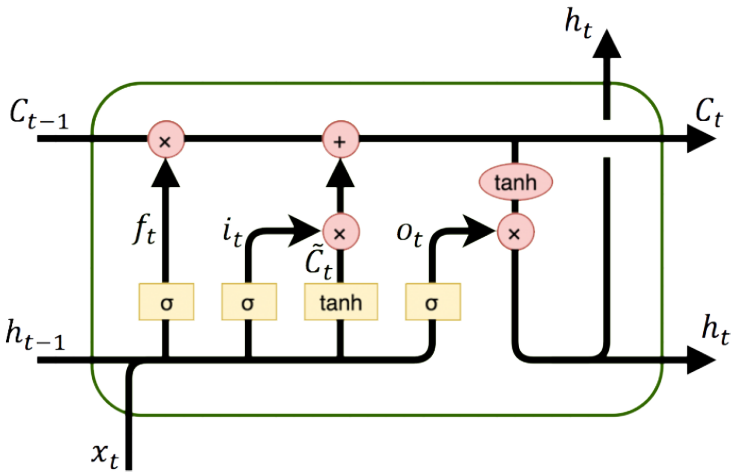


Figure 2.2: Long short term memory network.

where x_t is the input at the current time step, σ denotes the logistic sigmoid function, $\otimes$ is elementwise multiplication, i, f, o, and C are the input gate, forget gate, output gate, and cell state. The W's are the weight matrices for input, forget, output gates and memory cell, all of which are the same size as the hidden vector h.

LSTMs have been shown to be very successful at many tasks, such as machine translation [50], speech recognition [51], name entity recognition [52], and text generation [53]. While LSTMs are very successful at many tasks, they are not perfect. LSTMs are often slower and more resource intensive than standard recurrent neural networks [54]. This is due to the fact that LSTMs have more parameters than RNNs and the training process for LSTMs is more computationally intensive. Additionally, LSTMs are not always able to accurately learn long-range dependencies [49]. This is because the gates in LSTMs can occasionally block information required for the network to learn the dependency.

2.1.4 Bidirectional RNNs

Bidirectional RNNs (BRNNs) are a type of recurrent neural network that processes information from both the left and right sides. BRNNs can be seen as two stacked RNNs, one going forward and the other going backward. Such bidirectional information processing gives context to a computational unit from both the left and right sides. More formally, at time step t, a BRNN unit receives context from both $t-1$ and $t+1$. Bidirectionality is also being applied in long short-term memory (LSTM) networks [32].

BRNNs were first proposed by Schuster and Paliwal [55]. Since then, BRNNs have been shown to outperform traditional RNNs on a variety of tasks, such as speech recognition [56], machine translation [57] and sequence tagging [58]. BRNNs have also been applied to other areas such as time series analysis [59], and computer vision.

2.1.5 Seq2seq models

In many NLP applications, RNNs are used in sequence-to-sequence (seq2seq) modeling where input and output are both comprised of sequences [1, 31]. For instance, the neural machine translation task takes an input sequence (e.g. English) and outputs a target translation sequence (e.g. French). The seq2seq model follows an encoder-decoder structure, where an encoder reads and encodes an input sequence x, and the decoder outputs the target sequence y. Both the encoder and decoder are RNNs. Each cell block can be an LSTM or GRU unit. We show an example of seq2seq model in Figure 2.3.

Since the last state of the RNNs encoder network summarizes all the past inputs, it acts as the context vector for the decoder in the early implementation of the seq2seq model. The decoder is initialized with the fixed context vector and it generates output translation conditioning on all the previously predicted words. For instance, one RNN encodes a variable-length source sentence into a fixed length vector and the second RNN decodes variable length target sentence. At each step, the decoder is autoregressive meaning the model emits output one step at a time [47].

In a machine translation task consisting of an encoder-decoder for a language pair, the training objective is jointly maximizing the probability of correct translation given a source sentence, i.e. $argmax_y p(y|x)$. The whole encoder-decoder system is trained in an end-to-end manner using backpropagation.

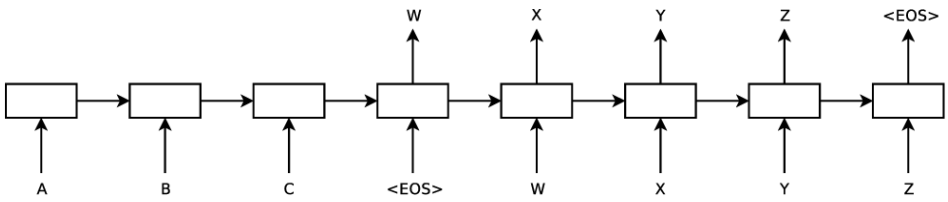


Figure 2.3: Sequence-to-sequence model [1].

The basic encoder-decoder approach works well for short sequences. Given a long input sequence, this becomes problematic. The encoder network information is squashed/compressed into a fixed context vector which the encoder transmits to the decoder as the contextual summary of the input. A potential issue is that the decoder network tends to forget the early information found in the encoder. This degrades the neural network training, especially for longer input sequences [50]. To address the problem of fixed context transmission, the *attention mechanism* was proposed [2, 60].

2.1.6 Attention Mechanism

The encoder-decoder framework with an attention mechanism separates the encoder from the decoder by a fixed context [2]. This fixed context acts as a bottleneck since it represents absolutely everything about the meaning of the source text. The attention mechanism is a solution to the bottleneck problem, a way of attention mechanism allowing the decoder to get information from all the hidden states of the encoder, not just the last hidden state. This allows for a much richer context for the decoder to consider.

The content-based attention mechanism is built on top of recurrent networks. It incorporates a distinct context vector for each decoder step. An alignment model lets the decoder put importance on parts of a source sentence by assigning attention weights to encoder hidden states. Intuitively, the attention mechanism allows the decoder to refer back to and attend to the relevant parts of an input sequence for target prediction. An example of content-based attention mechanism is shown in Figure 2.4.

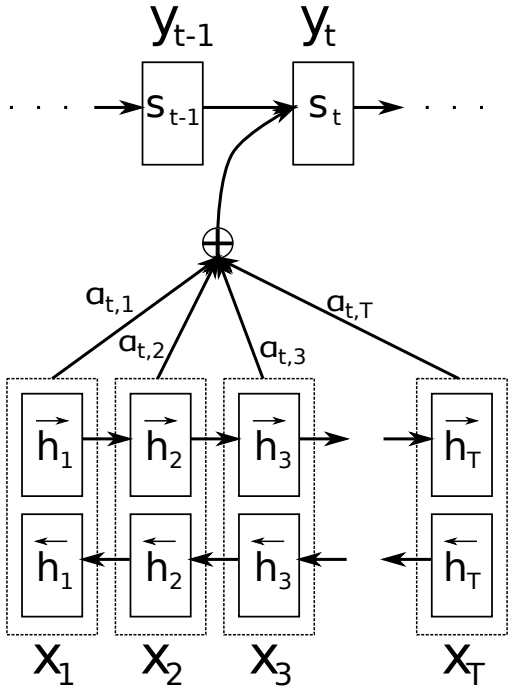


Figure 2.4: Attention mechanism [2].

In an RNN encoder-decoder network, we can decompose the target output probability as follows:

$$p(y) = \prod_{t=1}^{T} p(y_t|\{y_1, ..., y_{t-1}\}, c), \tag{2.6}$$

where c is the fixed context vector and target $y = y_1, ..., y_T$. We can parameterize the probability of decoding each word y_t:

$$p(y_t|y_{<t}, c) = g(y_{t-1}, s_t, c), \tag{2.7}$$

where g is a non-linear, possibly multi-layered transformation function that outputs the probability of y_t and s_t is the RNN hidden state.

In the content-based attention architecture, s_i the hidden state of time step i is computed by

$$s_i = f(s_{i-1}, y_{i-1}, c_i), \tag{2.8}$$

where f can either be a vanilla RNN, or LSTM unit or GRU unit. It is worth noting that, unlike the vanilla encoder-decoder architecture, here c_i is distinct and dynamically computed (see Equation 2.9) for each conditional probability at time step i.

Let's assume encoder hidden states $h_1, ..., h_T$ map the source sentence. A hybrid RNN is used to store high-dimensional representations h_i for i-th word along with surrounding information, both left and right side as shown in Figure 2.3. We need access to $h_1, ..., h_T$ to compute the context vector c_i.

The key aspect of the context vector here is that it can focus on parts of the input by assigning weights to the words seen in the encoder layer. Then, context vector c_i is computed as a weighted sum of precomputed h_is:

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j \tag{2.9}$$

where each weight α_{ij} is a probability and computed using an alignment model.

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})} \tag{2.10}$$

where

$$e_{ij} = a(s_{i-1}, h_j) \tag{2.11}$$

The alignment model e_{ij} is also called scoring function or energy function which scores the match between the output i and inputs around position j. The alignment model is a key component and is parameterized using feedforward neural networks. The encoder h_i's are precomputed, and all the components of the encoder-decoder including the alignment model are jointly trained using back-propagation.

Intuitively, this allows the decoder to have an attention mechanism while emitting output at each step. The decoder decides to focus on the parts of the input sequence by assigning high probability in Equation 2.9. By incorporating an attention mechanism in the decoder, the constraint of having a fixed length context vector is mitigated. The seminal work of attention mechanism has driven remarkable progress in neural machine translation and several NLP tasks [2].

2.1.7 Attention Variants

Attentions have applications in a diverse set of NLP and Computer Vision tasks including image captioning, visual question answering (VQA), document classification, and speech recognition. There are several variants of attention mechanism [60, 61, 62, 63, 64]. The variants are not mutually exclusive and often have task-specific modifications. A hierarchical attention network [63] is applied for document classification that consists of word attention and sentence attention. [61] proposed a co-attention model for VQA that jointly reasons about image and question attention. [62] proposed soft and hard attention for neural image captioning. [64] proposed a deep variant of the attention mechanism, stacked attention networks (SANs) that allow multi-step reasoning for image QA.

Variants of deep RNNs and state-of-the-art approaches in encoder-decoder architecture have established compelling performance in sequence modeling problems such as language modeling and machine translation [1, 2]. Since then language modeling has become a central research focus in NLP to push the boundary of recurrent and sequential modeling. The fundamental constraints of recurrent modeling in the modern computing era lie in its inherent nature of sequential processing on training data which precludes parallelization of underlying computation. Some works proposed tricks and recipes for efficient training that mitigates the problem to some extent, however, they do not solve the core problem of chained computation [65]. Recently proposed attention mechanisms can disregard the sequence distance information, but in most cases, attention mechanisms are typically coupled with RNNs [2, 60].

2.2 Transformers

Deep sequence models learn contexual representations with locality bias and have difficulties in modeling long-range interactions between symbols. Due to sequential processing, the utilization of modern parallel computer hardware is limited in variants of recurrent networks. To overcome the drawbacks in existing deep learning models, Vaswani *et al.* [3] proposed a deep learning model called Transformers which is completely based on self-attention. Self-attention allows for extensive parallelization compared to RNNs and can easily model long-term contexts as every token attends to all the tokens in the input sequence. The standard Transformers model consists of a deep encoder and decoder, each of which is a stack of identical blocks. With the help of deep encoder-decoder blocks composed of self-attention and feedforward network layers, Transformers can learn complex language representations. The NLP research community has produced impressive natural language understanding systems by leveraging the power of deep Transformers and self-supervised training on internet-scale unlabeled data. For an NLP task, the supervised data is very limited and it is very challenging to build large-scale labeled dataset. Unlabeled data is abundant on the internet, however they can easily be re-purposed for network training in a self-supervised fashion.

The training instability issue in RNNs was a bottleneck for large-scale self-supervised learning on text data. Transformers on the other hand can overcome instability issues faced in training large recurrent models, and can process sequences longer than traditional RNNs thanks to no recurrence computation bottleneck, and can be scaled efficiently thanks to their parallelization on modern GPUs and successfully used for deep network training on internet-scale data that was not possible with RNNs. As a consequence, Transformers have become established as a breakthrough in modern deep neural networks. This has created a virtuous cycle in deep learning, deep networks, and larger models leading to better systems than previous ones by virtue of scaling that has no sign of reaching a plateau yet [19, 21, 66]. For instance, scaling from millions to billions of parameters in language modeling has now become a norm in NLP.

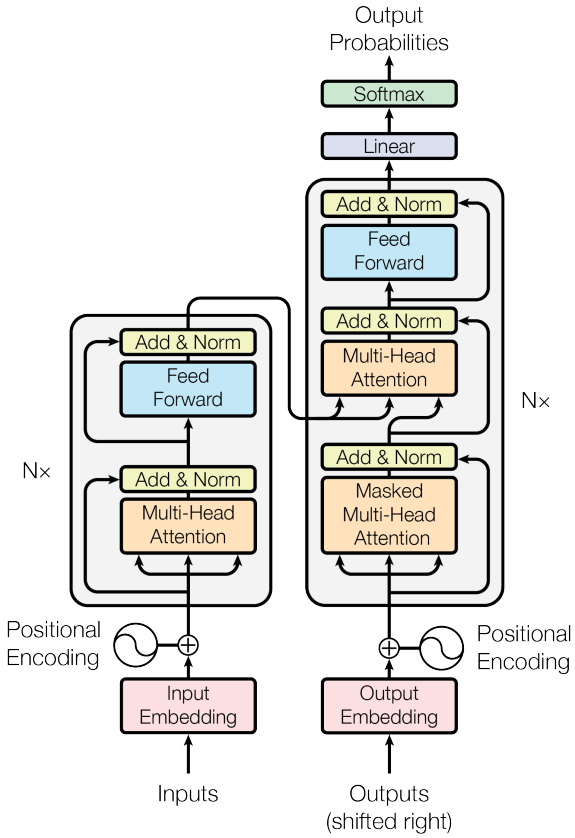


Figure 2.5: The Transformers architecture [3].

2.2.1 Transformers Architecture

The overall architecture of the vanilla Transformer is shown in Figure 2.5. Transformers follow an encoder-decoder architecture. The encoder block has a stack of $N = 6$ identical layers. Each layer has two sub-layers: i) a multi-head self-attention mechanism and ii) position-wise feed-forward networks. Similarly, the decoder has 6 identical layers. Each layer in the decoder block copies two sub-layers as found in the encoder and adds a cross-attention layer for building the connection between the decoder and encoder output representations. The number of identical layers can vary in different model sizes i.e. larger models can have more layers.

Multi-head Self-attention

In self-attention, every input token can attend the other tokens, creating a flow of information from each tokens and concurrently attending to them. The Transformer model adopts self-attention with Query, Key, and Value (QKV) layers, shown in Figure 2.6. The self-attention is a function that takes packed input matrices Q, K, and V with dimension d_k and outputs the attention matrix. First, a dot product is carried out with all queries and keys, scaled by $\sqrt{d_k}$. Then, a softmax function assigns the weights to the values.

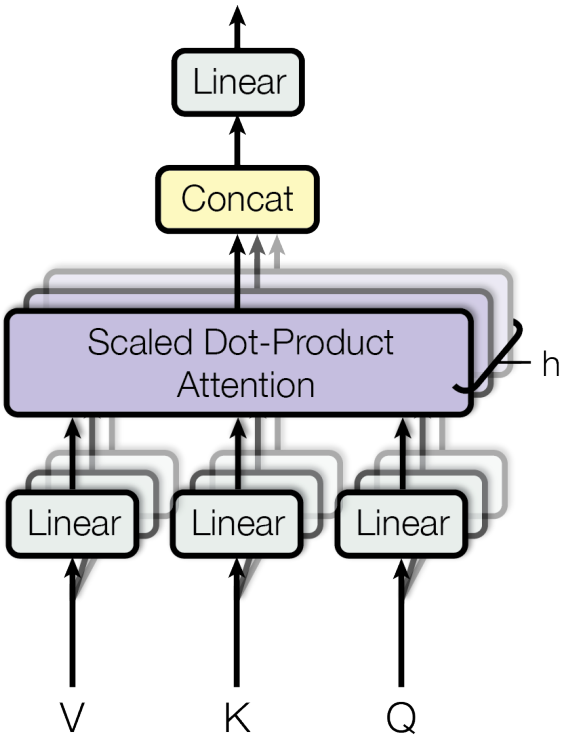


Figure 2.6: The Multi-head self-attention [3].

The scaled dot-product *attention matrix* is computed as follows:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2.12}$$

There are two common approaches to attention computation: i) additive and ii) multiplicative. The dot-product attention applied here is faster and more space efficient. However, dot-product attention in large dimensions (d_k) can cause vanishing gradients problem for the softmax function. The scaling factor $\sqrt{d_k}$ is applied to alleviate the vanishing gradients.

Transformers model use multi-head self-attention (MHSA). Instead of applying single attention, multi-head attention is used where Transformers linearly project the original d_m-dimensional queries, keys, and values h times with different sets of learned linear projections to d_k, d_k and d_v dimensions, respectively. The model computes the attention matrix on the projected keys, values, and queries in parallel, for each head, according to the identical function in Equation 2.12.

$$MultiHead(Q, K, V) = Concat(head_1, ..., head_h)W^O. \tag{2.13}$$

The resulting d_v dimensional output values for each of the heads are first combined and then projected

back to d_m-dimensional final values:

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V). \tag{2.14}$$

Position-wise Feed Forward Networks

Position-wise Fully-connected Feedforward Network (FFN) are sub-layers in each encoder and decoder block. FFN is applied to each position separately and identically. This consists of two linear transformations with a ReLU activation in between.

$$FFN(x) = RELU(xW_1 + b_1)W_2 + b_2. \tag{2.15}$$

Residual connection and Normalization

As seen in Figure 2.5, a residual connection [67] followed by layer normalization[68] is applied around each of the two sublayers in Transformers, as follows:

$$x' = LayerNorm(x + Sublayer(x)), \tag{2.16}$$

where $Sublayer(x)$ can either be SelfAttention() or FFN() implemented in the encoder-decoder block and LayerNorm() denotes standard layer normalization operation in neural nets.

Position Encodings

Unlike RNNs or convolutional networks, Transformers do not have sequence order information. To facilitate positional information in a sequence, the Transformers model injects some information about the relative or absolute positions of the tokens in the sequence.

It is easy to verify that recurrent or convolutional neural networks are not permutation invariant; output should depend on the order of the inputs. However, both self-attention and feed-forward networks are permutation invariant. This becomes problematic for most NLP tasks since the structure of inputs is desired. For instance, word order matters for modeling a sequence of text, and Transformers must encode the position of words. To facilitate this, "positional encodings" vector is infused into the input embeddings vector of both the encoder and decoder. In vanilla Transformers, positional encoding corresponds to a sinusoidal (sin/cos) function. However, there are several choices for positional encoding in subsequent literature [21, 69, 70].

2.2.2 Transformer Usage, Analysis and Comparison to Other Network Types

Transformer architecture is decomposable and can be used in three ways based on the use cases: i) encoder-decoder, ii) encoder only and iii) decoder only. The encoder-decoder represents full Transformers architecture depicted in Figure 2.5, commonly used for seq2seq modeling tasks such as language modeling and machine translation. For classification or sequence labeling tasks only the encoder representation is used. A common

Table 2.1: Maximum path lengths, per-layer complexity and minimum number of sequential operations for different layer types. n is the sequence length, d is the representation dimension, k is the kernel size of convolutions and r the size of the neighborhood in restricted self-attention.

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

choice is to feed the output of an encoder's last layer as an input to a task-specific classification layer. The decoder-only architecture is typically used for sequence generation tasks (e.g. language modeling). Transformers was first invented in machine translation, later their success has spanned to diverse domains: NLP [21], computer vision [71] and reinforcement learning [72].

The Transformers model has quadratic time complexity $O(n^2)$ with respect to sequence length [3]. There are recent studies that try to linearize the attention mechanism for computation efficiency. The comparison to other network types is presented in Table 2.1.

2.3 Neural Language Models

Language models are a fundamental component of many NLP tasks, including sentence classification, sequence labeling, question answering and language generation [73]. A language model (LM) can be used to learn the structure of language by predicting the next word in a sequence, and produce general-purpose language representations [16]. In this section, we will discuss the current state-of-the-art language models, including BERT [21], GPT-3 [19], and encoder-decoder variants of LMs [74]. These models are typically neural networks that are trained on large amounts of data based on some self-supervised pre-training objectives. We will also briefly discuss some applications of language models in transfer learning of downstream NLP tasks.

2.3.1 Language modeling

A language model predicts the probability of next word given previous words. In other words, a language model assigns probability to a sequence of words. Let's assume we want to model a sentence $x_1, x_2, .., x_T$ that consists of T tokens.

$$p(x) = p(x_1, .., x_T) \tag{2.17}$$

Here $p(x)$ is the probability distribution over strings or sentences in a fixed vocabulary V. Using the product rule of probability we can write as follows:

$$= \prod_{i=1}^{T} p(x_i|x_1, .., x_{i-1}). \tag{2.18}$$

The probability is decomposed into conditional probabilities of predicting each word given the previous words. Here, we modeled a left-to-right language model where each word in the sequence is predicted from left to right.

As in Equation 2.18 a neural language model needs two things: i) process context (i.e previous words) and ii) generate probability distribution for the next token. The context processing can be done with RNNs since they can model sequences and the next token probability assignment is a classification task over the giant vocabulary set. We can use an RNN to learn a probability distribution over a sequence by training it to predict the next symbol in a sequence. The dependencies between words are characterized by the dependencies between states in the RNN model. The parameters in an RNN are shared in different positions (time steps of the sequence), also know as *weight sharing*, which helps it to generalize to sequence of varying length. The training objective for neural LMs is pretty straightforward – we want to maximize the probability a model assigns to the correct next word.

2.3.2 Pretrained language models

Language modeling has a rich history, connecting to works done by Andrew Markov, Claude Shannon, and Noam Chomsky [75, 76]. In 2001, Yoshua Bengio and his co-authors proposed one of the first neural language models [16]. The neural LM proposed by Bengio *et al.* improves n-gram language models by learning a *distributed representation* for words and opened a new era of language modeling. Their n-gram model has an MLP architecture and can model only a small context window. Later, a representative set of language models are developed that can capture long-term contexts using variants of recurrent networks, including LSTMs [32].

A language model can be used to generate language (symbols, tokens, words) by random sampling from the language model. This makes tasks such as machine translation, dialogue generation, and text summarization most plausible to be based on language modeling. Moreover, the conditional language modeling can be transformed as a sequence-to-sequence learning problem [1]. A successful seq2seq model is Transformers architecture, which is widely adopted to develop many state-of-the-art pre-trained LMs [19, 21, 74, 77] for several architectural benefits (see discussion in section 2.2).

The basic idea of a pre-trained language model is as follows. First, a language model is implemented, for example, following the encoder-decoder architecture in Transformers [3]. The model learns in two phases:

pretraining and fine-tuning. A pre-training phase trains the model using some self-supervised learning objectives on a large corpus, and fine-tuning adapts the pre-trained model by supervised learning on a small amount of labeled data for a specific task. A large-scale pre-training helps to produce general-purpose and transferable distributed representation for words [16, 21].

In this section, we briefly introduce the three most common types of pre-trained language models: encoder-decoder language models, encoder only language models, and decoder only language models. We discuss the strengths and weaknesses of each model and provide examples of each.

- **Encoder-Decoder LM**. The encoder-decoder language model is the most common type of language model (e.g. T5[74], BART[77]). The encoder-decoder language model consists of two parts: an encoder and a decoder. The encoder encodes the input sequence of words into a fixed-length vector. The decoder then decodes the vector into an output sequence of words. They are typically used for tasks such as machine translation and speech recognition.

- **Encoder only LM**. BERT[21] is an example of an encoder only language model. Encoder only language models only have an encoder; they do not have a decoder. They are mostly suitable for classification and sequence labeling tasks.

- **Decoder only LM**. Decoder only language models only have a decoder. They cast language modeling as a text generation problem. The most notable example of a decoder only language model is GPT-3 [19].

2.3.3 BERT: Bidirectional Encoder Representations from Transformers

In seminal work [21], Devlin *et al.* proposed BERT: a pre-trained deep bidirectional transformer for language understanding. BERT is a large pre-trained model trained on large-scale web data using semi-supervised pretraining objectives. The two pretraining objectives used are masked language modeling (MLM) and next sentence prediction (NSP).

In MLM, the model is given a sentence with some words masked and is tasked with predicting the masked words based on the context. NSP predicts whether the second sentence is the next sentence given a pair of sentences. MLM allows the model to learn general-purpose language representations, while NSP helps the model learn representations that are useful for tasks that require understanding of the relationship between two sentences, such as question answering and natural language inference.

Fine-tuning BERT, a paradigm shift in NLP

The BERT paper is a paradigm shift in NLP. The pre-trained BERT learns powerful language representations that can be successfully applied to many NLP tasks. The knowledge learned in the pretraining stage can be transferred to solve an NLP task (such as sentence classification, and question answering) by an approach called fine-tuning [78]. This is done by adding a classifier layer on top of BERT layers and training further

using task-specific labeled data. The fine-tuning of BERT has reached wide applications in NLP where labeled data is limited such as biomedical, health, and economy domains. Since then 'pre-train fine-tune' has become a de-facto standard for solving language tasks [79].

The method of pre-training language models on a large neural network with internet-scale unlabeled data and fine-tuning in downstream tasks has made a breakthrough in several natural language understanding tasks in recent years [19, 74]. For instance, state-of-the-art language models have already achieved performance better than reported human accuracy on two widely used language understanding (NLU) benchmarks, GLUE, and SuperGLUE. A large collection of work now established that as the model grows bigger benchmarks become obsolete. By simply improving the language modeling recipes found in BERT and building on deep Transformers networks, we now have billions to trillions scale of parameters in language models. Such general-purpose language models have shown to be a zero-shot learner, solve a task 'in-context' without requiring any labeled data [19]. However, this requires large-scale data, increased parameters in deep neural networks, and millions of dollars of GPU computation in training which might not be feasible except for a few industry labs.

2.3.4 Multilingual Pre-trained Language Models

The success of pre-trained language models mostly contributed to progress in NLP for English and a few high-resource languages. A natural question is how to integrate such benefits toward low-resource languages and bridge the gap of systemic inequality in language technology across the world's languages [26]. One alternative which gained popularity in recent years is to train a multilingual pre-trained model (MPLM) that comprises training datasets from many languages – both high and low-resourced ones. Multilingual modeling typically uses a shared vocabulary across languages built from sub-word tokenization [80]. A representative set of state-of-the-art multilingual pre-trained language models are: mBERT[21], XLM[81], and XLM-R[82]. They differ mostly in the architecture (number of layers, parameters, etc), pre-training objective (monolingual masked language modeling, translation language pair), size and sources of pretraining data (Wikipedia, CommonCrawl), and number of languages involved (ranging from 10 to 100). For instance, in mBERT, authors trained a BERT model sourcing training data from the top 100 languages with the largest Wikipedia entries.[1] Multilingual pre-trained LMs are more efficient (one model to support all languages) and inclusive (more languages included in training). Studies that investigate MPLM's effectiveness in benchmark tasks suggest that shared multilingual models improve performance in resource-lean languages, specifically in limited data settings [83, 84]. A shared subword across languages and joint multilingual training are the key components of mBERT's surprising generalization capacity across languages.

[1] `https://github.com/google-research/bert/blob/master/multilingual.md`

2.4 Transfer learning in NLP

Transfer learning is a method of using a pre-trained model to initialize a model for a new task [85, 86]. Transfer learning can be used to improve the performance of a model on a new task, or to reduce the amount of data needed to train a model on a new task. Transfer learning stirred up immense success in computer vision with deep ConvNet models after 2012. A pre-trained ResNet model trained on the popular ImageNet dataset is used as pre-trained weight initialization for solving image recognition tasks of varying nature and domains [15]. Starting with early pre-trained word embeddings, transfer learning in NLP became ubiquitous with the rise of large pre-trained LMs [21, 78, 87]. Substantial improvements over previous state-of-the-art results on the GLUE benchmark tasks have been obtained by recent works on the large language models such as BERT, and GPT-3 with task-specific fine-tuning. Though transfer learning is not a recent phenomenon in NLP, for the sake of discussion here, the focus is mainly on transfer learning by leveraging pre-trained language models.

2.4.1 Word embeddings and contexualized representations

Early successful transfer learning in NLP is found in the use of word embeddings. Word embeddings, also known as word vectors are lookup tables that represent the meaning of a word in some abstract manner, typically a vector of weights for each word in a predefined vocabulary. Word vectors compute semantic similarity in a fixed dimension over a large vocabulary set. These word vectors provide rich distributed representations for words and can be used as features in a variety of applications including information retrieval, parsing, and named entity recognition [87]. The embeddings provide static features on an input sentence for the first layer of a neural network. It is then combined with further neural network layers (e.g. RNN) which enables contextualized learning of a task using supervised data. An extension to this direction is presented in learning contextualized word representations in Embeddings from Language Model (ELMO) [17]. Where early embeddings are static, contextualized representations of embeddings can be learned using a language modeling approach. Such contextualized representations have achieved superior performance in many NLP tasks [17, 78].

2.4.2 Recipes for Successful Fine-tuning of a Pre-trained Language Model

There are several choices for effective knowledge transfer of pre-trained language models. There is no single all-rounder method for transferring effectively. It conditionally depends on the specific tasks and choice of pre-trained models. Howard *et al.* [78] proposed several fine-tuning methods called ULMFiT with an extensive analysis of language models fine-tuning in sentence classification tasks. It provides rules-of-thumb lessons while fine-tuning a language model for downstream applications. Some approaches studied in seminal ULMFiT are *discriminative* fine-tuning where learning rates are different for each layer of the network, target task classifier fine-tuning with *gradual unfreezing* where network layers are unfrozen gradually, *full* fine-tuning

of all the network layers, and fine-tuning only the *last* few layers. All of these tactics bring some improvement over training from scratch with varying success. Catastrophic forgetting is a common problem in sequential (full) fine-tuning. Fine-tuning tricks such as freezing some network layers and training only the last few layers might be beneficial to mitigate the catastrophic forgetting problem. Similar fine-tuning approaches can be applied to BERT-like models [79]. The standard BERT model has 12 layers and freezing can be done on a subset of layers. A detailed analysis on fine-tuning for text classification shows top layers are more transferable than lower layers and suitable learning rates are critical for functional transfer [79] .

2.4.3 Cross-lingual Transfer Learning

Most world languages are considered low-resource in NLP. Languages that have limited resources such as corpus, datasets, models, and research are deemed low-resource. This has led to systematic inequalities in language technology and more attention is needed to tackle the gap – which is also one of the core motivations of this thesis. Pre-trained language models such as BERT, and GPT-3 have shown to be effective in many downstream NLP tasks. The success of large pre-trained models also surfaced in multilingual arenas with the promise of zero-shot cross-transfer generalization for low-resource languages. Multilingual modeling usually comprises high-resource (source) and low-resource (target) languages enabling knowledge transfer from source to target languages. It involves a cross-lingual representation learning setting where high-resourced information can be transferred to learning low-resource tasks. This notion is called "interlingua" which enables cross-lingual transfer generalization. Recently, many attempts have been made to learn cross-lingual representations with increased interest. Notably, mBERT[21] is the early work in this area where the authors trained a deep transformer-based BERT model for 100 languages. In recent years, mBERT-like models were successfully applied to zero-shot cross-lingual transfer yielding research focus on machine translation to entity linking, word sense disambiguation (WSD), Natural language inference (XNLI), Question Answering (TyDiQA), paraphrase reasoning (XCOLA), etc [83]. Learning from small or no data, respectively, few-shot and zero-shot learning is a notable research inquiry in the multilingual NLP community. We present some discussion below on the limited data learning approaches using large pre-trained models.

2.4.4 Transfer Learning Approaches on Limited Data

Zero-shot Learning

Zero-shot learning is a method of training a model on a new task without any labeled data for the new task [88]. Zero-shot learning can be used to train a model on a new task without the need for a large amount of labeled data. In many multilingual tasks, we may not have labeled data for a target language. In practice, we train models using supervised data from high-resource languages collected from a task of interest. The trained model is then used directly for predicting on test set of a target language without further fine-tuning. Moreover, some autoregressive lagnuage models can directly be used without any task-specific fine-tuning

[74]. This evaluation procedure is considered as zero-shot learning [27, 84, 89].

Few-shot Learning

Few-shot learning (FSL) is a method of training a model on a new task with a small number of labeled data items [90, 91, 92, 93, 94, 95, 96, 97]. FSL is mainly constrained by the lack of available training data, real-world scenarios where large supervised data is impossible to acquire due to privacy, safety or ethic issues. Where standard deep learning model training requires large-scale datasets in a supervised setting, few-shot learning specifically investigates learning from a few training samples. This setting is further popularized in the 'pre-train fine-tune' paradigm under the assumption that large-scale fine-tuning produces a generalized model and downstream tasks can be learned with few samples, imitating the notion of how humans learn a task. There are several approaches to few-shot learning: metric learning [92], meta-learning [96] and in-context learning [19]. Learning in a low-resource multilingual setting is also a reasonable real-world setting for FSL as labeled data is scarce for many world languages.

Meta-learning

Meta-learning, "learning to learn," is a few-shot learning paradigm for solving machine learning tasks with small to zero-labeled data. Model-Agnostic Meta-Learning (MAML) [98], a gradient-based meta-learning optimisation algorithm allows faster adaptation on novel tasks. It has an *inner loop* and an *outer loop*. The inner loop learns a machine learning task in each iteration and the outer loop learns a meta-objective. This meta-learner objective results in improved initialization for a novel task adaptation. It follows episodic training and testing and mimics learning multiple tasks so that the meta-learner learns to deal with fast adaptation to a novel unseen task. Meta-learning was successfully used for solving many vision tasks and has also applied to some NLP tasks in the last few years.

Meta-learning is of recent interest for few-shot learning of multilingual NLP tasks. It is hard to achieve sufficient labeled datasets for low-resourced languages. Usually, a model fine-tuned using small data on a low-resource target task suffers from overfitting. Since meta-learning model allows faster adaptation, it is expected to produce better initialization while adapting to a low-resource language using limited training samples. Much of the work in multilingual NLP explored meta-learning as a training algorithm for solving zero or few-shot classification tasks. X-MAML [99] uses MAML for meta-training cross-lingual NLP tasks (e.g. Natural Language Inference (NLI) and Question Answering (QA)). The authors experiment with varying auxiliary languages to find the best language pair composition to achieve the best performance in the target. Gu *et al.* developed a Meta-NMT [100] system, the first attempt to use meta-learning in multilingual machine translation. The authors applied MAML to low-resource machine translation by viewing language pairs as separate tasks. The proposed model has two essential components: i) *learn-* the language-specific part which is learning a language translation task from meta initialization. This is very similar to training any supervised task in ML from an initialization point. ii) *meta-learn* - define a meta-objective and maximize using meta-

gradient updates. Meta-learning powers this idea of a meta-objective to be defined and updated using SGD which ensures better initialization of novel tasks. Specifically, it finds a better initialization on target by repeatedly simulating low-resource translation scenarios using auxiliary, high-resource language pairs. The resulting meta initialization (pre-trained model) will then be fine-tuned for low-resource translation of target language pairs.

2.4.5 Multi-task Learning

Multi-task learning (MTL) has a rich history in machine learning [101]. Two categories of approaches are frequently explored: a) multi-task feature learning, which learns a shared representation [102, 103, 104] and b) multi-task relationship learning, which learns to task grouping or measures task covariance [105, 106, 107]. A general approach shares most parameters across tasks but assigns some task-specific parameters [102].

Natural language processing tasks can form a clear hierarchy, for instance, $POS \rightarrow NER \rightarrow Entailment$. A joint-many-task model [108] assumes such a hierarchy exists and develops a multi-task model guided by a linguistic hierarchy of NLP tasks. The task hierarchies are POS tagging, parsing, relatedness, and textual entailment respectively. The architecture is hand-designed, stacking LSTM networks at multiple layers forming a hierarchy of inputs to the model at different layers in an incremental fashion. Another line of work considers methods that share all the parameters across tasks to produce a general-purpose model. This requires the model to have same the input and output space, as such, casting all the tasks into a form of question answering (NLP Decathlon, McCann *et al.*, [104]). The Multi-Task Deep Neural Networks (MT-DNN) [103] model utilizes pre-trained BERT models for multi-task learning on several NLU tasks. The lower layers are shared across all tasks, while the top layers represent task-specific prediction heads. However, parameter sharing is an act of art in multi-task learning and has been explored frequently. We have many choices when it comes to splitting the shared and task-specific components of a network. This requires a lot of trials and model tuning to train successfully. Outside of the NLP domain, parameter sharing was extensively explored in the computer vision community [86, 105, 106, 109].

Domain Adaptation

Domain adaptation is studied to build models that are robust to distribution shift problem, which arises when a mismatch is found in source and target domain distributions. Studies on domain adaptation can lie in both supervised [110, 111, 112], where labeled data is available for both source and target, and unsupervised approaches [113, 114], where no target-labeled data is not found. The general assumption is a model trained on source domain X fails to generalize on target domain Y. Source domain is the domain where a model was initially trained on. The target domain is the case where we have very limited or no labeled data. This research study approaches to learn under two kinds of shift e.g. label shift and co-variate shift. Early works on domain adaptation focus on data augmentation, and semi-supervised learning. One common interest in domain adaptation is to learn a shared representation (invariant) of both domains using neural nets and

use it for adapting to the target domain. A series of works address this problem via explicitly modeling domain-invariant representation learning during training e.g. introducing kernel Hilbert space embedding for task-specific layers [115], domain adversarial training of neural layers [116, 117]. Gu *et al.* [117] attempts to match feature space distributions via adversarial training towards domain-invariant representation learning.

3 Model-agnostic Meta Learning for Low-resource Hate Speech Detection

3.1 Introduction

Motivation. Online hate speech that expresses hate or encourages violence towards a person or a group based on characteristics such as race, religion, or gender is a growing global concern. The hateful content has broken the cohesiveness of online social communities and resulted in violent hate crimes. [1] Therefore, detecting and moderating hate speech in online social media is a pressing issue.

The gravity of the situation has motivated social media platforms and academic researchers to propose traditional machine learning and deep learning solutions to detect online hate speech automatically [23, 118]. Among the solutions, large pre-trained language models (LMs), which have demonstrated their superiority in many NLP tasks [21], have also shown excellent performance in the hate speech detection task [119]. However, existing studies have predominantly focused on detecting hate speech in English and content from Western cultures. This neglects the large portion of online hate speech from other languages and cultures. Low-resource languages often have very limited or no training samples available. This can make it challenging to develop supervised classifiers. We need models that can efficiently adapt with few training data, or methods that can transfer knowledge from another dataset. To meet the challenge, a high-resource language can augment the low-resource language by knowledge transfer. For example, Aluru *et al.* [120] proposed a method to fine-tune a multilingual BERT model for hate speech detection in low-resource languages that leverages cross-lingual transfer from high-resource pre-training.

In multilingual hate speech detection, English and a few high-resource languages are considered "source," and low-resource languages of interest are considered "target." There are very few works that deal with multilingual hate speech detection. A viable approach is to fine-tune pre-trained LMs, which is explored in existing studies [119, 120, 121]. The underlying intuition is that the large LMs generate shared embeddings in many languages, enabling cross-lingual transfer from supervised training in the *high-resource* languages such as English. Such cross-lingual transfer can be achieved as follows: (i) training only in the source language and evaluating in the target language, known as 'zero-shot,' or (ii) training in the target language only or joint training of source and target languages, known as 'few-shot' learning. A source dataset is usually a magnitude larger than a target dataset. However, fine-tuning pre-trained LMs also has several limitations [122]. An

[1] https://thewire.in/law/hate-speech-what-it-is-and-why-it-matters

obvious issue is the data scarcity in target low-resource languages; it is challenging to capture hate speech and its nuances with insufficient observations. Also, the performance of cross-lingual transfer learning may adversely be affected if source and target languages are from distant language families [122, 123].

Hate speech datasets can come from many domains, which can be divided into two categories: one comes from language shift, and the other is content in each language. For instance, a dataset in one language may be collected from the Twitter platform, while another could be collected from Youtube. Standard fine-tuning overfits the training languages and will face the difficulty of adapting to a new domain or an unseen language [78]. Conneau *et al.* [82] show that adding more languages improves the performance on low-resource languages but hurts the performance on high-resource languages. We adopt a meta-training strategy for low-resource hate speech detection to address these concerns. Meta-learning has been successful in low-resource settings and can be used to learn a good initial point for fine-tuning a new task, making the fine-tuning process more efficient.

Research objectives. In this thesis, we aim to tackle the challenges of multilingual hate speech detection and address the limitations of existing fine-tuned pre-trained LMs in hate speech detection. We propose HateMAML, a model-agnostic meta-learning-based framework that performs hate speech detection in low-resource languages. Unlike existing hate speech detection approaches that fine-tune LMs on the source and target only, we focus on resource maximization by using resources in other auxiliary languages beyond source and target. More specifically, we adopt the model-agnostic meta-learning (MAML) [98] strategy, in which simulated training in few-shot meta-tasks produces a meta-learner that can learn a target task quickly. Recent studies have explored adopting MAML in low-resource cross-lingual adaptation for NLP tasks [99, 100].

The underlying intuition of HateMAML is to train on auxiliary languages and adapt the LM to detect hate speech in a target language in which there are few or no datasets available [99]. To address diversity and nuance in detecting hate speech across languages and domains, the meta-learning loss function in HateMAML is extended with a domain adaptation loss. Intuitively, a task-specific learner evaluates their own training samples and samples from another language. This mimics the scenario we are interested in: training in one language while evaluating an unseen language during test time. This training strategy forces the meta-learner to generalize to a target language domain while learning from source domain samples.

Considering the data scarcity of non-English languages, we further propose a self-guided meta-learning-based fine-tuning mechanism. This mechanism utilizes unlabeled examples of the low-resource target language and generates pseudo-labels for fine-tuning the target. Using training data from high-resource English, we first construct a multilingual predictor model. The predictor model predicts unlabeled samples in a target language (e.g. Spanish). The predicted samples are filtered for quality. Each filtered sample has a pseudo-label, and a subset of these is used as training data for the target language. We also adapt HateMAML into an iterative self-refining loop that replaces and refines the current predictor model and improves the quality of the adapted model in the target.

Finally, we investigate whether meta-training is better than standard multilingual fine-tuning in two

scenarios: when training examples are only available for some languages and when both high and low-resource languages have sufficient training data. The first scenario is a stress test for cross-lingual domain adaptation, evaluating the trained model on a set of reserved low-resource languages. From a set of low-resource languages, a subset is selected for training and the remaining is used for zero-shot evaluation. The other scenario explores meta-training benefits over standard fine-tuning. In early studies, cross-lingual meta-training was mostly limited to language pairs [99]. This setting is, however, not truly multilingual since we need separate models for each target language. The effectiveness of meta-training is examined when training data is available in more than one low-resource language. Though the early investigation was limited to one auxiliary and target language, the current configuration allows us to investigate cross-lingual meta-training for hate speech detection at scale. In other words, we adopt meta-training as a proxy for general multilingual fine-tuning. Our results show that when training data is available from many languages, meta-training HateMAML yields better performance than standard fine-tuning.

Contributions. To the best of our knowledge, HateMAML is the first model that adopts a meta-learning framework for multilingual hate speech detection at scale. Our major contributions are summarized below:

1. We propose HateMAML, a domain-adaptive MAML algorithm for hate speech detection in low-resource languages. Our algorithm outperforms state-of-the-art cross-lingual fine-tuning baselines. One of the novelties of HateMAML lies in the efficient utilization of available samples in the validation set of source and training data in auxiliary languages.

2. To mimic an extreme data-scarce scenario in the target language (zero-shot), we introduce a self-refining variant of HateMAML. An iterative meta-learning approach is adapted to generate pseudo-labels from unlabeled data in the target language.

3. Across eight different low-resource languages, we carry out extensive experiments in zero-shot and fine-tuning settings. Our experimental results show that HateMAML consistently outperforms state-of-the-art methods by more than 3% in the cross-lingual transfer settings.

4. Our ablation studies examines the contribution of an auxiliary language across each *auxiliary-target* language pairs. We also assess the algorithm's robustness by varying (i) the amount of meta-training samples, (ii) cross-domain adaptation, and (iii) one model to support all language configurations.

5. We conduct cross-domain hate speech detection to address data set diversity in various languages and show that HateMAML achieves a significant domain adaptation. Our experiments on meta-training over all available language training data suggest that meta-learning could be an alternative to standard fine-tuning that yields superior performance.

3.2 Related Work

Multilingual hate speech detection In the last few years, hate speech detection has gained significant

attention, with the majority of work dedicated towards monolingual hate speech [23, 118]. This skew in attention has naturally led to the majority of datasets being in English [10, 11, 124]. However, hate speech is a global phenomenon, and increased diversity in available resources is critical for developing automated systems. Recent efforts addressed this issue, including multiple shared tasks (such as SemEval 2020 [28], HASOC 2020 [125], Evalita 2018 [126] and HateEval 2019 [22]). Such efforts helped in bringing much-needed traction toward multilingual hate speech research. Additionally, recent development in transformer-based large multilingual LMs (mBERT [21], XLM-R [82]), pre-trained on more than 100 languages, has aided in developing state-of-the-art classifiers in resource-lean languages. Prior studies on multilingual hate speech detection have explored: (i) resource development such as creating datasets [22, 28, 29, **?**], shared tasks and organizing workshops, (ii) cross-lingual transfer learning with multilingual shared embeddings and pre-trained LMs [120, 121], (iii) utilization of additional features from relevant domains [127] (e.g., emotion, sentiment), and (iv) data augmentation techniques including use of external service (e.g., translation APIs) for supervised training [121, 128, 129].

Cross-lingual and multilingual transfer. Knowledge transfer from a high-resource language (i.e., English) to a low-resource one (i.e., Hindi) has been shown to be an effective resource optimization technique. Ranasinghe and Zampieri [119] used a two-step training method: first training multilingual transformer models in English, and then the resulting model is fine-tuned on the low resource target language. Such multi-phase training was found to be a better model initialization for fine-tuning the target. Aluru *et al.* [120] conducted an extensive study on state-of-the-art multilingual embeddings including LASER [89] and MUSE [81]. Another line of studies [121, 130, 131] utilized translation-based solutions to alleviate the shortage of data in low-resource languages in hate speech. These models rely heavily on translation APIs, which might be susceptible to producing low-quality translations. Moreover, the translation of a large source corpus can be expensive.

Majority of the studies on cross-lingual and multilingual hate detection fall into pre-trained model fine-tuning [130, 132, 133]. However, Nozza [122] found that due to the domain shift of hate speech across languages, standard fine-tuning methods are less effective for both zero-shot and few-shot settings. To address the limitation of low-resource language fine-tuning, we develop HateMAML for fine-tuning hate classifiers with limited data and an improved mechanism for domain-adaptive cross-lingual transfer.

Meta-learning. Meta-learning, also known as "learning to learn," is a few-shot learning technique geared toward learning few-shot tasks and fast adaptation. Unlike standard supervised fine-tuning on downstream tasks that result in a *final* classifier, meta-learning focuses on producing a rich initialization point from where an unseen target task can be learned quickly. Some common meta-learning approaches include (i) optimization-based [98, 134], and (ii) metric-based [135, 136]. Finn et al. [98] introduced model-agnostic meta-learning (MAML), an optimization-based meta-learning framework for few-shot tasks. Several recent studies have explored gradient-based meta-learning for few-shot text classification [90, 94, 97]. Meta-learning for domain generalization is also studied to handle domain shift during training in diverse problems, including

supervised learning and reinforcement learning [137]. Meta-learning has also been explored in zero-shot and few-shot cross-lingual settings. [100] successfully adapted MAML for low-resource neural machine translation (NMT) tasks using auxiliary languages and achieved competitive performance using only a fraction of training data. Gu *et al.* [99] proposed X-MAML for meta-training cross-lingual NLU tasks. Similar to our interest, X-MAML experimented with varying auxiliary languages to find the best composition for zero-shot cross-lingual transfer. However, most multilingual hate speech datasets comprise training data in only two or three languages, making searching for a good auxiliary language extremely difficult.

Data augmentation/self-training. Data augmentation approaches can be divided into three categories: (i) rule-based [138], (ii) interpolation-based [139, 140, 141], and (iii) model-based [142, 143, 144]. In cross-lingual hate speech, translation techniques [121, 130, 131] are commonly used for data augmentation during training. Rather than collecting and annotating new hate speech data, bootstrapping on unlabeled samples to create pseudo-labeled data provides a way for data augmentation and fine-tuning on low-resource languages. We draw inspiration from Bigoulaeva *et al.* [128] to devise a bootstrapping-based self-training approach for HateMAML.

How is our approach different? We explore cross-lingual meta-training in the domain of hate speech for both zero-shot and fine-tuning configurations. Our proposed method of zero-shot meta-training, HateMAML, is a novel idea that can be further adapted for languages with no availability of labeled data. The proposed modeling focuses on resource maximization and domain generalization while transferring task-specific knowledge to low-resource languages. We carry out a large-scale study in multilingual hate speech detection across diverse domains on available hate speech datasets.

3.3 Multilingual Hate Speech Detection

We devise a gradient-based meta-learning algorithm named HateMAML for multilingual hate speech detection. Our proposed algorithm is tailored to improve cross-lingual transfer in target low-resource languages. We create a set of meta-tasks from samples for both high- and low-resource languages and simulate episodic meta-training similar to MAML [98]. The goal is to produce better initialization of model parameters that can adapt to a low-resource target language task using (i) none (i.e., zero-shot) or (ii) only a small number (i.e., few-shot) of labeled examples. An auxiliary language is selected to (meta-)train the model in the zero-shot setting. In the few-shot setting, we meta-train the model on the target language. We assume some labeled data (e.g., 2024 samples) is available for supervised fine-tuning on the target language. Hate speech datasets from different languages often comprise several domains and topics. For example, while many English datasets capture the domain of racism-related hate, hate speech datasets in Hindi are more religion/politics-oriented. Here, the term 'domain' is loosely used, referring to different datasets and languages. Therefore, the task of cross-lingual hate speech detection implicitly encompasses cross-domain adaptation. The objective is to *adapt quickly to both target language and domains.* To account for this, the proposed meta-adaptation model

includes a domain generalization loss. The idea is to produce a good initialization that can perform well on diverse domains.

3.3.1 Model Agnostic Meta-Learning (MAML)

MAML assumes a probability distribution $p(\mathcal{T})$ of tasks $\{\mathcal{T}_1, \mathcal{T}_2, \cdots, \mathcal{T}_m\}$. A meta-model $f_\theta(\cdot)$ with parameters θ is learned through episodic meta-training over sample tasks $\mathcal{T} \sim p(\mathcal{T})$. MAML has two loops: (i) an inner loop for task-specific adaptation and (ii) an outer loop to *meta-learn* that fast adapt to an unseen novel task.

MAML fine-tunes the meta model f_θ to a particular task $\mathcal{T}_i$ through gradient descent:

$$\theta_i' \leftarrow \theta - \alpha \nabla \mathcal{L}_{\mathcal{T}_i}(f_\theta), \tag{3.1}$$

where α is the step size, and $\mathcal{L}$ is the classification loss. The task-specific training outcome is evaluated on the associated test set.

The meta-learner optimization objective is to minimize the *meta loss* computed from all the training tasks:

$$\min_\theta \sum_{i=1}^{m} \mathcal{L}_{\mathcal{T}_i}(f_{\theta'}) = \sum_{i=1}^{m} \mathcal{L}_{\mathcal{T}_i}(f_{\theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)}). \tag{3.2}$$

The meta parameter θ is then updated:

$$\theta = \theta - \beta \nabla_\theta \sum_{i=1}^{m} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'}), \tag{3.3}$$

where β is the meta-learner learning rate. Multiple episodes can be accumulated to update θ. Though MAML training requires double gradient descent optimization, a first-order approximation is used in practice. Unlike standard fine-tuning, meta-training does not result in a final model. However, it is a reasonably good starting point (initialization) from which learning a new task can be executed quickly. Intuitively, training across a set of meta-tasks can be seen as auxiliary; fast adaptation on the unseen target task is the main goal. A few recent studies build on this motivation [99, 100] where supports from auxiliary languages are used in cross-lingual meta-training. This is also a key inspiration for the model.

3.3.2 Proposed Model: HateMAML

The training of the proposed HateMAML model facilitates hate classifier adaptation and cross-lingual transfer in target languages while requiring only a small number of samples for model fine-tuning. HateMAML can be used for training with/without labeled data, only requiring access to training data in the source language for the initial fine-tuning. Any multilingual encoders that have shared embeddings, such as mBERT and XLM-R can be used as the base model. To further improve cross-lingual transfer, a few labeled samples is used in the target language (few-shot training), which helps quick adaptation. Additionally, HateMAML benefits from

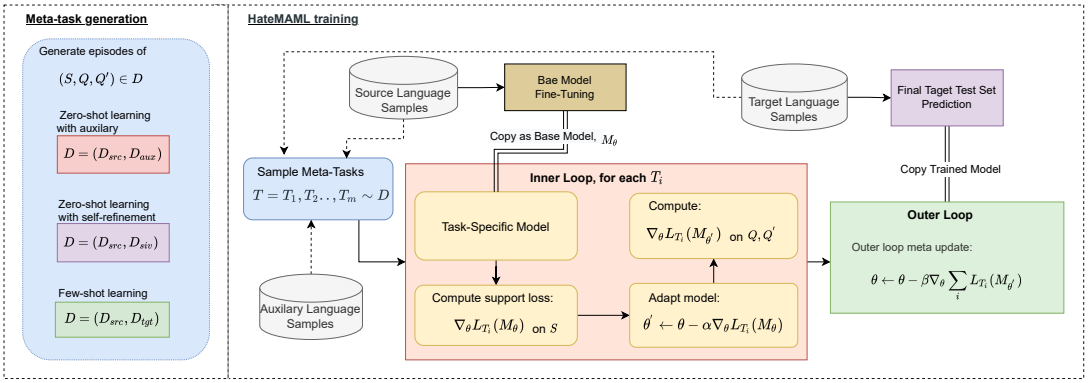


Figure 3.1: Proposed HateMAML model sketch. (Left) Model training in zero-shot and few-shot setups. We use an auxiliary language validation set in zero-shot learning, and target language labeled data is used in few-shot. In both cases, we include a source language validation set and create a (virtual) domain set that helps us to mimic target domain performance evaluation during meta-training. We sample a batch of triplets $(\mathcal{S}, \mathcal{Q}, \mathcal{Q}')$ and simulate episodic training. (Right) Self-training procedure with HateMAML, which produces pseudo-labels from unlabeled data in the target language, model training on those, replace base model. Repeat the procedure N times.

task-specific training evaluation on a (virtual) domain test (query) set at each episode that mimics target domain evaluation during meta-training [137]. The intuition of the domain query set is to achieve training generalization in unseen target languages and domains. We discuss training datasets accumulation, meta-tasks generation, and domain loss in HateMAML and present a concise training procedure in detail below. A rough sketch of HateMAML can be found in Figure 3.1.

HateMAML Training Data. A set of support and query batches is sampled from $\mathcal{D}$, where $\mathcal{D}$ refers to available training data from training languages, generally small in size. Suppose we have samples in source ($\mathcal{D}_{src}$), auxiliary ($\mathcal{D}_{aux}$) and target ($\mathcal{D}_{tgt}$) languages. The training data $\mathcal{D}$ consists of the tuple: (i) $(\mathcal{D}_{src}, \mathcal{D}_{aux})$ in the zero-shot setting, and (ii) $(\mathcal{D}_{src}, \mathcal{D}_{tgt})$ in the few-shot setting. To this end, we split the training set for each language into a support ($\mathcal{D}_{lang^{support}}$) and query ($\mathcal{D}_{lang^{query}}$), where $lang \in \{src, aux, tgt\}$. We randomly sample a (virtual) domain set $\mathcal{D}^{domain-query}$ from either $\mathcal{D}_{aux}^{query}$ or $\mathcal{D}_{tgt}^{query}$. By doing so, we can mimic real train-test domain shifts so that over many iterations, we can train a model to achieve better generalization on the target language's final test set.

Meta-Tasks Generation. HateMAML requires episodic training on meta-tasks containing support (meta-training) and query (meta-testing) sets. For each episode, three subsets of examples are sampled separately: the support set $\mathcal{S}$, the query set $\mathcal{Q}$, and the (virtual) domain query set $\mathcal{Q}'$.

$$
\begin{aligned}
&S \leftarrow S \cup \wedge(X, Y) \in \mathcal{D}, \\
&Q \leftarrow Q \cup \wedge(X, Y) S \in \mathcal{D}, and \\
&Q' \leftarrow Q' \cup \wedge(X, Y)(S \cup Q) \in \mathcal{D}.
\end{aligned}
\tag{3.4}
$$

We repeat this process multiple times. Suppose we have a total of D training samples in a given language and K samples used for support, and L for both query sets in each episode. In total, we will have $D/K + L$

episodes. Note that $\mathcal{Q}'$ is virtual and selected randomly.

(Virtual) Domain Query Loss. We add a query loss term in Equation 3.3 that accounts for domain generalization. The meta-update in Equation 3.3 is computed as follows:

$$\theta = \theta - \beta \nabla_\theta \sum_{i=1}^{m} \mathcal{L}_{\mathcal{T}_i}^{(\mathcal{Q}_i, \mathcal{Q}'_i)}(f_{\theta'_i}). \tag{3.5}$$

It is a variation of standard MAML training that addresses supervised training limitations such as domain shift in low-resource languages. The effectiveness of our proposed training method is demonstrated by comparison with MAML.

3.3.3 HateMAML Algorithm

HateMAML training requires a base classifier. A base classifier can be any multilingual encoder, such as mBERT, XLM-R, fine-tuned on a source language such as English. The choice between zero-shot and few-shot training is then made based on the availability of training data. The training choice is passed to Algorithm 1 along with the base model parameters. For the zero-shot scenario, an auxiliary language is selected for model training, and the resulting meta-model will be evaluated on the selected target language. Meta-learner parameters θ are initialized from a fine-tuned base model (*line 2*). Now, we sample a batch of pseudo-tasks $\mathcal{T}_i$. Each task contains a triplet $(\mathcal{S}, \mathcal{Q}, \mathcal{Q}\prime)$ representing the support, query, and domain query for model training, respectively. We take one inner loop gradient step using training loss on $\mathcal{S}$ (*line 10*) and adapt model parameters to θ'. Now, we evaluate task-specific training outcomes using $\mathcal{Q}$ and $\mathcal{Q}'$, and save it for outer loop update (*line 12*). Each task has its own task-adaptive model parameters θ'_i. At the end of iterations of all tasks, the meta-learner parameters θ are updated (*line 13*). One outer loop is completed at this point. If model training is not finished, lines 8 through 13 are repeated. After training completion, based on the earlier choice of zero-shot or few-shot, the resulting model's performance is evaluated on the target language's **final** test set.

3.3.4 Self-refinement and Zero-shot Learning

We now present a semi-supervised self-training procedure shown in Figure 3.1 (left) for hate speech classifier adaptation while no target or auxiliary language is available. Our proposed approach requires the availability of the source language. As before, a 'base model' is developed by fine-tuning source language training data. Now, the trained model can be used to make predictions on unlabeled samples from the target language. We want to use these predicted samples for supervised training. Directly including all the predicted samples in training may not result in a good classifier, as training samples are noisy and do not represent ground-truth labels. Samples with low prediction confidence are filtered out using a threshold value to keep it balanced. The *pseudo-labels* from the filtered results are used to generate a new training dataset. HateMAML few-shot training algorithm is used for model training. The meta-training needs a smaller sample size than fine-tuning, which reduces noise injection. After completing one full training on the pseudo-labeled training data, the

Algorithm 1: HateMAML

Training choice `C`, high-resource language `h`, set of low-resource languages `L`, Meta-learner `M`, fast lr α, meta lr β
Fine-tune θ on `h` (base model)
Initialize $\theta \leftarrow \theta$
if *C is zero-shot* **then**
    Select auxiliary from `L` to get $\mathcal{D} = (\mathcal{D}_{src}, \mathcal{D}_{aux})$
else
    Select target language get `L` to form $\mathcal{D} = (\mathcal{D}_{src}, \mathcal{D}_{tgt})$
while *not done* **do**
    Sample batch of tasks $\mathcal{T} = \{\mathcal{T}_1, \mathcal{T}_2.., \mathcal{T}_m\} \sim \mathcal{D}$
    for *all* $\mathcal{T}_i = (\mathcal{S}, \mathcal{Q}, \mathcal{Q}')$ *in* $\mathcal{T}$ **do**
        Compute $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(\mathtt{M}_\theta)$ on S
        Compute adapted parameters with gradient descent: $\theta^{'} = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(\mathtt{M}_\theta)$
        Compute $\mathcal{L}_{\mathcal{T}_i}(\mathtt{M}_{\theta'})$ using $\mathcal{Q}$, $\mathcal{Q}_d$ for outer loop meta-update
    Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_i \mathcal{L}_{\mathcal{T}_i}(\mathtt{M}_{\theta'})$
Perform test set evaluation on the target language using $\mathtt{M}_\theta$

Table 3.1: A summary of the multilingual hate speech datasets.

Dataset	Language	Size of training set	Size of test set	Domain
Founta-EN[124]	English (EN)	80000	—	Twitter stream
HatEval19 [22]	English (EN), Spanish (ES)	**EN**: 9000, **ES**: 4500	**EN**: 2971, **ES**: 1600	Immigrant, Woman
SemEval20 [28]	Arabic (AR), Danish (DA), Greek (EL), Turkish (TR)	**AR**: 8000, **DA**: 2961, **EL**: 8743, **TR**: 31756	**AR**: 200, **DA**: 329, **EL**: 1544, **TR**: 3528	Twitter, Facebook, Reddit, News
HASOC20 [29]	English (EN), Hindi (HI), German (DE)	**EN**: 3708, **HI**: 2963, **DE**: 2373	**EN**: 1628, **HI**: 1326, **DE**: 1052	Twitter stream
HaSpeeDe [145]	Italian (IT)	**IT**: 6839	$\mathbf{IT_{tweeter}}$: 1263, $\mathbf{IT_{news}}$: 500	Immigration, Roma, Muslims

new classifier is expected to have better target language knowledge. To make it work in practice requires self-refinement with multiple iterations. We repeat this procedure by treating the resulting model as the base for generating pseudo-labels again, replacing the old base model. The number of iterations needed for effective meta-training is explored empirically.

3.4 Experiments

We consider three training setups: (i) *zero-shot* with no fine-tuning in the target, (ii) *domain adaptation* in which we consider training on a set of auxiliary languages and test on a held-out language set, and (iii) *full* fine-tuning using the training samples from all languages. Our experimental setup also requires a *base model*, acquired from a pre-trained model, which is fine-tuned on the source (English) language.

Table 3.2: A summary of the multilingual hate speech datasets.

Dataset	Language	Size of training set	Domain
Founta-EN	English (EN)	80000	Twitter stream
HatEval19	English (EN), Spanish (ES)	**EN**: 9000, **ES**: 4500	Immigrant, Woman
SemEval20	Arabic (AR), Danish (DA), Greek (EL), Turkish (TR)	**AR**: 8000, **DA**: 2961, **EL**: 8743, **TR**: 31756	Twitter, Facebook, Reddit, News
HASOC20	English (EN), Hindi (HI), German (DE)	**EN**: 3708, **HI**: 2963, **DE**: 2373	Twitter stream
HaSpeeDe	Italian (IT)	**IT**: 6839	Immigration, Roma, Muslims

Table 3.3: List of languages and their ISO codes used in our experiments.

Language	ISO 639-1 code	Family
Arabic	AR	Afro-Asiatic
Danish	DA	IE: Germanic
German	DE	IE: Germanic
Greek	EL	IE: Greek
English	EN	IE: Germanic
Hindi	HI	IE: Indo-Aryan
Spanish	ES	IE: Italic
Turkish	TR	Turkic

Eight low-resource languages and one high-resource language (English) are used in our experimentation for the cross-linguistic analysis. The base model for the experiments on HASOC20 and HatEval19 is trained using the English samples from the training set. Since there are no English training data available, we use Founta-EN for the experiments on SemEval2020 and HaSpeedDe20.

The cross-domain experiments train of aggregated training data from a set of languages and test the trained model on target reserved language set. The languages in the selected languages families, namely EN, DA, DE, ES, and IT, are considered auxiliary and used in the training set. The trained model is evaluated on the held-out languages. We use training samples in increasing order to evaluate the model performance in the order of available data. Each language's varying training data size has three values (1024, 2048, and 4096). However, if a language does not contain the selected number of samples (e.g. DE has only 2373 training samples), the maximum of found training samples is used in training.

The meta-training samples are retrieved from the source languages' *validation sets* as well as the auxiliary

and target languages' *training sets*. The source languages' *training sets* are used to train the *base model*. Samples from each language are combined, and the resulting training set is used to generate meta-tasks. Task triplets are sampled with a specified number of shots, $K = L$, where $K = 32$.

We utilize two multilingual transformer-based encoders as base models: (i) **mBERT** [21] (bert-base-multilingual-uncased), and (ii) **XLM-R** [82] (xlm-roberta-base). For both models, the output from the *pooler* layer is fed to a 2-layer FFN classification head.

3.4.1 Evaluation

In this section, we outline the primary building blocks of the metrics used in the evaluation of classification models. We will define accuracy, precision, recall and F1 score metrics that are commonly used. For binary classification:

- A *true positive* (TP) result is one where the model correctly identified the positive class.
- Similar to a true positive, a *true negative* (TN) is a result where the model accurately predicted the negative class.
- A *false positive* (FP) is an outcome where the model incorrectly predicts the positive class.
- A *false negative* (FN) is a result when the model predicts the negative class incorrectly.

Precision is defined as follows:

$$Precision = \frac{TP}{TP + FP} \tag{3.6}$$

Recall is defined as follows:

$$Precision = \frac{TP}{TP + FN} \tag{3.7}$$

F1 score is a harmonic mean of precision and recall:

$$\text{F1 score} = 2 * \frac{precision * recall}{precision + recall} \tag{3.8}$$

Accuracy has the following definition:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} \tag{3.9}$$

We report the performance of classification models on the test set as the macro-averaged (unweighted mean) F1 score. To alleviate the randomness of the scores, the mean and standard deviation of five different runs are reported.

3.4.2 Baselines

We investigate the following baselines in our experiments:

- **Fine-tuning**: Standard LM fine-tuning is adapted as a baseline for both zero-shot and few-shot scenarios. Here, we fine-tune on **mBERT** and **XLM-R**.
- **X-MAML** [99]: Cross-lingual episodic meta-training using auxiliary language composition, including pairs. Meta-learner weights are initialized from the base model obtained from English fine-tuning. The meta-learned model is then evaluated on the test set in the target language. Our implementation only considers one auxiliary language.

3.4.3 Variations of HateMAML

Besides benchmarking the baselines, we formulate and evaluate different variations of HateMAML:

- **HateMAML$_{zero}$:** Simulated meta-training on a batch of tasks created from aggregated samples in source language 'validation set' and 'training set' from auxiliary language. It provides zero-shot as training choice in Algorithm 1 (subsection 3.3.3).
- **HateMAML$_{self\text{-}training}$:** Choose a target language for self-training on pseudo-labeled data. First, we generate pseudo-labels and keep only 300 samples. Next, HateMAML is applied in an iterative manner to the pseudo-labeled data from the target language as explained in subsection 3.3.4.
- **HateMAML$_{ft}$:** Meta-task is generated from a set of available languages and HateMAML is trained using the aggregated training samples. This is similar to fine-tuning on aggregated training examples from the selected language set.

3.4.4 Implementation Details

We implement our proposed model using the HuggingFace `transformers` library. The chosen pre-trained model is initialized from pre-trained weights provided in the `transformers` library.[2] The classification heads are implemented using the pytorch linear layers, initialized randomly.[3] We use the `learn2learn` library to implement the meta-learning features, such as first-order approximation.[4]

3.5 Results and analysis

3.5.1 Zero-shot Experiments

Experiments in the zero-shot setting can be found in Table 3.4. The mBERT initialization uses the ♣ sign, and ♠ refers to XLM-R initialization. The reported results of all the experiments are average values across five random runs. HateMAML$_{zero}$ outperforms baseline models including X-MAML with both initializations

[2]https://huggingface.co/transformers/
[3]https://pytorch.org/
[4]https://github.com/learnables/learn2learn

Table 3.4: Zero-shot evaluation on target low-resource languages in several multilingual hate speech datasets. Empty cells (−) indicate that meta-learning experiments are impossible due to the unavailability of the required auxiliary language. In each cell (x_y), x and y denote the mean and standard deviation of macro-F1 values across five runs by varying the random seed initialization.

Model	Aux lang	HatEval19	SemEval20				HaSpeedDe20	HASOC20		AVG
		ES	AR	DA	EL	TR	IT_{news} IT_{tweets}	HI	DE	
mBERT	-	$0.499_{0.025}$	$0.448_{0.000}$	$0.650_{0.000}$	$0.457_{0.000}$	$0.452_{0.010}$	$0.436_{0.023}$ $0.489_{0.25}$	$0.426_{0.000}$	$0.414_{0.000}$	$0.473_{0.007}$
♣$X-MAML$	✓	-	$0.562_{0.024}$	$0.654_{0.017}$	$0.501_{0.040}$	$0.585_{0.002}$	-	$0.455_{0.015}$	$0.629_{0.032}$	$0.564_{0.021}$
♣HateMAML$_{zero}$	✓	-	$0.583_{0.010}$	$0.668_{0.017}$	$0.532_{0.004}$	$0.601_{0.005}$	-	$0.448_{0.073}$	$0.745_{0.046}$	$0.592_{0.026}$
♣HateMAML$_{self_training}$	-	$\mathbf{0.525_{0.033}}$	$0.529_{0.019}$	$0.657_{0.046}$	$0.547_{0.021}$	$0.586_{0.002}$	$0.569_{0.031}$ $\mathbf{0.662_{0.056}}$	$0.509_{0.029}$	$0.572_{0.048}$	$0.562_{0.028}$
XLM-R	-	$0.421_{0.064}$	$0.524_{0.000}$	$0.683_{0.000}$	$0.529_{0.000}$	$0.478_{0.000}$	$0.457_{0.028}, 0.593_{0.039}$	$0.448_{0.000}$	$0.576_{0.000}$	$0.523_{0.014}$
♠$X-MAML$	✓	-	$0.647_{0.007}$	$0.699_{0.007}$	$0.604_{0.023}$	$0.527_{0.014}$	-	$0.473_{0.010}$	$0.672_{0.007}$	$0.604_{0.011}$
♠HateMAML$_{zero}$	✓	-	$0.651_{0.007}$	$0.735_{0.006}$	$0.619_{0.003}$	$0.580_{0.028}$	-	$0.497_{0.017}$	$0.650_{0.013}$	$0.622_{0.012}$
♠HateMAML$_{self_training}$	-	$\mathbf{0.436_{0.0846}}$	$0.621_{0.048}$	$0.688_{0.036}$	$0.635_{0.042}$	$0.600_{0.016}$	$0.492_{0.020}$ $0.637_{0.021}$	$0.598_{0.003}$	$0.661_{0.005}$	$0.591_{0.031}$

Table 3.5: Fine-tuning and meta-training evaluation on multilingual hate speech datasets. F1 score is reported. Here, first block means the pre-trained model is initialized with BERT and the second block refers to XLM-R initialization.

Model	data	HatEval19	SemEval20				HaSpeedDe20	HASOC20		AVG
		ES	AR	DA	EL	TR	IT_{news}, IT_{tweets}	HI	DE	
♣FT	1024	$0.663_{0.009}$	$0.721_{0.016}$	$0.712_{0.027}$	$0.700_{0.080}$	$0.610_{0.053}$	$0.629_{0.039}, 0.686_{0.009}$	$0.590_{0.029}$	$0.760_{0.017}$	$0.675_{0.064}$
♣FT	2048	$0.718_{0.024}$	$0.765_{0.020}$	$0.737_{0.028}$	$0.717_{0.041}$	$0.652_{0.050}$	$0.652_{0.037}, 0.708_{0.021}$	$0.609_{0.054}$	$0.796_{0.012}$	$0.706_{0.065}$
♣FT	4096	$0.717_{0.017}$	$0.785_{0.024}$	$0.747_{0.037}$	$0.722_{0.007}$	$0.670_{0.035}$	$0.680_{0.045}, 0.684_{0.068}$	$0.584_{0.071}$	$0.803_{0.004}$	$0.712_{0.077}$
♣HateMAML$_{ft}$	1024	$0.693_{0.015}$	$0.726_{0.025}$	$0.692_{0.018}$	$0.685_{0.044}$	$0.626_{0.018}$	$0.593_{0.026}, 0.702_{0.020}$	$0.583_{0.020}$	$0.743_{0.023}$	$0.671_{0.059}$
♣HateMAML$_{ft}$	2048	$0.735_{0.013}$	$0.779_{0.011}$	$0.716_{0.013}$	$0.732_{0.026}$	$0.665_{0.007}$	$0.670_{0.019}, 0.730_{0.011}$	$0.627_{0.006}$	$0.789_{0.013}$	$0.716_{0.052}$
♣HateMAML$_{ft}$	4096	$0.757_{0.018}$	$0.816_{0.012}$	$0.752_{0.010}$	$0.742_{0.022}$	$0.689_{0.012}$	$0.643_{0.042}, 0.741_{0.010}$	$0.627_{0.016}$	$0.793_{0.009}$	$0.729_{0.063}$
♠ FT	1024	$0.630_{0.063}$	$0.680_{0.106}$	$0.774_{0.028}$	$0.595_{0.064}$	$0.593_{0.077}$	$0.605_{0.081}, 0.643_{0.029}$	$0.621_{0.036}$	$0.735_{0.059}$	$0.653_{0.085}$
♠ FT	2048	$0.684_{0.019}$	$0.730_{0.043}$	$0.782_{0.022}$	$0.673_{0.061}$	$0.671_{0.032}$	$0.683_{0.039}, 0.629_{0.058}$	$0.641_{0.043}$	$0.800_{0.006}$	$0.699_{0.067}$
♠ FT	4096	$0.704_{0.015}$	$0.806_{0.021}$	$0.774_{0.008}$	$0.705_{0.021}$	$0.684_{0.013}$	$0.651_{0.033}, 0.679_{0.023}$	$0.632_{0.022}$	$0.806_{0.011}$	$0.716_{0.064}$
♠HateMAML$_{ft}$	1024	$0.728_{0.004}$	$0.781_{0.016}$	$0.736_{0.001}$	$0.703_{0.004}$	$0.695_{0.001}$	$0.608_{0.015}, 0.741_{0.018}$	$0.664_{0.010}$	$0.789_{0.000}$	$0.716_{0.056}$
♠HateMAML$_{ft}$	2048	$0.742_{0.012}$	$0.803_{0.002}$	$0.733_{0.003}$	$0.692_{0.018}$	$0.695_{0.024}$	$0.652_{0.015}, 0.748_{0.007}$	$0.672_{0.005}$	$0.823_{0.009}$	$0.729_{0.057}$
♠HateMAML$_{ft}$	4096	$0.755_{0.005}$	$0.832_{0.010}$	$0.736_{0.022}$	$0.711_{0.019}$	$0.706_{0.033}$	$0.694_{0.036}, 0.738_{0.022}$	$0.657_{0.016}$	$0.812_{0.008}$	$0.738_{0.056}$

(♣ and ♠). We achieve the best overall score with an average of 11% over mBERT, 7% over XLM-R, 7% over ♣ X-MAML. Baseline models, mBERT and XLM-R, have poor *zero-shot* performance. The use of auxiliary language to obtain the best meta-learner produces rewarding outcomes for all the experiments. Even though X-MAML uses a similar cross-lingual, auxiliary language-based, meta-training setting to ours, HateMAML gives superior performance, likely due to the domain-adaptive training in HateMAML. To summarize, in most cases, HateMAML's auxiliary to target transfer improves the results of zero-shot learning substantially compared to the standard fine-tuning baselines.

The XLM-R-initialized models perform better than mBERT in zero-shot prediction. Even the best performing ♣HateMAML model has a small gap between XLM-R zero-shot performance. One explanation is that XLM-R is trained on a relatively large corpus and a deeper transformer architecture, giving it a

Table 3.6: Zero-shot experiments and results using auxiliary languages on SemEval2020 dataset. All the results are produced from XLM-R initialization and average over 5 runs.

Model	Auxiliary lang	Target lang			
		AR	TR	DA	EL
HateMAML$_{zero}$	AR	-	$0.553_{0.026}$	$0.718_{0.011}$	$0.591_{0.008}$
HateMAML$_{zero}$	TR	$0.651_{0.007}$	-	$0.738_{0.012}$	$0.619_{0.003}$
HateMAML$_{zero}$	DA	$0.628_{0.022}$	$0.580_{0.028}$	-	$0.553_{0.026}$
HateMAML$_{zero}$	EL	$0.530_{0.017}$	$0.494_{0.021}$	$0.657_{0.044}$	-

strong pre-training benefit for zero-shot transfer learning. We observe similar trends in performance with HateMAML when initialized with XLM-R base models across all the datasets. For the SemEval20 Danish (DA) test set, it is noted that the XLM-R model performs the best compared to other meta-learning models. One reason could be that this dataset was created from Facebook, Reddit, and news sources, unlike the other datasets that have been created from Twitter.

Language similarity and cross-lingual transfer. We analyze the support that each target language receives from different auxiliary languages. The SemEval20 dataset, with the largest number of possible (auxiliary, target) language pairs, shows the benefit of meta-training on auxiliary language for both X-MAML and HateMAML$_{zero}$ models. Meta-training helps in increasing the zero-shot performance for SemEval20 from 0.473 avg. F1 (mBERT) to 0.592 F1 (♣HateMAML). X-MAML also shows substantial improvement for both initializations. We notice that SemEval20 Turkish (TR) tends to be the best auxiliary language dataset in almost all cases for both base models (see Table 3.6). For HASOC20 experiments, we notice small gain in F1 score while training on auxiliary samples. The languages – German (DE) and Hindi (HI), do not seem to be a good mix, coming from two distant language families (details of family languages is provided in Table 3.3).

Self-training improves zero-shot performance. For datasets that have no auxiliary language, i.e., HatEval19 and HaSpeedDe20, self-training HateMAML is a convenient algorithm to maximize the performance. It mitigates the need for auxiliary language or a labeled target language training set. We find that HateMAML$_{self-training}$ produces comparable performance to meta-training in auxiliary HateMAML$_{zero}$. The gain is achieved from meta-training on pseudo-labels of the target using multiple iterations of self-refinement. Self-training also uses the source language validation set. The intuition is to have some gold-labeled samples to balance between noise and ground truth. We observe a slight improvement given the source ground-truth labels in training.

3.5.2 Domain Adaptation Experiments

Table 3.7 reports the results of domain adaptation experiments. We first select two language families for training: Germanic and Romance. It can be seen that HateMAML produces superior results on average

compared to fine-tuning. We find that domain-adaptive meta-training in HateMAML$_{ft}$ (♣ or ♠) has a consistently better F1 score than fine-tuning. We achieve an overall improvement of 3% over mBERT, 24% over XLM-R while training 2048 samples per language. Both baseline models face a performance drop in held-out languages, in which XLM-R faces a significant performance drop. To summarize, standard fine-tuning does not generalize well during training on the held-out target languages. On the other hand, HateMAML's domain-adaptive training helps retain consistent performance on both auxiliary and target languages.

Table 3.7: Domain Adaptation experiments on both fine-tuning and meta-training. We use languages = EN,ES,IT,DA,DE for training and evaluate on all eight languages. Here, the first block means the pre-trained model is initialized with BERT and the second block refers to XLM-R initialization.

Model	data	HatEval19	SemEval20				HaSpeedDe20	HASOC20		AVG
		ES	AR	DA	EL	TR	IT_{news}, IT_{tweets}	HI	DE	
♣*FT*	1024	$0.684_{0.022}$	$0.490_{0.035}$	$0.711_{0.020}$	$0.533_{0.012}$	$0.504_{0.027}$	$0.666_{0.035}, 0.703_{0.014}$	$0.537_{0.023}$	$0.735_{0.046}$	$0.618_{0.098}$
♣*FT*	2048	$0.727_{0.049}$	$0.517_{0.041}$	$0.726_{0.018}$	$0.515_{0.022}$	$0.541_{0.024}$	$0.667_{0.035}, 0.695_{0.031}$	$0.464_{0.038}$	$0.781_{0.010}$	$0.626_{0.115}$
♣*FT*	4096	$0.753_{0.024}$	$0.493_{0.037}$	$0.753_{0.021}$	$0.510_{0.035}$	$0.546_{0.030}$	$0.658_{0.041}, 0.736_{0.010}$	$0.462_{0.041}$	$0.805_{0.005}$	$0.635_{0.130}$
♣HateMAML$_{ft}$	1024	$0.709_{0.006}$	$0.495_{0.059}$	$0.723_{0.022}$	$0.545_{0.015}$	$0.539_{0.032}$	$0.607_{0.058}, 0.711_{0.004}$	$0.498_{0.030}$	$0.766_{0.013}$	$0.622_{0.106}$
♣HateMAML$_{ft}$	2048	$0.737_{0.019}$	$0.546_{0.009}$	$0.724_{0.019}$	$0.559_{0.017}$	$0.556_{0.014}$	$0.688_{0.031}, 0.722_{0.026}$	$0.474_{0.021}$	$0.795_{0.011}$	$0.645_{0.108}$
♣HateMAML$_{ft}$	4096	$0.746_{0.018}$	$0.556_{0.020}$	$0.748_{0.025}$	$0.559_{0.008}$	$0.567_{0.009}$	$0.695_{0.017}, 0.702_{0.035}$	$0.475_{0.009}$	$0.802_{0.015}$	$0.650_{0.109}$
♠ FT	1024	$0.521_{0.154}$	$0.517_{0.088}$	$0.574_{0.111}$	$0.484_{0.047}$	$0.517_{0.079}$	$0.521_{0.135}, 0.492_{0.164}$	$0.492_{0.072}$	$0.580_{0.162}$	$0.522_{0.113}$
♠ FT	2048	$0.561_{0.180}$	$0.482_{0.045}$	$0.623_{0.149}$	$0.507_{0.047}$	$0.527_{0.077}$	$0.540_{0.145}, 0.507_{0.188}$	$0.438_{0.025}$	$0.638_{0.207}$	$0.536_{0.137}$
♠ FT	4096	$0.579_{0.191}$	$0.468_{0.024}$	$0.626_{0.146}$	$0.513_{0.060}$	$0.500_{0.060}$	$0.570_{0.167}, 0.536_{0.182}$	$0.431_{0.016}$	$0.646_{0.211}$	$0.541_{0.142}$
♠HateMAML$_{ft}$	1024	$0.725_{0.011}$	$0.608_{0.014}$	$0.716_{0.030}$	$0.582_{0.027}$	$0.624_{0.004}$	$0.664_{0.042}, 0.732_{0.013}$	$0.552_{0.018}$	$0.769_{0.008}$	$0.663_{0.075}$
♠HateMAML$_{ft}$	2048	$0.726_{0.022}$	$0.637_{0.035}$	$0.749_{0.019}$	$0.609_{0.023}$	$0.614_{0.029}$	$0.663_{0.031}, 0.681_{0.046}$	$0.510_{0.032}$	$0.806_{0.010}$	$0.666_{0.088}$
♠HateMAML$_{ft}$	4096	$0.738_{0.021}$	$0.562_{0.061}$	$0.745_{0.043}$	$0.595_{0.019}$	$0.586_{0.017}$	$0.702_{0.009}, 0.700_{0.034}$	$0.481_{0.025}$	$0.805_{0.007}$	$0.657_{0.104}$

3.5.3 Full Fine-tuning vs Meta-training

To make one model support all languages, we evaluate a meta-training strategy while training on data available in eight languages. Meta-tasks are created by gathering support and query sets from eight languages. Aggregated training data is also used for fine-tuning. By varying the size of the training data, we evaluate model performance under increasing data availability. In Table 3.5, we show how HateMAML$_{ft}$ performs in comparison to mBERT and XLM-R fine-tuning. Increasing the quantity of training data across all languages generally shows an upward trend. As expected, the meta-learned initialized HateMAML$_{ft}$ models perform better than the standard fine-tuning. HateMAML$_{ft}$ outperforms on average for increasing the order of available data. We observe that HateMAML$_{ft}$ gains in F1 score for all languages – 2.4% and 3% improvement over baselines mBERT and XLM-R using 4096 samples. Increasing the training data, if available, will help performance even more. This suggests that meta-training can be an alternative to standard fine-tuning for hate speech detection.

3.6 Major takeaways

Our experiments show that HateMAML is able to perform well in both zero-shot and few-shot hate speech detection by improving cross-lingual transfer in target languages. HateMAML can be trained with or without labeled data. This model-agnostic approach supports multilingual encoders with shared embeddings such as mBERT and XLM-R as base learners. The proposed zero-shot self-refining technique adapts a base learner to be an effective predictor in the target, reducing the need for ground-truth labels in fine-tuning. To further improve cross-lingual transfer, we use a few labeled samples in the target language (few-shot training), which helps to adapt fast and boosts fine-tuning performance. Additionally, our method benefits from task-specific learner evaluation on a (virtual) domain test (query) set at each episode that mimics target domain evaluation during cross-lingual meta-training. The intuition of the domain query set is to achieve training generalization in unseen target languages and domains. We also found that meta-learner adaptation effectively supports all available languages using a single predictor model, making it highly suitable for detecting hate speech in multiple languages and domains.

4 Multilingual Rumour Detection

4.1 Introduction

Motivation. The rapid spread of unverified information (i.e., rumours) on social media creates discord in society and reduces trust in the information shared online. To combat this fast-growing problem, debunking services such as Snopes have attempted to debunk rumours by manually fact-checking the information.[1] Manual fact-checking however is inefficient in coping with the massive and viral spread of rumours disseminated on social media platforms. Researchers have hence proposed machine learning solutions to automatically detect and debunk rumours spread on social media platforms.

Towards automated solutions, researchers proposed the rumour verification task, which aims to determine whether the source claim in a conversation thread is a *false rumour*, *true rumour*, or *unverified rumour* [146]. Rumour verification datasets were also created and made available to support such research. Examples of commonly-used rumour verification datasets include PHEME [147] and Twitter15/16 [12], which contain rumour and non-rumour Twitter conversation threads and tweets. These datasets have encouraged the development of many rumour verification solutions [25, 146, 148, 149].

In spite of this, the existing rumour verification datasets are mostly in English and collected based on the US context. This neglects the bulk of the rumours disseminated in other non-US communities and languages. For instance, Figure 4.1 illustrates examples of two false rumour threads. The two threads are discussing the same topic (i.e., cure for COVID-19) but in different languages. Existing rumour verification solutions may not be able detect the rumour in thread (B) as they are not designed to detect rumours in multilingual settings.

Research Objectives. We aim to address the existing research gaps by introducing the cross-lingual and multilingual rumour verification tasks. We propose the **Mu**ltilingual **S**ource **C**o-**A**ttention **T**ransformer (MUSCAT) model, which builds on a multilingual pre-trained language model (MPLM) to perform multilingual rumour verification. Specifically, MUSCAT pivots the source claims in multilingual conversation threads with co-attention transformers to attend the source claim and replies in a sub-thread. Subsequently, the hidden states of all sub-threads are aggregated with a global context encoder before feeding the final thread representation into a Softmax classifier for rumour verification. To support experimental evaluation, we collected and annotated a multilingual rumour verification dataset in the Southeast Asian context. Additionally,

[1] https://www.snopes.com/

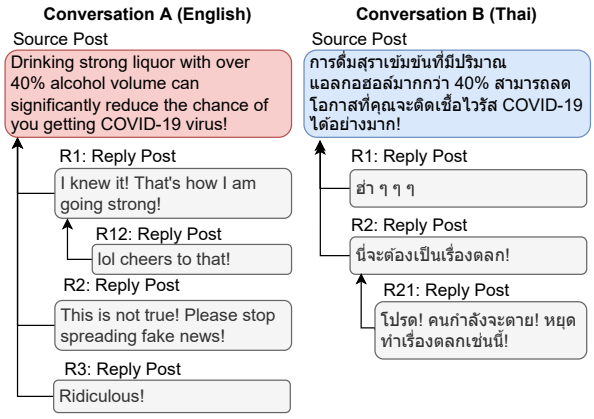


Figure 4.1: Examples of two rumour conversations in English (left) and Thai (right). Both conversations are spreading falsehoods on COVID-19 cures.

we construct two other multilingual rumour verification datasets by translating the PHEME and Twitter16 datasets into four Southeast Asia low-resource languages (i.e., Bahasa Indonesia, Bahasa Melayu, Thai, and Vietnamese).

Contributions. Our major contributions are summarized below:

- We introduce cross-lingual and multilingual rumour verification tasks and propose MUSCAT, that performs rumour verification in multilingual settings. To the best of our knowledge, this is the first work that proposes performing rumour verification in multilingual settings.
- We construct three multilingual rumour verification datasets to support our experimental evaluations and encourage future research on multilingual rumour verification.
- We conducted extensive experiments to evaluate our proposed model and discussed the interesting findings and challenges of performing multilingual rumour verification.

4.2 Related Work

4.2.1 Rumour Verification

The existing studies on rumour verification can broadly be categorized into two groups. The first line of work explored training classifiers using a myriad of handcrafted features such as post content, user profile, and propagation patterns [13, 150, 151, 152, 153]. The second line of work proposed using deep learning to automatically capture useful latent features from the post content and propagation patterns [25, 148, 149, 154, 155, 156, 157, 158]. More recently, researchers have also explored adopting multi-task learning to perform rumour verification along with stance classification [159, 160, 161]. For instance,Yu et. al., [161] proposes a coupled hierarchical transformers (CHT) for multi-task learning that exploits inter-task dependencies of

stance and rumour veracity. Most of the above-proposed rumour verification models were evaluated in a monolingual setting. Specifically, existing rumour verification models are evaluated on the English rumour datasets such as the Twitter15/16 [12] and PHEME datasets [147]. This neglects the bulk of the non-English rumours disseminated on social media. We aim to fill this research gap by proposing datasets and a multilingual Co-Attention Transformer model to support cross-lingual and multilingual rumour verification.

4.2.2 Multilingual Pre-Trained Language Models

Transformer-based large multilingual pre-trained language models (PLMs) such as mBERT [21] and XLM-R [82] contain rich contextualized representation from more than 100 languages and can be fine-tuned for many natural language processing and understanding tasks. The multilingual PLMs are especially helpful in low-resource language settings where labeled data is limited. For instance, Wu *et al.* [27] have explored leveraging multilingual PLMs to transfer knowledge from high-resource language to low-resource languages. Our study aims to leverage the multilingual PLMs to perform rumour verification in cross-lingual and multilingual settings. For instance, in the cross-lingual setting, we will explore fine-tuning the multilingual PLMs with an English rumour dataset for zero-shot prediction in the Southeast Asia languages rumour test sets.

4.2.3 Handling Long Sequences and Hierarchical Transformers

A common approach to process conversation in sequential modeling is to flatten it into a long document. However, processing long documents using neural networks is a challenging task. For instance, RNN suffers from the vanishing gradient problem [32, 162]. Furthermore, one of the major limitations of PLM encoders is that they are unable to accept input longer than a few hundred words. A simple solution to this problem is to split the long conversation document into segments with overlap [163]. Each chunk is first processed using the same encoder and then propagated to an MLP or recurrent layer before performing the final classification with the Softmax layer. Yu *et al.* [161] argued that concatenating those hidden vectors on top of a softmax classifier might not be able to capture the global context of the long document. Instead, the researchers adopted the Hierarchical Transformers [164], where another transformer layer is used to capture the interaction between segments. The proposed MUSCAT model is inspired by this architecture, where we adopted Longformer [165] as the global encoder to encode the interaction between segments.

4.3 Task Formulation

Given a text corpus, we denote a set of conversation threads as $\mathbb{D} = C_1, C_2, \cdots, C_{|\mathbb{D}|}$. Each thread C_i is then assumed to consist of a post with the source claim S^0 and a sequence of reply posts sorted in chronological order, denoted by $R^1, R^2, \cdots, R^N$. We assume that each C_i is associated with a veracity label y_i, which belongs to one of the four classes, namely *false rumour*, *true rumour*, *unverified rumour*, and *non-rumour*. The goal of the rumour verification task is to learn a classification function $f : C_i \rightarrow y_i$.

As we are interested in investigating rumour verification in multilingual settings, we further define three task settings:

- **Monolingual**: The rumour verification models are trained using conversation threads from a specific language l_s. The trained models are tested on a set of conversation threads in language l_s. Note that this is the commonly adopted setting in existing rumour verification studies.

- **Cross-lingual**: The rumour verification models are trained using conversation threads from a specific language l_s. The trained models are tested on a set of conversation threads in another language l_t. The goal of this setting is to examine the models' ability to transfer knowledge learned from conversation threads in the source language (i.e., typically, high-resource language such as English) and verify rumours in target languages (i.e., low resource languages).

- **Multilingual**: The rumour verification models are trained using conversation threads from a set of languages L. The trained models are tested on a set of conversation threads in the same set of languages L. This setting aims to examine the models' generalizability to adapt and verify rumours from multiple languages.

4.4 Methodology

Figure 4.2 illustrates the overall framework of the proposed Multilingual Source Co-Attention Transformer (MUSCAT) model. The model has three main components: (i) A local context encoder (LCE) that encodes sub-threads in a conversation. (ii) A global context encoder (GCE) that encodes the sub-threads local representations. (iii) the output layer classifies the conversation thread based on the aggregated global representation.

Motivation. Our goal is to design a multilingual model that can share information across languages during training and improve rumour verification performance in multilingual settings. Recent work has reported that fine-tuning multilingual pre-trained transformers could achieve promising performance for low-resource NLP tasks [27]. Therefore, we adapted mBERT [21] as the LCE in the MUSCAT model. Ideally, we would directly fine-tune mBERT for the rumour verification task. However, mBERT has a limitation on its input size and is unable to handle long documents. Hence, we adopt a hierarchical approach to break the long conversation thread into shorter sub-threads and encode each sub-thread using mBERT before aggregating the sub-thread representation using Longformer as the GCE. Nevertheless, this approach also has its limitation: due to the chunking process, the source post information is no longer available for some of the sub-threads. We propose a source co-attention layer that pivots the source post and exploits inter-dependency between source and corresponding replies in the LCE to address this limitation. The details of the working mechanism of MUSCAT's LCE and GCE will be discussed in the subsequent sections.

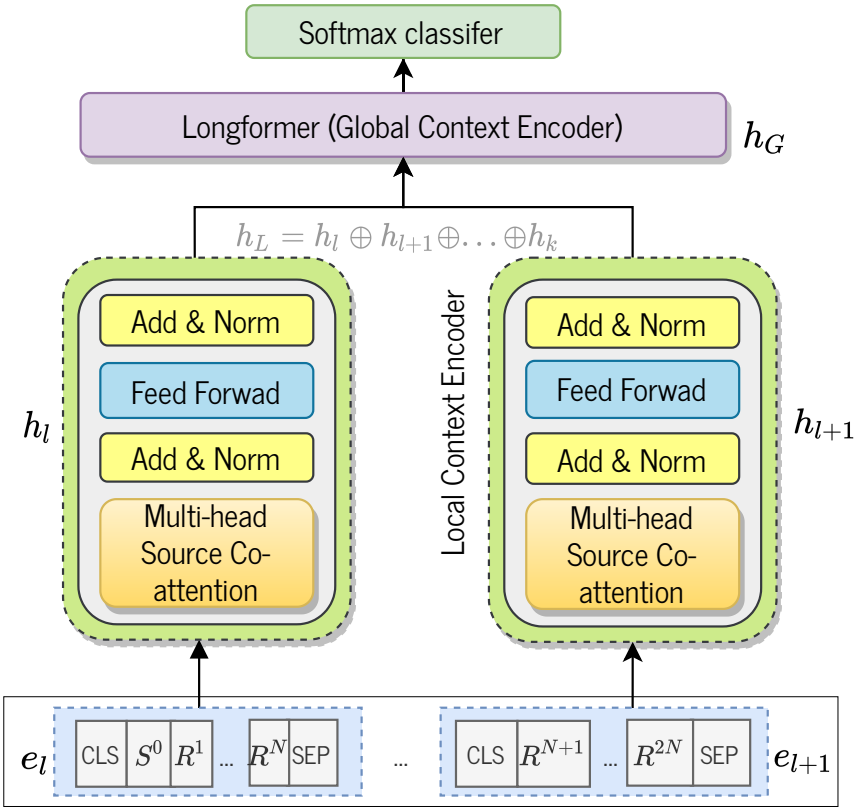


Figure 4.2: An overview of the MUSCAT model. A conversation comprises k sub-threads ($k = 2$ is shown here). Each sub-thread is input to a local context encoder (LCE) with the Multi-head Source Co-Attention layer. A Longformer is adopted as the global context encoder to attend and aggregate the local representations. Finally, the aggregated global representation is fed into a Softmax classifier.

4.4.1 Input Representation

Given a conversation thread C_i, we first arrange the posts in C_i in chronological order and then flatten the thread into a long sequence. Next, we split the flattened sequence into multiple sub-threads with an equal number of posts. In practice, we first tokenize the flattened sequence using the mBERT tokenizer and split the flattened sequence into three equal-length sub-threads. We insert two special tokens, i.e., [CLS] and [SEP], to each sub-thread's beginning and end. The mBERT has 12 layers, and the `CLS` token is used to extract final representations from the last layer of mBERT. Formally, let $C_i = (S^0, R^1, \cdots, R^N)$ denote the flattened thread, where S^0 is the source post, and R^j refers to the j-th reply post. As shown in the bottom of Figure 4.2, we assume that C_i is decomposed into k sub-threads.

4.4.2 Local Context Encoder

Next, we employed mBERT as the local context encoder (LCE) to separately process the k sub-threads' local interactions between adjacent posts within each sub-thread. We also postulate that the thread's source post

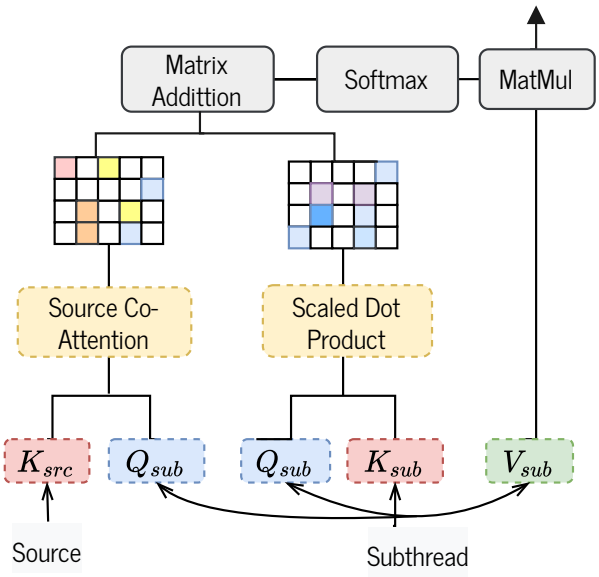


Figure 4.3: Source Co-Attention Module.

contains information that influences rumour verification. Therefore, we modify the multi-head self-attention mechanism in the mBERT to capture the inter-dependency of the source post and the corresponding replies in the sub-threads. We call this modified attention mechanism the *source co-attention* module (illustrated in Figure 4.3). The source co-attention module has two components: (i) standard self-attention and (ii) source-claim attends replies in the sub-thread.

In standard self-attention[3], we first transform input sequence into linear projections (Q), (K) and (V) of dimension d. The Q, K, V attention score is computed as follows:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V. \tag{4.1}$$

The above attention is carried out in multiple heads such that it allows the model to jointly attend to information from different representation sub-spaces at different positions.

$$MHA(Q, K, V) = [head_i, \cdots, head_n] \text{ where } head_i = Attention(Q_i, K_i, V_i). \tag{4.2}$$

In our modified source co-attention module, we introduce an additional projection K_{src} for source post representations along with the standard projections of a given sub-thread $Q_{sub}, K_{sub}, V_{sub}$. Therefore, we first compute the attention score on standard projections using equation 4.2. Next, we compute the source co-attention by replacing K_{sub} with K_{src}. The updated attention score:

$$MHA(Q_{sub}, K_{sub}, V_{sub}, K_{src}) = softmax(\frac{Q_{sub}K_{sub}^T}{\sqrt{d}} + \frac{Q_{sub}K_{src}^T}{\sqrt{d}}) * V_{sub}. \tag{4.3}$$

Finally, we apply layer normalization and feed-forward layer similar to the multi-head self-attention. The modified mBERT encodes each input sub-thread and outputs hidden vectors.

$$h_l = Encoder_L(e_l), l = 1, 2, \cdots, k, \tag{4.4}$$

where e_l is the input embeddings for the l-th sub-thread and h_l is the resulting hidden vector.

4.4.3 Global Context Encoder

Longformer is shown to be able to process long sequences efficiently [165]. Therefore, we adopt Longformer as the global context encoder (GCE) to encode the learned sub-threads representations. Specifically, we first concatenate the hidden states from all sub-threads $h_L = h_1 \oplus h_2 \oplus \cdots h_k$. Next, we pass h_L to our GCE module. The Longformer layer applies a set of attention patterns on h_L and outputs global representations vector h_G:

$$h_G = Encoder_G(h_L). \tag{4.5}$$

Output Layer. Finally, h_G is input to a feed-forward layer (FFN), and its output is fed into a SoftMax classifier to predict the conversation thread's rumour veracity.

4.5 Dataset Construction

As this is the first multilingual rumour verification study, we had to construct suitable datasets for experimental evaluation. We proposed two approaches to construct multilingual rumour datasets: (i) Translating existing rumour datasets, and (ii) collecting new multilingual rumour datasets.

4.5.1 Translating Existing Rumour Datasets

The PHEME [147] and Twitter15/16 [12] datasets are commonly used in existing rumour verification studies. Both datasets contain English tweets from Twitter. Table 4.1 provides a statistical summary of the two datasets. Our goal was to translate these two datasets into various Southeast Asia languages for rumour verification in multilingual settings.

As the two datasets only provide the tweet ids of the posts, we first utilize the Twitter API to retrieve the content of the posts. Note that the number of tweets retrieved is less than the reported numbers in the originating papers, as some of the tweets have either been removed by Twitter due to content policy violations or have since been deleted by its users. Next, we apply the Google Cloud Translation API to translate all the tweets in the two datasets into four Southeast Asia languages: Bahasa Indonesia, Bahasa Melayu, Vietnamese, and Thai. [2] The resulting dataset for each language has the same number of tweets as the original dataset as was recently retrieved. For instance, the Thai translated version of PHEME will have 105,354 post in Thai language. The PHEME and Twitter16 datasets, together with their translated versions are used in our experimental evaluation.

[2] `https://cloud.google.com/translate`

Table 4.1: Statistical summary of PHEME and Twitter16.

	PHEME	Twitter16
# of threads	6,425	817
# of posts	105,354	16,259
Avg. of posts/thread	16.40	19.90

4.5.2 Collecting Multilingual Rumour Dataset

Besides translating existing rumour datasets, we also collected and constructed SEAR (**S**outh**E**ast **A**sia **R**umours), a multilingual rumour dataset with English and Bahasa Indonesia tweets. Broadly, SEAR contains tweets that are related to fact-checked articles. The intuition is that the fact-checked articles are usually controversial, and spreading the information on these articles on social media could be seen as propagating rumours.

Collection. Our multilinugal data collection procedure is similar to previous works in rumour detection [150, 147]. We first gathered fact-checked articles from credible fact-checking websites, namely Black Dot Singapore for English articles in Singapore's context, and Cekfakta for Bahasa Indonesia articles in Indonesia's context.[3][4] we employed KeyBERT to extract the keywords from the article title and queried the Twitter API using the extracted keywords to retrieve relevant tweets.[5] As we are interested in performing rumour verification for conversation threads, we filtered and retained only tweets that are part of a conversation thread. A total of 1043 threads in Bahasa Indonesia and 560 threads of conversation in English have been gathered primarily. After annotation and quality-checking, there are 278 threads in English and 798 threads in Bahasa Indonesia.

Annotation. Next, we annotated the veracity of the collected conversation threads. Similar to existing rumour datasets, we annotated the conversation threads' source post (i.e., root tweet). Two annotators were recruited for this task. The annotators were undergrad students and we recruited them as intern for this project. The annotators first underwent a process of familiarization where they annotated a sample of 100 source tweets using an established codebook that follows the veracity labels used in the PHEME dataset [147]. Specifically, the annotators are presented with the content of the source tweets, along with the title and first paragraph of the corresponding fact-checked articles as contextual knowledge. Since the annotators are fluent in English, we translated the Bahasa Indonesia tweets into English to facilitate the annotation. After the familiarization exercise, the annotators discussed and aligned their understanding of the veracity labels.

Table 4.2 shows the statistical summary of the annotated SEAR dataset. Overall, we computed agreement using Cohen's Kappa and achieved 0.967 for English and 0.927 for Bahasa Indonesia. Both achieved almost

[3] `https://blackdotresearch.sg/factcheck/`
[4] `https://cekfakta.com/`
[5] `https://maartengr.github.io/KeyBERT/`

Table 4.2: Statistical summary of SEAR dataset. EN refers to English and ID refers to Bahasa Indonesia

	EN	ID
# of threads	278	798
# of posts	10,714	5,227
# of true rumour	58	495
# of false rumour	76	116
# of unverified rumour	26	15
# of non-rumour	119	189

perfect agreement, indicating a high degree of agreement in our annotations [166].

4.6 Experiments

4.6.1 Baselines

We benchmark MUSCAT using the following baselines:

- Support Vector Machines (SVM) classifier with radial basis function (rbf) kernels trained using top 5000 tf-idf features.
- Long-short term memory networks (LSTM) [32] with FastText pre-trained word embeddings for each corresponding language.[6]
- **TD-RvNN** [25] that utilize a top-down recursive neural networks (RvNN) to preserve tree-structure while processing a conversation. We use n-gram counts and top 5000 features to match the original experiment setting.
- Bidirectional graph convolutional neural networks (BiGCN) [154], which utilizes a Graph Convolutional Network (GCN) [167] with both top-down and bottom-up directed graph.
- **mBERT** [21], which is is a pre-trained model of 100 languages.
- **XLM-R** [82], which is a transformer base language model trained on large multilingual corpus and deep transformers layers.

We also adapted the single-task setting **CHT** [161] for rumour verification. In CHT, a conversation thread is processed as a chunk of sub-threads using a pre-trained BERT model. We replace the BERT model in CHT with mBERT for multilingual settings. Note that for tree-structured baselines such as TD-RvNN and

[6] `https://fasttext.cc/`

BiGCN, the models accept the conversation threads' propagation trees and post content as input. For the other sequence modeling baselines, we flatten the conversation threads in chronological order and concatenate the tweets in each thread into a long sequence as models' input.

4.6.2 Experimental Settings

Monolingual. We train and test on a specific language rumour dataset. For example, we train a model on the training set of the Vietnamese PHEME dataset and evaluate the model on the test set of the same Vietnamese PHEME dataset.

Cross-Lingual. We train models on a specific language rumour dataset and test the model on a rumour dataset of another language. For example, we train a model on the training set of the Vietnamese PHEME dataset and evaluate the model on the test set of the Thai PHEME dataset. Note that the train and test sets are carefully split to ensure no overlap or data leakage regardless of the languages.

Multilingual. We train and test models on the combined rumour datasets of multiple languages. For example, we train a model on the combined training set of PHEME datasets of all languages (i.e., English, Bahasa Indonesia, Bahasa Melayu, Vietnamese, and Thai) and evaluate on the test sets of PHEME datasets in various languages.

4.6.3 Evaluation Settings

In this section, we describe the key components of the metrics used in the evaluation of classification models. We will define the frequently used metrics for F1 score, recall, accuracy, and precision. For binary classification:

- A *true positive* (TP) result is one where the model correctly identified the positive class.
- Similar to a true positive, a *true negative* (TN) is a result where the model accurately predicted the negative class.
- A *false positive* (FP) is an outcome where the model incorrectly predicts the positive class.
- A *false negative* (FN) is a result when the model predicts the negative class incorrectly.

Precision is defined as follows:

$$Precision = \frac{TP}{TP + FP} \tag{4.6}$$

Recall is defined as follows:

$$Precision = \frac{TP}{TP + FN} \tag{4.7}$$

F1 score is a harmonic mean of precision and recall:

$$\text{F1 score} = 2 * \frac{precision * recall}{precision + recall} \tag{4.8}$$

Accuracy has the following definition:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} \tag{4.9}$$

K-fold validation. For Twitter16 and SEAR datasets, we perform 5-fold cross-validation (CV) and report the average accuracy and F1 scores. Since it is multi-class classification problem, we take the average of the F1 score of each class and report the mean.

Leave-one-out. As the PHEME dataset is organized by events, we adopt the evaluation approach proposed in [160] to perform leave-one-out CV at the event level. Specifically, we train models on conversation threads from eight events and test on the treads on the remaining event. Finally, we report the average accuracy and F1 scores. Note that this is a more challenging setting as it evaluates the models' ability to perform domain adaptation when they verify rumours for an unseen event.

4.6.4 Implementation Details

The TD-RvNN, BiGCN, and CHT models are reproduced using the available codes published in the original papers. We fine-tune mBERT and XLM-R from Huggingface transformers checkpoints.[7] The baseline models' hyperparameters are empirically optimized for the various datasets and experimental settings. For MUSCAT, we initialize most of the weights in LCE using `bert-base-multilingual-case` pre-trained checkpoint. The modified attention in the source co-attention module and LongFormer weights are learned. We empirically set the number of input sub-threads to 3. Adam optimizer with learning rate scheduler and warmup steps is used for model training. A fixed learning rate of 3e-5 is used in fine-tuning and batch size is set to 8.

4.7 Results & Analysis

Monolingual. We present the monolingual rumour verification experimental results for Twitter16, PHEME, and SEAR datasets in Table 4.3 and 4.4 respectively. MUSCAT is observed to outperform the state-of-the-art baselines in the three datasets. Specifically, MUSCAT has outperformed the best baseline by 4%, 4%, and 10% F1 scores on the Twitter16, PHEME, and SEAR datasets, respectively. We noted that all models performed worse for the PHEME experiments as it is a more challenging setting; the leave-one-out event evaluation setting will require the models to perform rumour verification in an unseen domain (i.e., event). Interestingly, we observe that MUSCAT and CHT outperformed the mBERT and XLM-R, supporting that the hierarchical approach is able to overcome the limitation of PLMs in handling long documents. MUSCAT has also outperformed CHT, suggesting the usefulness of the source co-attention module in capturing important signals in the source posts.

Multilingual. We present the multilingual rumour verification experimental results for Twitter16, PHEME, and SEAR datasets in Table 4.5 and 4.6. We have only benchmarked the performance of PLM-based

[7] `https://github.com/huggingface/transformers`

Table 4.3: Monolingual results on Twitter16 and PHEME. We report both accuracy and F1 score. EN - English, ID - Bahasa Indonesia, VI - Vietnamese, MS - Bahasa Melayu, and TH - Thai

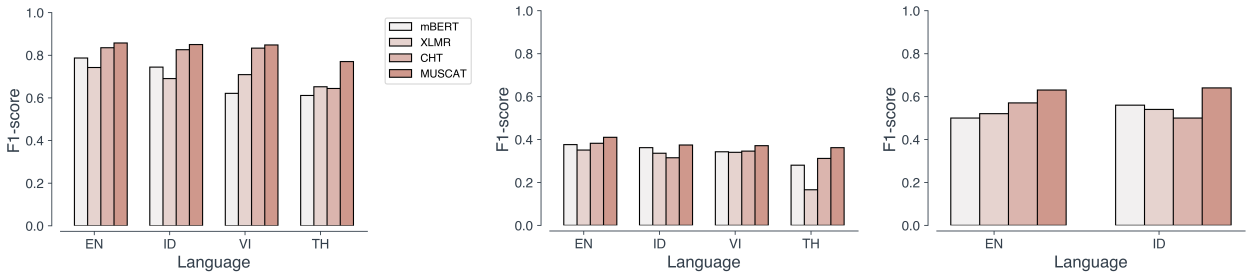
Model	TWITTER16										PHEME									
	EN		**ID**		**VI**		**TH**		**MS**		**EN**		**ID**		**VI**		**TH**		**MS**	
	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1
SVM-tfidf	0.856	0.856	0.843	0.846	0.815	0.817	0.773	0.776	0.849	0.852	0.662	0.324	0.654	0.348	0.628	0.338	0.627	0.334	0.654	0.348
SVM-bow	0.742	0.744	0.689	0.691	0.657	0.659	0.643	0.647	0.701	0.705	0.503	0.309	0.606	0.331	0.551	0.333	0.596	0.321	0.591	0.328
LSTM	0.684	0.691	0.599	0.601	0.527	0.529	0.399	0.399	0.614	0.617	0.479	0.288	0.482	0.279	0.320	0.228	0.364	0.249	0.443	0.291
TD-RvNN	0.811	0.810	0.797	0.793	0.785	0.780	0.689	0.683	0.801	0.798	0.595	0.306	0.604	0.315	0.514	0.232	0.457	0.262	0.615	0.311
BiGCN	0.840	0.830	0.830	0.825	0.825	0.825	0.644	0.644	0.798	0.793	0.652	0.236	0.657	0.258	0.650	0.224	0.631	0.236	0.689	0.273
mBERT	0.787	0.787	0.746	0.744	0.631	0.621	0.617	0.611	0.758	0.757	0.634	0.376	0.543	0.362	0.554	0.343	0.467	0.280	0.572	0.339
XLM-R	0.746	0.742	0.691	0.690	0.711	0.709	0.659	0.652	0.711	0.708	0.605	0.351	0.565	0.336	0.601	0.340	0.236	0.166	0.550	0.324
CHT	0.835	0.835	0.826	0.826	0.833	0.834	0.644	0.645	0.820	0.820	**0.686**	0.382	0.668	0.315	**0.675**	0.346	**0.694**	0.312	0.667	0.327
MUSCAT	**0.857**	**0.857**	**0.849**	**0.850**	**0.848**	**0.848**	**0.775**	**0.777**	**0.850**	**0.850**	0.665	**0.410**	**0.670**	**0.374**	0.651	**0.371**	0.665	**0.362**	**0.700**	**0.372**

Table 4.4: Monolingual results on SEAR.

Model	**EN**		**ID**	
	Acc	F1	Acc	F1
SVM-tfidf	0.658	0.520	0.779	0.489
SVM-bow	0.608	0.499	0.752	0.504
LSTM	0.518	0.441	0.603	0.436
TD-RvNN	0.561	0.398	0.714	0.425
BiGCN	0.679	0.546	0.785	0.475
mBERT	0.529	0.508	0.761	0.567
XLM-R	0.522	0.523	0.744	0.548
CHT	0.601	0.571	0.774	0.508
MUSCAT	**0.665**	**0.634**	**0.811**	**0.648**

models as the other baselines are mainly designed for monolingual settings and cannot be used for multilingual rumour verification. Similar to the monolingual experiments, MUSCAT outperforms the state-of-the-art baselines in the three datasets. Specifically, MUSCAT outperforms the best baseline by 6%, 2%, and 4% F1 scores on the Twitter16, PHEME, and SEAR datasets, respectively. More interestingly, we noted that the performance of the various models did not worsen in multilingual rumour verification. This demonstrates the strength of multilingual PLMs in transferring the knowledge learned across languages for multilingual rumour verification.

Cross-Lingual. To further examine MUSCAT's ability to transfer knowledge learned across languages, we performed the cross-lingual evaluation on Twitter16 and SEAR datasets. Figure 4.6 shows the F1 scores for MUSCAT's cross-lingual rumour verification performance on the two datasets. We observe that the MUSCAT performed well in the Twitter16 dataset, achieving more than 0.65 F1 scores in the cross-lingual



(a) Twitter16 **(b)** PHEME **(c)** SEAR

Figure 4.4: Mono-lingual performance on (a) Twitter16, (b) PHEME and (c) SEAR.

Table 4.5: Multilingual results on Twitter16 and PHEME.

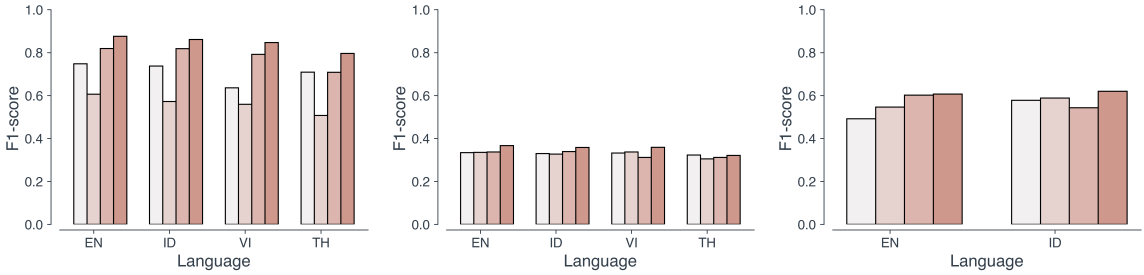
Model	TWITTER16										PHEME									
	EN		**ID**		**VI**		**TH**		**MS**		**EN**		**ID**		**VI**		**TH**		**MS**	
	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1
mBERT	0.749	0.748	0.737	0.737	0.627	0.616	0.642	0.636	0.710	0.709	0.656	0.334	0.654	0.330	0.657	0.332	0.640	0.322	0.643	0.323
XLM-R	0.609	0.606	0.575	0.572	0.562	0.559	0.514	0.507	0.518	0.511	0.648	0.335	0.639	0.327	0.643	0.337	0.585	0.305	0.632	0.316
CHT	0.818	0.819	0.817	0.818	0.791	0.792	0.707	0.708	0.800	0.801	0.664	0.337	**0.679**	0.339	**0.674**	0.312	**0.653**	0.312	**0.676**	0.324
MUSCAT	**0.877**	**0.876**	**0.862**	**0.861**	**0.847**	**0.847**	**0.796**	**0.796**	**0.856**	**0.855**	0.639	**0.367**	0.630	**0.358**	0.652	**0.359**	0.532	**0.321**	0.618	**0.344**

setting. Specifically, MUSCAT has achieved a high F1 score of 0.77 when trained on Bahasa Indonesia (i.e., source language) and tested on Bahasa Melayu, and vice versa. As Bahasa Indonesia is an isolect of Bahasa Melayu, both share some grammatical and vocabulary features. Therefore, MUSCAT may be able to transfer knowledge across languages better when the languages are related or from the same linguistic family. Interestingly, while it is easier to train on Thai rumours and transfer the knowledge to other languages, the reverse setting (i.e., Thai as the target language) is more challenging; transferring knowledge to perform Thai rumour verification has the worse F1 scores.

Surprisingly, we observe a drastic performance drop when performing cross-lingual evaluation on the SEAR dataset. As shown in Figure 4.6(b), MUSCAT has F1 score less than 0.3 when trained on Bahasa Indonesia (i.e., source language) and tested on English, and vice versa. A possible reason for the observed result might be the nature of the SEAR dataset. Unlike the translated Twitter16 dataset, the Bahasa Indonesia and English tweets were collected based on different new sources and events. Thus, when performing cross-lingual rumour verification, MUSCAT will have to verify rumours in unseen language and domain (i.e., events). This compiled the difficulty in performing cross-lingual rumour verification on the SEAR dataset.

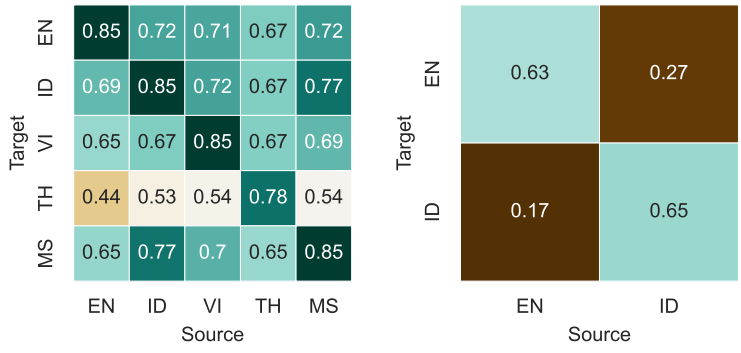
Table 4.6: Multilingual results on SEAR.

Model	EN		ID	
	Acc	F1	Acc	F1
mBERT	0.554	0.492	0.774	0.578
XLM-R	0.619	0.546	0.776	0.588
CHT	0.626	0.602	0.774	0.543
MUSCAT	**0.637**	**0.607**	**0.803**	**0.620**



(a) Twitter16 **(b)** PHEME **(c)** SEAR

Figure 4.5: Multi-lingual performance on (a) Twitter16, (b) PHEME and (c) SEAR



(a) Twitter16 **(b)** SEAR

Figure 4.6: MUSCAT's cross-lingual performance on (a) Twitter16 and (b) SEAR.

5 Conclusion

This dissertation studies two significant contributions on the applications of multilingual NLP in tackling social value tasks.

1. We show that meta-learning can improve cross-lingual transfer over standard fine-tuning for multilingual hate speech detection.

2. We present a novel Transformer-based model for multilingual rumour detection in two low-resource South East Asian languages.

This chapter provides a summary of the thesis and directions for future work.

5.1 Summary

Chapter 1 present motivation for automatic neural applications in tackling two pressing problems in social media: hate speech detection and multilingual rumour detection. First, the challenges in OSPs on the dimension of hatred and disinformation that results in discord in society are discussed. Second, neural applications and the need for developing resources in multiple languages to address social value tasks are discussed. Finally, the benefits of neural applications in social media are presented.

Chapter 2 provides the technical foundation of neural networks and their applications. The chapter starts with an overview of the fundamental concepts of neural networks. Modern architectures in text processing are then described, followed by a discussion of prior works on knowledge transfer to improve low-resource applications.

Chapter 3 present multilingual hate speech detection task in extreme low-resource setting. The system architecture is described, as well as the different language-specific resources that have been used to train the system. It use a novel meta-learning learning algorithm that benefits from resource maximization to improve cross-lingual transfer. The experiments in this chapter investigate pairwise cross-lingual transfer, cross-domain transfer and a comparison with fine-tuning methods. It shows that meta-learning is a viable alternative for hate speech detection in low-resource languages.

Chapter 4 proposes a new model for multilingual rumour detection. It formulates the problem of multilingual rumour detection, collects a dataset for two South East Asian languages, and discusses challenges in building datasets in low-resource communities. The proposed model and baseline are used to discuss multilingual rumour detection. The results show that the proposed system outperforms the baselines, and that it

is possible to detect rumours in multiple languages with a single model. We analyse our improvements and discuss remaining challenges.

5.2 Future work

Some possible future extensions of this work are discussed here.

5.2.1 Constructing datasets for social value tasks in low-resource languages

Resources in deep learning applications for tackling social value tasks are still limited. Chapter 3 investigates multilingual hate speech detection in eight languages. However, the study was limited to available hate corpora and there is a need for exploring hate speech in other languages. To facilitate further research, the construction of datasets for extreme low-resource languages should be explored. It is also useful to focus on domain-specific data for careful evaluation of cross-domain hate speech detection that matches the real-world setting. In Chapter 4, we discuss some of the difficulties while constructing rumour detection datasets in multiple languages. We believe it requires further attention to build datasets for many other low-resource languages since rumour detection in those remains an open challenging problem. In particular, future research can focus on the integration of fact-checking websites for a longer period to achieve broader coverage of rumour events.

5.2.2 Exploring meta-learning for fine-grained and implicit hate speech detection

This study can be extended to explore meta-learning for fine-grained and implicit hate speech detection. The proposed meta-learning algorithm in Chapter 3 treats hate speech as binary tasks, however, it is possible to use this framework to solve multi-class hate speech detection on fine-grained labels. Chapter 3 reveals some insights into language similarity and cross-lingual transfer. This study can be elaborated to gain further insight into other language families while training on a wide variety of language families. It is worth noting that our study was mainly devoted to explicit hate speech detection. Future work can take motivation from recent studies on implicit hate speech to investigate implicit hatred in languages beyond English. Furthermore, it would be interesting to explore meta-learning for cross-lingual transfer of implicit hate forms of varying cultures and domains.

5.2.3 Detecting rumour with fact-checking resources in multiple languages

Detecting rumours with fact-checking resources in multiple languages is a very challenging task due to the limited resources that are available. The natural extension to multilingual rumour detection would be to add more languages in the implementation of models and explore the nature of large-scale cross-lingual rumour

detection. The collected datasets are small and it is possible to extend them to multiple domains and explore resource languages beyond South East Asia. Current neural modeling only uses a single source of information for verifying rumour events. Relying on only the conversations happening in social media might make them brittle to coordinated attacks such as bots. To strengthen further knowledge utilization in these systems, we can leverage external knowledge sources such as major news outlets and fact-checking websites. A potential extension is exploring neural methods that capture signals from multiple sources beyond social media to verify a rumour event.

In conclusion, this thesis examines deep learning applications for tackling social value tasks in two challenging problems: hate speech detection and rumour detection. For both tasks, we explore resource utilization and develop neural models for automated solution. First, we present a novel algorithm for low-resource hate speech detection that maximizes the resource utilization. We show that meta-learning is a viable alternative to multilingual fine-tuning and achieve stronger performance in cross-domain setting. Second, the multilingual rumour detection can be improved by modifying Transformers to prioritize source post while processing replies. We present extensive studies on two South Asian languages and present challenges on data collection to multilingual modeling.

References

[1] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Proceedings of the Advances in Neural Information Processing Systems*, volume 27, December 2014.

[2] Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015*, May 2015.

[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Proceedings of the Advances in Neural Information Processing Systems*, 30, December 2017.

[4] Jeffrey Gottfried and Elisa Shearer. News use across social media platforms 2016. 2019.

[5] Hunt Allcott and Matthew Gentzkow. Social media and fake news in the 2016 election. *Journal of economic perspectives*, 31(2):211–36, 2017.

[6] Mehrdad Farajtabar, Jiachen Yang, Xiaojing Ye, Huan Xu, Rakshit Trivedi, Elias Khalil, Shuang Li, Le Song, and Hongyuan Zha. Fake news mitigation via point process based intervention. In *Proceedings of the International Conference on Machine Learning*, pages 1097–1106, 2017.

[7] Mainack Mondal, Leandro Araújo Silva, and Fabrício Benevenuto. A measurement study of hate speech in social media. In *Proceedings of the 28th ACM Conference on Hypertext and Social Media*, HT '17, page 85–94, New York, NY, USA, 2017.

[8] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

[9] Nenad Tomašev, Julien Cornebise, Frank Hutter, Shakir Mohamed, Angela Picciariello, Bec Connelly, Danielle Belgrave, Daphne Ezer, Fanny Cachat van der Haert, Frank Mugisha, et al. AI for social good: unlocking the opportunity for positive impact. *Nature Communications*, 11(1):1–6, 2020.

[10] Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. Automated hate speech detection and the problem of offensive language. In *Proceedings of the 11th International AAAI Conference on Web and Social Media*, ICWSM '17, pages 512–515, 2017.

[11] Zeerak Waseem and Dirk Hovy. Hateful symbols or hateful people? Predictive features for hate speech detection on Twitter. In *Proceedings of the NAACL Student Research Workshop*, pages 88–93, San Diego, California, June 2016.

[12] Jing Ma, Wei Gao, and Kam-Fai Wong. Detect rumors in microblog posts using propagation structure via kernel learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 708–717, Vancouver, Canada, July 2017.

[13] Ke Wu, Song Yang, and Kenny Q. Zhu. False rumors detection on sina weibo by propagation structures. *Proceedings of the 2015 IEEE 31st International Conference on Data Engineering*, pages 651–662, 2015.

[14] Quanzhi Li, Qiong Zhang, Luo Si, and Yingchi Liu. Rumor detection on social media: Datasets, methods and opportunities. In *Proceedings of the Second Workshop on Natural Language Processing for Internet Freedom: Censorship, Disinformation, and Propaganda*, pages 66–75, Hong Kong, China, November 2019.

[15] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25, December 2012.

[16] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, 2003.

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

[18] Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. SuperGlue: A stickier benchmark for general-purpose language understanding systems. *Proceedings of the Advances in Neural Information Processing Systems*, 32, December 2019.

[19] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Proceedings of the Advances in Neural Information Processing Systems*, 33:1877–1901, December 2020.

[20] Zeerak Waseem. Are you a racist or am i seeing things? annotator influence on hate speech detection on twitter. In *Proceedings of the First Workshop on NLP and Computational Social Science*, pages 138–142, 2016.

[21] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina 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)*, pages 4171–4186, 2019.

[22] Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Manuel Rangel Pardo, Paolo Rosso, and Manuela Sanguinetti. SemEval-2019 task 5: Multilingual detection of hate speech against immigrants and women in Twitter. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 54–63, Minneapolis, Minnesota, USA, June 2019.

[23] Rui Cao, Roy Ka-Wei Lee, and Tuan-Anh Hoang. Deephate: Hate speech detection via multi-faceted text representations. In *Proceedings of the 12th ACM Conference on Web Science*, pages 11–20, July 2020.

[24] Michael Wiegand, Josef Ruppenhofer, and Elisabeth Eder. Implicitly abusive language–what does it actually look like and why are we not getting there? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 576–587, 2021.

[25] Jing Ma, Wei Gao, and Kam-Fai Wong. Rumor detection on Twitter with tree-structured recursive neural networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1980–1989, Melbourne, Australia, July 2018.

[26] Damián Blasi, Antonios Anastasopoulos, and Graham Neubig. Systematic inequalities in language technology performance across the world's languages. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5486–5505, 2022.

[27] Shijie Wu and Mark Dredze. Beto, Bentz, Becas: The surprising cross-lingual effectiveness of BERT. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 833–844, Hong Kong, China, November 2019.

[28] Marcos Zampieri, Preslav Nakov, Sara Rosenthal, Pepa Atanasova, Georgi Karadzhov, Hamdy Mubarak, Leon Derczynski, Zeses Pitenis, and Çağrı Çöltekin. SemEval-2020 task 12: Multilingual offensive language identification in social media (OffensEval 2020). In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 1425–1447, Barcelona (online), December 2020.

[29] Thomas Mandl, Sandip Modha, Anand Kumar M, and Bharathi Raja Chakravarthi. Overview of the hasoc track at fire 2020: Hate speech and offensive language identification in tamil, malayalam, hindi, english and german. In *Forum for Information Retrieval Evaluation*, FIRE 2020, page 29–32, New York, NY, USA, 2020.

[30] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[31] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

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

[33] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[34] Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the European Conference on Machine Learning*, pages 137–142. Springer, 1998.

[35] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(ARTICLE):2493–2537, 2011.

[36] Geoffrey E Hinton. Learning multiple layers of representation. *Trends in Cognitive Sciences*, 11(10):428–434, 2007.

[37] Clement Farabet, Camille Couprie, Laurent Najman, and Yann LeCun. Learning hierarchical features for scene labeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1915–1929, 2012.

[38] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *Proceedings of the International Conference on Machine Learning*, pages 1310–1318, 2013.

[39] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *Proceedings of the IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013.

[40] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[41] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[42] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010.

[43] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

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

[45] Jeffrey L Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990.

[46] Yoshua Bengio. Deep learning of representations for unsupervised and transfer learning. In *Proceedings of the ICML workshop on unsupervised and transfer learning*, pages 17–36, 2012.

[47] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

[48] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, Jürgen Schmidhuber, et al. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001.

[49] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.

[50] Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, Doha, Qatar, October 2014.

[51] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *Proceedings of the 2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649, 2013.

[52] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California, June 2016.

[53] Kaisheng Yao, Geoffrey Zweig, and Baolin Peng. Attention with intention for a neural network conversation model. *arXiv preprint arXiv:1510.08565*, 2015.

[54] Klaus Greff, Rupesh K Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. LSTM: A search space odyssey. *Proceedings of the IEEE Transactions on Neural Networks and Learning Systems*, 28(10):2222–2232, 2016.

[55] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

[56] Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional LSTM networks. In *Proceedings of the 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 4, pages 2047–2052, 2005.

[57] Martin Sundermeyer, Tamer Alkhouli, Joern Wuebker, and Hermann Ney. Translation modeling with bidirectional recurrent neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 14–25, Doha, Qatar, October 2014.

[58] Zhiheng Huang, Wei Xu, and Kai Yu. Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991*, 2015.

[59] Wei Cao, Dong Wang, Jian Li, Hao Zhou, Lei Li, and Yitan Li. Brits: Bidirectional recurrent imputation for time series. *Proceedings of the Advances in Neural Information Processing Systems*, 31, December 2018.

[60] Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal, September 2015.

[61] Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. Hierarchical question-image co-attention for visual question answering. *Proceedings of the Advances in Neural Information Processing Systems*, 29, December 2016.

[62] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2048–2057, Lille, France, 07–09 Jul 2015.

[63] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489, San Diego, California, June 2016.

[64] Zichao Yang, Xiaodong He, Jianfeng Gao, Li Deng, and Alex Smola. Stacked attention networks for image question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 21–29, 2016.

[65] Daniel Neil, Michael Pfeiffer, and Shih-Chii Liu. Phased LSTM: Accelerating recurrent network training for long or event-based sequences. In *Proceedings of the Advances in Neural Information Processing Systems*, volume 29, December 2016.

[66] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

[67] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

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

[69] Ofir Press, Noah A Smith, and Mike Lewis. Train short, test long: Attention with linear biases enables input length extrapolation. *arXiv preprint arXiv:2108.12409*, 2021.

[70] Vighnesh Shiv and Chris Quirk. Novel positional encodings to enable tree-based transformers. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Proceedings of the Advances in Neural Information Processing Systems*, volume 32, 2019.

[71] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[72] Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, et al. A generalist agent. *arXiv preprint arXiv:2205.06175*, 2022.

[73] Christopher Manning and Hinrich Schutze. *Foundations of statistical natural language processing*. MIT press, 1999.

[74] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. Exploring the limits of transfer learning with a unified text-to-text transformer. *Joural of Machine Learning Research*, 21(140):1–67, 2020.

[75] Peter F Brown, Vincent J Della Pietra, Peter V Desouza, Jennifer C Lai, and Robert L Mercer. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–480, 1992.

[76] Claude Elwood Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948.

[77] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019.

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

[79] Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. How to fine-tune bert for text classification? In Maosong Sun, Xuanjing Huang, Heng Ji, Zhiyuan Liu, and Yang Liu, editors, *Proceedings of the Chinese Computational Linguistics*, pages 194–206, Cham, 2019.

[80] Taku Kudo and John Richardson. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*, 2018.

[81] Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. Word translation without parallel data. *arXiv preprint arXiv:1710.04087*, 2017.

[82] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online, July 2020.

[83] Junjie Hu, Sebastian Ruder, Aditya Siddhant, Graham Neubig, Orhan Firat, and Melvin Johnson. XTREME: A massively multilingual multi-task benchmark for evaluating cross-lingual generalisation. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 4411–4421, 13–18 Jul 2020.

[84] Telmo Pires, Eva Schlinger, and Dan Garrette. How multilingual is multilingual BERT? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4996–5001, Florence, Italy, July 2019.

[85] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *Proceedings of the IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2009.

[86] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *Advances in Neural Information Processing Systems*, 27, December 2014.

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

[88] Bernardino Romera-Paredes and Philip Torr. An embarrassingly simple approach to zero-shot learning. In *Proceedings of the International Conference on Machine Learning*, pages 2152–2161, 2015.

[89] Mikel Artetxe and Holger Schwenk. Massively Multilingual Sentence Embeddings for Zero-Shot Cross-Lingual Transfer and Beyond. *Transactions of the Association for Computational Linguistics*, 7:597–610, 09 2019.

[90] Yujia Bao, Menghua Wu, Shiyu Chang, and Regina Barzilay. Few-shot text classification with distributional signatures. In *Proceedings of International Conference on Learning Representations*, April 2020.

[91] Li Fei-Fei, Robert Fergus, and Pietro Perona. One-shot learning of object categories. *Proceedings of the IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4):594–611, 2006.

[92] Tianyu Gao, Xu Han, Hao Zhu, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. FewRel 2.0: Towards more challenging few-shot relation classification. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6250–6255, Hong Kong, China, November 2019.

[93] Ruiying Geng, Binhua Li, Yongbin Li, Yuxiao Ye, Ping Jian, and Jian Sun. Few-shot text classification with induction network. *CoRR*, abs/1902.10482, 2019.

[94] Chengcheng Han, Zeqiu Fan, Dongxiang Zhang, Minghui Qiu, Ming Gao, and Aoying Zhou. Meta-learning adversarial domain adaptation network for few-shot text classification. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1664–1673, Online, August 2021.

[95] Brenden Lake, Ruslan Salakhutdinov, Jason Gross, and Joshua Tenenbaum. One shot learning of simple visual concepts. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 33, 2011.

[96] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning, 2018.

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

[98] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135, 06–11 Aug 2017.

[99] Farhad Nooralahzadeh, Giannis Bekoulis, Johannes Bjerva, and Isabelle Augenstein. Zero-shot cross-lingual transfer with meta learning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4547–4562, Online, November 2020.

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

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

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

[103] Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. Multi-task deep neural networks for natural language understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4487–4496, Florence, Italy, July 2019.

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

[105] Mingsheng Long, ZHANGJIE CAO, Jianmin Wang, and Philip S Yu. Learning multiple tasks with multilinear relationship networks. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 1593–1602. December 2017.

[106] Yongxi Lu, Abhishek Kumar, Shuangfei Zhai, Yu Cheng, Tara Javidi, and Rogério Schmidt Feris. Fully-adaptive feature sharing in multi-task networks with applications in person attribute classification. In *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 1131–1140, 2017.

[107] Sen Wu, Hongyang Zhang, and Christopher Ré. Understanding and improving information transfer in multi-task learning. *ArXiv*, abs/2005.00944, 2020.

[108] Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. A joint many-task model: Growing a neural network for multiple NLP tasks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1923–1933, Copenhagen, Denmark, September 2017.

[109] Ishan Misra, Abhinav Shrivastava, A. Gupta, and M. Hebert. Cross-stitch networks for multi-task learning. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3994–4003, 2016.

[110] Hal Daumé III. Frustratingly easy domain adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 256–263, Prague, Czech Republic, June 2007.

[111] Bill Yuchen Lin and Wei Lu. Neural adaptation layers for cross-domain named entity recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2012–2022, Brussels, Belgium, October-November 2018.

[112] Jenny Rose Finkel and Christopher D. Manning. Hierarchical Bayesian domain adaptation. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 602–610, Boulder, Colorado, June 2009.

[113] John Blitzer, Ryan McDonald, and Fernando Pereira. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 120–128, Sydney, Australia, July 2006.

[114] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario March, and Victor Lempitsky. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17(59):1–35, 2016.

[115] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. Learning transferable features with deep adaptation networks. In *Proceedings of the International Conference on Machine Learning*, pages 97–105, 2015.

[116] Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. Domain separation networks. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Proceedings of the Advances in Neural Information Processing Systems*, volume 29, December 2016.

[117] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1180–1189, Lille, France, 07–09 Jul 2015.

[118] Ji Ho Park and Pascale Fung. One-step and two-step classification for abusive language detection on Twitter. In *Proceedings of the First Workshop on Abusive Language Online*, pages 41–45, Vancouver, BC, Canada, August 2017.

[119] Tharindu Ranasinghe and Marcos Zampieri. Multilingual offensive language identification with cross-lingual embeddings. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5838–5844, Online, November 2020.

[120] Sai Saketh Aluru, Binny Mathew, Punyajoy Saha, and Animesh Mukherjee. A deep dive into multilingual hate speech classification. In *ECML/PKDD*, 2020.

[121] Endang Wahyu Pamungkas and Viviana Patti. Cross-domain and cross-lingual abusive language detection: A hybrid approach with deep learning and a multilingual lexicon. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 363–370, Florence, Italy, July 2019.

[122] Debora Nozza. Exposing the limits of zero-shot cross-lingual hate speech detection. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 907–914, Online, August 2021.

[123] Hala Al Kuwatly, Maximilian Wich, and Georg Groh. Identifying and measuring annotator bias based on annotators' demographic characteristics. In *Proceedings of the Fourth Workshop on Online Abuse and Harms*, pages 184–190, 2020.

[124] Antigoni-Maria Founta, Constantinos Djouvas, Despoina Chatzakou, Ilias Leontiadis, Jeremy Blackburn, Gianluca Stringhini, Athena Vakali, Michael Sirivianos, and Nicolas Kourtellis. Large scale crowdsourcing and characterization of twitter abusive behavior. In *Proceedings of the 11th International Conference on Web and Social Media, ICWSM 2018*, 2018.

[125] Thomas Mandl, Sandip Modha, Prasenjit Majumder, Daksh Patel, Mohana Dave, Chintak Mandlia, and Aditya Patel. Overview of the hasoc track at fire 2019: Hate speech and offensive content identification in indo-european languages. In *Proceedings of the 11th Forum for Information Retrieval Evaluation*, FIRE '19, page 14–17, New York, NY, USA, 2019.

[126] Elisabetta Fersini, Debora Nozza, and Paolo Rosso. Overview of the evalita 2018 task on automatic misogyny identification (ami). In *EVALITA@CLiC-it*, 2018.

[127] Ilia Markov, Nikola Ljubešić, Darja Fišer, and Walter Daelemans. Exploring stylometric and emotion-based features for multilingual cross-domain hate speech detection. In *Proceedings of the Eleventh Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 149–159, Online, April 2021.

[128] Irina Bigoulaeva, Viktor Hangya, and Alexander Fraser. Cross-lingual transfer learning for hate speech detection. In *Proceedings of the First Workshop on Language Technology for Equality, Diversity and Inclusion*, pages 15–25, Kyiv, Ukraine, April 2021.

[129] Tomer Wullach, Amir Adler, and Einat Minkov. Fight fire with fire: Fine-tuning hate detectors using large samples of generated hate speech, 2021.

[130] Aiqi Jiang and Arkaitz Zubiaga. Cross-lingual capsule network for hate speech detection in social media. In *Proceedings of the 32nd ACM Conference on Hypertext and Social Media*, HT '21, page 217–223, New York, NY, USA, 2021.

[131] Hajung Sohn and Hyunju Lee. MC-BERT4HATE: Hate speech detection using multi-channel bert for different languages and translations. In *2019 International Conference on Data Mining Workshops (ICDMW)*, pages 551–559, 2019.

[132] Shubhanshu Mishra. 3idiots at hasoc 2019: Fine-tuning transformer neural networks for hate speech identification in indo-european languages. In *Proceedings of Forum for Information Retrieval Evaluation (FIRE)*, 2019.

[133] Punyajoy Saha, Binny Mathew, Pawan Goyal, and Animesh Mukherjee. Hatemonitors: Language agnostic abuse detection in social media. *CoRR*, abs/1909.12642, 2019.

[134] Jonathan Gordon, John Bronskill, Matthias Bauer, Sebastian Nowozin, and Richard E. Turner. Meta-learning probabilistic inference for prediction. 2019.

[135] Gregory Koch, Richard Zemel, Ruslan Salakhutdinov, et al. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2, page 0. Lille, 2015.

[136] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, koray kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Proceedings of Advances in Neural Information Processing Systems*, volume 29, December 2016.

[137] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy Hospedales. Learning to generalize: Meta-learning for domain generalization. In *Proceedings of AAAI Conference on Artificial Intelligence*, 2018.

[138] Jason Wei and Kai Zou. EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6383–6389, Hong Kong, China, November 2019.

[139] Demi Guo, Yoon Kim, and Alexander Rush. Sequence-level mixed sample data augmentation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5547–5552, Online, November 2020.

[140] Amit Jindal, Arijit Ghosh Chowdhury, Aniket Didolkar, Di Jin, Ramit Sawhney, and Rajiv Ratn Shah. Augmenting NLP models using latent feature interpolations. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6931–6936, Barcelona, Spain (Online), December 2020.

[141] Xinyi Wang, Hieu Pham, Zihang Dai, and Graham Neubig. SwitchOut: an efficient data augmentation algorithm for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 856–861, Brussels, Belgium, October-November 2018.

[142] Steven Y. Feng, Aaron W. Li, and Jesse Hoey. Keep calm and switch on! Preserving Sentiment and Fluency in Semantic Text Exchange. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2701–2711, Hong Kong, China, November 2019.

[143] Sosuke Kobayashi. Contextual augmentation: Data augmentation by words with paradigmatic relations. In Marilyn A. Walker, Heng Ji, and Amanda Stent, editors, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 2 (Short Papers)*, pages 452–457, 2018.

[144] Rico Sennrich, Barry Haddow, and Alexandra Birch. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany, August 2016.

[145] Manuela Sanguinetti, Gloria Comandini, Elisa Di Nuovo, Simona Frenda, Marco Stranisci, Cristina Bosco, Tommaso Caselli, Viviana Patti, Irene Russo, and ILCCNR Pisa. HaSpeeDe 2@ EVALITA2020: Overview of the EVALITA 2020 hate speech detection task. In *EVALITA*, 2020.

[146] Arkaitz Zubiaga, Ahmet Aker, Kalina Bontcheva, Maria Liakata, and Rob Procter. Detection and resolution of rumours in social media: A survey. *ACM Computing Surveys (CSUR)*, 51(2):1–36, 2018.

[147] Arkaitz Zubiaga, Maria Liakata, Rob Procter, Geraldine Wong Sak Hoi, and Peter Tolmie. Analysing how people orient to and spread rumours in social media by looking at conversational threads. *Public Library of Science ONE*, 11(3):1–29, 03 2016.

[148] Ling Min Serena Khoo, Hai Leong Chieu, Zhong Qian, and Jing Jiang. Interpretable rumor detection in microblogs by attending to user interactions. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8783–8790, Apr. 2020.

[149] Elena Kochkina, Maria Liakata, and Isabelle Augenstein. Turing at SemEval-2017 task 8: Sequential approach to rumour stance classification with branch-LSTM. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 475–480, Vancouver, Canada, August 2017.

[150] Xiaomo Liu, Armineh Nourbakhsh, Quanzhi Li, Rui Fang, and Sameena Shah. Real-time rumor debunking on twitter. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, page 1867–1870, New York, NY, USA, 2015.

[151] Vahed Qazvinian, Emily Rosengren, Dragomir R. Radev, and Qiaozhu Mei. Rumor has it: Identifying misinformation in microblogs. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, page 1589–1599, USA, 2011.

[152] Qiao Zhang, Shuiyuan Zhang, Jian Dong, Jinhua Xiong, and Xueqi Cheng. Automatic detection of rumor on social network. In *NLPCC*, 2015.

[153] Zhe Zhao, Paul Resnick, and Qiaozhu Mei. Enquiring minds: Early detection of rumors in social media from enquiry posts. *Proceedings of the 24th International Conference on World Wide Web*, 2015.

[154] Tian Bian, Xi Xiao, Tingyang Xu, Peilin Zhao, Wenbing Huang, Yu Rong, and Junzhou Huang. Rumor detection on social media with bi-directional graph convolutional networks. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 549–556, 2020.

[155] Tong Chen, Xue Li, Hongzhi Yin, and Jun Zhang. Call attention to rumors: Deep attention based recurrent neural networks for early rumor detection. In Mohadeseh Ganji, Lida Rashidi, Benjamin C. M. Fung, and Can Wang, editors, *Proceedings of the Trends and Applications in Knowledge Discovery and Data Mining*, pages 40–52, Cham, 2018.

[156] Bo Liu, Xiangguo Sun, Qing Meng, Xinyan Yang, Yang Lee, Jiuxin Cao, Junzhou Luo, and Roy Ka-Wei Lee. Nowhere to hide: Online rumor detection based on retweeting graph neural networks. *Proceedings of the IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[157] Jing Ma and Wei Gao. Debunking rumors on twitter with tree transformer. In *COLING*, 2020.

[158] Kaimin Zhou, Chang Shu, Binyang Li, and Jey Han Lau. Early rumour detection. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1614–1623, Minneapolis, Minnesota, June 2019.

[159] Elena Kochkina, Maria Liakata, and Arkaitz Zubiaga. All-in-one: Multi-task learning for rumour verification. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3402–3413, Santa Fe, New Mexico, USA, August 2018.

[160] Sumeet Kumar and Kathleen Carley. Tree LSTMs with convolution units to predict stance and rumor veracity in social media conversations. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5047–5058, Florence, Italy, July 2019.

[161] Jianfei Yu, Jing Jiang, Ling Min Serena Khoo, Hai Leong Chieu, and Rui Xia. Coupled hierarchical transformer for stance-aware rumor verification in social media conversations. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1392–1401, Online, November 2020.

[162] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.

[163] R. Pappagari, Piotr Żelasko, Jesús Villalba, Yishay Carmiel, and Najim Dehak. Hierarchical transformers for long document classification. *Proceedings of the 2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 838–844, 2019.

[164] Xingxing Zhang, Furu Wei, and Ming Zhou. HIBERT: Document level pre-training of hierarchical bidirectional transformers for document summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5059–5069, Florence, Italy, July 2019.

[165] Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer. *ArXiv*, abs/2004.05150, 2020.

[166] Mary L. McHugh. Interrater reliability: the kappa statistic. *Biochemia Medica*, 22(3):276–282, October 2012.

[167] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.