GRANULAR EMOTION DETECTION FOR MULTI-CLASS SENTIMENT ANALYSIS IN SOCIAL MEDIA

by

Robert H. Frye

A dissertation submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in
Computing and Information Systems

Charlotte

2022

Approved by:

Dr. David C. Wilson

Dr. Xi Niu

Dr. Nadia Najjar

Dr. Yaorong Ge

Dr. Taghi Mostafavi

ABSTRACT

ROBERT H. FRYE. Granular emotion detection for multi-class sentiment analysis in social media. (Under the direction of DR. DAVID C. WILSON)

Sentiment analysis for text classification generally refers to assessing the polarity of the emotional context of written text, whether in a binary (e.g. positive or negative) or trinary (e.g. positive, neutral, or negative) state. Granular emotion detection is a more specialized form of sentiment analysis, wherein we move from predicting sentiment polarity to detecting specific classes of emotions within text (e.g. happy, sad, anger, love, hate, etc.), whether that context is a reflection of the author's own emotional state or the emotional state the author intended to convey. Granular emotion detection is broadly applicable to the business world, with common applications in customer satisfaction and retention, as well as studies of marketing effectiveness. Other applications include attempting to identify angry people based on their social media posts and prevent them from committing acts of violence. Current approaches to multi-class emotion classification show mixed or limited results, and improving accuracy for multiple classes of emotions is an open research challenge. Moreover, many modern application contexts align more directly with social media content or have a shorter format more akin to social media, where texts often bend or violate standard language conventions. Overall, understanding emotion detection in social media (EMDISM) contexts is an open challenge.

To address the challenge of granular emotion detection in social media text, I have investigated ensemble approaches that combine a variety of individual classifiers to address tradeoffs in performance. This involved first investigating EMDISM performance for individual traditional machine learning (ML), deep learning (DL), and transformer learning (TL) classifiers. Based on this analysis, the second stage investigated the creation of ensembles of the most accurate classifiers across these general

classes which offer comparatively improved performance. The approaches were evaluated based on a large Twitter dataset with more than 1.2M samples and encompassing seven discrete emotions. I provide results and analysis for each classifier I considered as well as the most accurate ensembles I created from the most accurate singleton classifiers. Results show that the proposed ensemble approaches - simple voting, weighted voting, cascading, and cascading/switching - improve upon the state of the art for average accuracy, weighted precision, weighted recall, and weighted f-measure as compared to the most accurate single classifier for EMDISM.

DEDICATION

This work is dedicated to my loving wife, Mrs. Jody R. Frye, and our three children, Audrey West, Samuel Frye, and Layna West. Each has contributed in some way to a long journey that began with returning to complete my degree in Computer Science in 2011. Without the help, encouragement, patience, and tolerance of every person in my family, none of this research would have been possible. I hold each of you close to my heart, and I love and cherish you always.

I also dedicate this research to my friend and mentor, Dr. Bjarne Berg-Sæther. Your guidance along the way in my career, my education, and in life lessons has been invaluable. Thanks for being a friend and mentor, for encouraging me when needed, for pushing me when required, and for believing in me.

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my PhD Advisor, Dr. David C. Wilson for his patience, diligence, and effective guidance towards completing my research. I also want to thank Dr. Samira Shaikh, Dr. Xi Niu, and Dr. Yaorong Ge for serving on my qualifying committee and my dissertation committee and for your patience and your thought-provoking guidance throughout this process. Thank you, as well, Dr. Nadia Najjar for joining my dissertation committee late in the process and providing valuable feedback prior to my defense.

I would also like to thank Armin Seyeditabari for his assistance in hydrating the large Twitter dataset used in conducting my research. Your assistance arrived at the perfect time to enable my dissertation research, so thank you sincerely for pointing me in the right direction to build the dataset I needed.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

BERT An acronym for Bidirectional Encoder Representations from Transformers, a type of transformer used for text classification.

C-LSTM An abbreviation for Convolutional Long Short-Term Memory, a type of neural network used for text classification.

CNN An abbreviation for Convolutional Neural Network.

DL An abbreviation for Deep Learning.

DT An abbreviation for Decision Tree.

ELECTRA An acronym for Efficiently Learning an Encoder that Classifies Token Replacements Accurately, a type of transformer used for text classification.

EMDISM An acronym for Emotion Detection In Social Media.

FAA An abbreviation for Forensic Authorship Attribution.

GRU An abbreviation for Gated Recurrent Unit, a type of neural network used for text classification.

KNN An abbreviation for K-Nearest Neighbor, a traditional machine learning clustering approach used for text classification.

LSTM An abbreviation for Long Short-Term Memory, a type of neural network used for text classification.

ML An abbreviation for Machine Learning.

NB An abbreviation for Naïve Bayes, a traditional machine learning classifier used for text classification.

NER An abbreviation for Named Entity Recognition.

NLP An abbreviation for Natural Language Processing.

POS An abbreviation for Part Of Speech, usually referencing tagging text during pre-processing.

RNN An abbreviation for Recurrent Neural Network.

RoBERTa An acronym for Robustly Optimized BERT Pretraining Approach, a type of transformer used for text classification.

SRL An abbreviation for Semantic Role Labeling.

SVM An abbreviation for Support Vector Machine, a traditional machine learning classifier used for text classification.

TF-IDF An abbreviation for Term Frequency - Inverse Document Frequency, an approach used to focus a tokenizer on the most important words of a corpus, based on their frequency of use within a given document.

TL An abbreviation for Transformer Learning.

XLM An abbreviation for Cross-lingual Language Models, a type of transformer used for text classification.

XLM-RoBERTa An abbreviation for a transformer approach combining aspects of XLM and RoBERTa, a type of transformer used for text classification.

CHAPTER 1: INTRODUCTION

Understanding a person's mood and circumstances by way of sentiment analysis or finer-grained emotion detection can play a significant role in intelligent systems and modern applications. More specifically, *understanding emotion in written text* is important across a wide variety of disciplines. Ranganathan [1] notes that "Emotion mining has its root in many disciplines apart from computer science as follows: human science, psychiatry, nursing, psychology, neuro-science, linguistics, social science, anthropology, communication science, economics, criminology, political-science, philosophy etc." Yue et al. [2] highlight politics, public security, and commercial as three important application areas. In politics, sentiment and emotion data can help inform political strategies for both candidates and incumbents. And recognizing sentiment-driven trends in social media can help to understand emerging, potentially disruptive world events. In addition, commercial applications can provide valuable reaction-based feedback to merchants, manufacturers, and consumers about the quality and reception of different products.

For example, customer service centers can employ sentiment analysis to route emails to the correct customer engagement representative [3] or identify frustrated and dissatisfied customers based on email content [4]. Moreover, understanding emotion in text is particularly important in the specific context of social media, where texts tend to be shorter and less formal. For example, airlines monitor social media for upset customers [5], and marketers can measure the emotional impact of advertising and tailor their ads based on customer responses [6, 7]. Beyond commercial applications, mental health providers can identify people at risk of depression based on the content of their social media posts [8]. And research has shown that sentiment analysis

can play an important role in understanding extremist content online [9, 10, 11, 12], which could help to identify real threats [13, 14, 15, 16, 11]. Overall, more accurate sentiment analysis and emotion detection, particularly in the social media context, can drive applications with significant benefits across society.

1.1 Sentiment Analysis and Emotion Detection

Sentiment Analysis investigates classification and related methods to identify the emotional context of written text, whether that context is a reflection of the author's own emotional state or the emotional state the author intended to convey. Basic forms of sentiment analysis typically involve a binary (positive vs. negative feeling) or trinary (positive vs. neutral vs. negative feeling) classification of emotions. To illustrate, consider a possible written review about a popular movie, *"I hate Avengers: Infinity Wars. Worst movie ever!"* It is reasonably clear that the author is presenting a strong negative sentiment. Now consider a second possible review of the same movie. *"Infinity War was too exciting to hate, too long to watch in one sitting, and too heartbreaking to love."* It should be reasonably clear that sentiment is being expressed, but there is much greater ambiguity, and even expert human evaluators may be hard pressed to agree on a straightforward positive or negative view. More generally, an accurate understanding of emotional content in written text can depend on factors such as contextual understanding of the subject matter, knowledge of societal events, an understanding of sarcasm, and even cultural norms which may differ between audiences. Overall, text annotation and classification for sentiment analysis is an open and challenging research problem [17].

Emotion detection in text is an advanced form of sentiment analysis, which considers a more granular categorization of expressed emotion (e.g., happiness, sadness, anger, fear), which goes beyond beyond general sentiment - positive / negative / neutral. Because it makes more nuanced distinctions, emotion detection is typically more challenging than basic sentiment analysis. In the second of the previous movie

review examples, part of the difficulty in assigning positive and negative sentiment is that the text contains emotions from competing binary categories. For example, the phrase *"...too exciting to hate..."* references both positive (*joy*, *excitement*) and negative (*hate*) emotions. This kind of complexity presents significant open challenges for understanding emotional content in written text.

Emotion detection is grounded in understanding the range of possible emotions. Shaver et. al [18] and Ekman [19] established six basic emotions, including **anger, disgust, fear, happiness, sadness,** and **surprise** in the context of facial expressions, which are useful as foundational categories for emotion detection. However, a much wider variety of emotion classes is possible. As Theodore D. Kemper noted [20]:

> *Fundamental in the field of emotions is the question of how many emotions there are or there can be. The answer proposed here is that the number of possible emotions is limitless. As long as society differentiates new social situations, labels them, and socializes individuals to experience them, new emotions will continue to emerge.*

The subjective, complex, and evolving nature of emotions creates many unique challenges in the field of emotion detection within the broader area of natural language processing research. The overall challenge of emotion detection is to accurately identify foundational fine-grained emotion categories, but with sufficient flexibility to address an evolving spectrum of emotions.

To help address this challenge, my research focuses on the finer-grained task of emotion detection, investigating ensemble approaches to more accurately classify emotions in text. This draws upon foundational research in both sentiment analysis and emotion detection, and I will refer to both in the development of this research. In addition, the nature of the text being considered also provides important context for classification, and my research focuses on emotion detection specifically for shorter,

less formal written text in the context of social media.

1.2 Emotion Mining in Social Media

The task of correctly identifying specific emotions portrayed in written text is challenging, even with richer data where the text is longer, well-written and closely follows rules of grammar, spelling, punctuation, and style. Given that texts for user interactions in modern communication are often shorter and more aligned in structure with social media interactions, there are additional challenges to the detection task. For example, Hassan et al. [21] note that "...Twitter harbors a lot of noise, including spam, the short colloquial communication style adopted by users, irrelevant content, and an abundance of neutral content." More generally, noise issues in short-text online communication include: misspellings, inconsistent and repeated punctuation, use of emojis and emoticons, user tags, hashtags, URLs, and case mixing [22].

For example, consider the first entry from my primary dataset: "t-minus 10 minutes until interview timeeeee. #nervous". There are differences from formal style in terms of sentence length and general content, including the lack of capitalization, misspelling of time, and use of a hashtag to self-label the tweet. Considerations such as these are important aspects of context in typical short-text online communication, and need to be accounted for in emotion detection, particularly when the leading available text analysis components are foundationally trained on longer, more formal texts.

Maynard et al [23] described other challenges in social media opinion mining, including relevance to a specific topic, opinion target identification, handling of negation words, contextualization, time-based volatility of opinions, and the sheer number and variety of opinions which may apply to a specific topic. Vinodhini and Chandrasekaran [24] noted that people often express the same sentiment in different ways and that summarizing opinions from multiple users is a much different task than summarizing other facts about a text, such as how many people discussed a topic.

1.3 EMDISM Approach

Having introduced both emotion detection in general text, as well as more specifically in social media, the specific challenge my research addresses is to investigate potential improvements in finer-grained emotion detection in social media text (EMDISM). I seek to improve the state of the art in granular emotion detection for multi-class sentiment analysis in social media. To address this challenge, I have proposed and investigated the potential of ensemble approaches (e.g. [25, 26, 27, 28, 29]) to improve performance in EMDISM. Ensemble approaches bring together sets of individual classifiers in order to help address tradeoffs between the individual approaches. This involves investigating both the characteristics of representative component classifiers in the EMDISM context, as well as the performance of different ensemble approaches for combining the component classifiers.

Ensembles are usually assembled from the most accurate of numerous approaches which have been examined and described for the problem of text classification. I categorize these approaches as traditional machine learning (ML) [30, 31], deep learning (DL) [32, 33], and transformer learning (TL) [34, 35] approaches. ML approaches were among the first text classification algorithms and generally take a statistical or logic-based approach to classifying text. DL approaches are those which use neural networks to process tokenized embedding layers into class predictions through backwards propagation of error correction across the layers and nodes of the network. TL approaches are typically faster and more accurate for text classification due to their self-attention mechanism which is highly effective in preserving contextual clues relevant to any specific word or phrase in a selection of text.

1.3.1 Conceptual Framework for EMDISM

As a conceptual framework for this research, I have developed a variation on the natural language processing pipeline proposed in [36] for forensic authorship attribu-

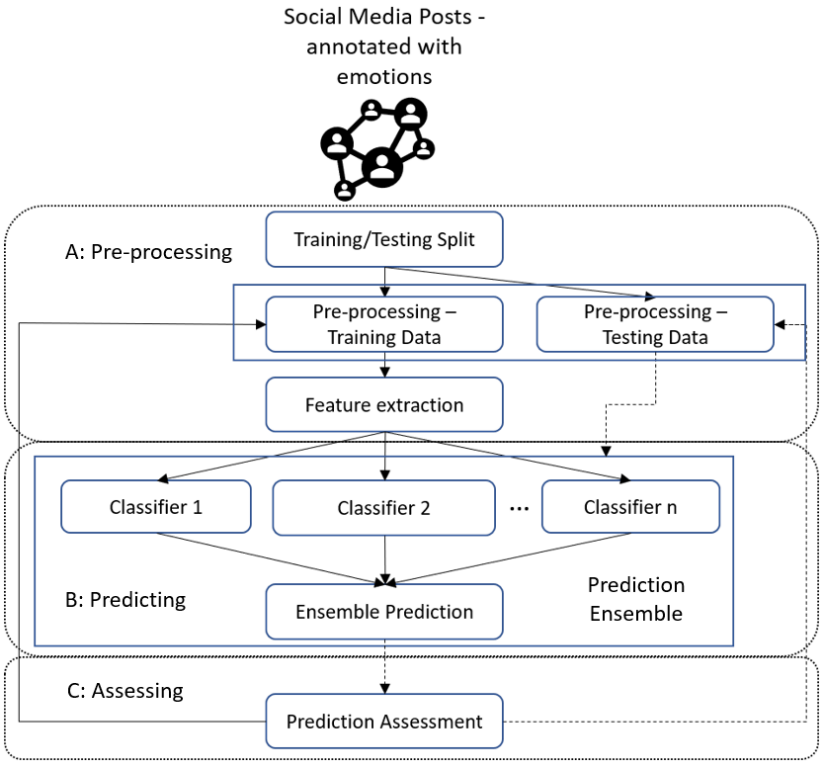

Figure 1.1: Emotion Detection Research Framework.

tion (FAA). I originally extended the Rocha et al. pipeline for my own FAA research [37], and have adapted the framework for ensemble EMDISM, as shown in Figure 1.1. The process begins with data acquisition and text pre-processing. I extract features to use for classification, then transform these features to support the primary types of classification approaches: TF-IDF tokenized vectors for machine learning (ML) algorithms and tokenized embedding vectors for deep learning (DL) and transformer learning (TL) algorithms. I then apply classification algorithms using the specified feature sets with their relevant classification algorithms. Next, I create a prediction using a decision algorithm based on the predictions of each classification algorithm. This decision algorithm can be simple or weighted voting or can be more complicated, with approaches like detecting super-classes of emotions and then assigning the predicted super-classes to classification models trained on individual classes within each super-class. The primary sequence of steps within my framework is as follows, and I

will refer to these in the detailed development of my approach:

- A — Pre-processing
- B — Classifier application
- C — Ensemble prediction

1.3.2 Research Questions

My primary research questions are aligned with the main steps of my EMDISM framework. In discussing these questions I refer to the primary EMDISM dataset employed in my research, HarnessingTwitter (HT), which is described in more detail in Sections 3.2 and 4.2.

1.3.2.1 RQ1: Impact of Transformers and Embedders

RQ1: How can we best integrate transformers, like BERT [34], ELECTRA [38], RoBERTa [35], XLM-R [39], and XLNet [40], and their embedders in our ensemble algorithms? This question is designed to investigate the tradeoffs in applying various pre-processing techniques to social media texts prior to classification. Natural language processing researchers typically describe their pre-processing approaches for their datasets, and Symeonidis, Effrosynidis, and Arampatzis [41] note a preferred order of pre-processing, which I use in pre-processing the HarnessingTwitter (HT) dataset as the first denoising step in my primary research. However, given the shift from using pre-trained embedders like Word2Vec [42] and GloVe [43] to custom embedders created for a specific dataset or pre-trained embedders included with TL algorithms, I seek to determine if substituting embedders from TL algorithms in the pre-processing pipeline for my DL algorithms may increase the accuracy of my DL algorithms. TL algorithms are also thoroughly examined in my research as part of RQ2.

1.3.2.2 RQ2: Best Component Classifiers for Ensemble EMDISM

RQ2: Which ML, DL, and TL classifiers are most accurate for use in a voting ensemble classification approach for EMDISM? In my pilot research, I compared the performance of some standard machine learning classifiers to deep learning emotion detection approaches and determined that SVM was the most accurate ML algorithm and C-LSTM was the most accurate DL algorithm with the Crowdflower dataset. In my primary research, I create new ML models and include Decision Trees, based on the research of Ranganathan et al. [44], in a new comparison of all ML, DL, and TL models using the HT dataset. I create new models with the DL algorithms using the HT dataset for comparison, and I also extend the algorithms planned for consideration for RQ2 to include TL algorithms, including BERT, ELECTRA, RoBERTa, XLM-R, and XLNet. A key part of adding TL algorithms included discovery of new questions regarding hyperparameter optimization for TL algorithms, specifically regarding learning rates and training epochs. As part of my TL research, I extend the knowledge in the field of EMDISM to recommend better hyperparameters for fine-tuning base models across these algorithms [22].

1.3.2.3 RQ3: Evaluating Ensemble Approaches for EMDISM

RQ3: Do ensemble classifiers perform better than singleton classifiers for multi-class emotion detection, and if so, which ensembles are most effective? After addressing RQ2, I utilize the most accurate classifiers to build ensemble classifiers and investigate which are more effective than singleton classification approaches. Many ensemble approaches have been described in NLP and related literature, including applications for classification [45, 46], recommender systems [26], and modern approaches to sentiment analysis and emotion detection [47, 48, 49, 50, 51]. As other researchers have demonstrated that ensemble classifiers are at least comparable in accuracy to singleton classifiers, I develop and assess numerous ensemble clas-

sification approaches to evaluate whether they are better able to detect fine-grained emotions, using voting ensembles, cascading, or switching ensembles. Whereas some researchers have studied a few limited ensembles for sentiment analysis and emotion detection, I conduct an extensive examination of numerous specific ensembles combining traditional ML, DL, and TL techniques, including four categories of ensembles across twenty-one different algorithms, to assess which types of ensembles and ensemble components are most effective in answering RQ3.

1.4 Contributions

My contributions to the field of EMDISM are as follows:

- **Pre-processing:** I create and assess DL models built using TL embedders and compare these to a custom embedder built specifically on the HT dataset. My results demonstrate the custom embedder yields more accurate results than the TL embedders as the input layer of DL algorithms.

- **Transformer learning - hyperparameters:** I conduct a rigorous examination of TL hyperparameters to determine which provide the most accurate results without overfitting. My research presents recommendations for more effective combinations of learning rates and fine-tuning epochs in the EMDISM domain, beyond the general guidance reported in the literature for the TL approaches.

- **Transformer learning - ensembles:** I create and assess multi-TL ensemble classifiers and report my findings, which indicate ensembles of TLs can provide more accurate predictions than singleton classifiers. My results demonstrate ensembles of TLs can provide more accurate predictions than singleton TL classifiers.

- **Multi-discipline ensembles:** I conduct and report a rigorous assessment of fifteen different individual classifiers across three different classifier disciplines

(ML, DL, and TL) and provide results for individual accuracy as well as for ensembles leveraging all components and combinations thereof, including ensembles combining ML and TL classifiers. Results show that ensembles combining TLs and the most accurate ML classifier are more accurate than the most accurate singleton classifiers.

- **Ensembles:** I conduct and report a rigorous assessment of EMDISM ensembles including up to 15 different component classifiers, in 21 unique combinations, across 4 ensemble categories. I provide experimental results proving that more than half of my ensembles provide more accurate results than the best singleton classifier, with better precision, recall, and f-measure.

To summarize, my research applies pre-processing techniques in novel ways to assess the value of using pre-trained embedders from TL algorithms in the input layer of DL algorithms. I compare ML, DL, and TL algorithms and present research in hyperparameter optimization to establish new baselines for using TL algorithms for EMDISM. Finally, I present extensive, rigorous experimental results for four categories of ensembles with twenty-one different algorithms to support the accuracy and effectiveness of using ensembles for EMDISM.

CHAPTER 2: RELATED WORK

Emotion detection in social media relies on aspects of natural language processing and sentiment analysis. This chapter provides an overview of background research across all of these areas. First, I discuss research relevant to RQ1, wherein I examine the application of pre-trained embedders from TL algorithms in the embedding layer of DL algorithms. Specifically, I provide an overview of common tasks, discuss feature selection, and how to de-noise datasets for use with ML, DL, and TL classifiers. Second, I discuss research relevant to RQ2, in which I investigate which classifiers are the best candidates to include in ensemble approaches. I provide background on ML, DL, and TL classifiers and discuss some of the key differences between approaches in order to understand how one classifier's strengths may offset an other's weaknesses. Third, I discuss research related to RQ3, wherein I develop and evaluate different types of ensembles for EMDISM. I review related work on ensembles, how they are commonly constructed, and how they have been applied to sentiment analysis and emotion detection tasks.

Discussion of related work begins with an overview of common tasks in NLP research (2.1) and data pre-processing (2.2). Sentiment analysis and emotion detection classification can be generally partitioned into traditional machine learning approaches (older approaches) which typically apply some variation of statistical [30] or logic-based assessment [31], in contrast to neural networks and deep learning approaches for classification (newer approaches) [32, 33] which convert text to input layers in neural networks of varying composition and complexity using back-propagation for training to create classification models, and finally transformer learning approaches (newest approaches) which are specialized neural networks using combinations of

encoders, decoders, feed-forward layers that also incorporate crucial *self-attention* mechanisms, which combine the encoding and decoding layers to create classification models [34]. Section 2.3 presents background on traditional ML classifiers, Section 2.4 covers background on DL classifiers, and Section 2.5 discusses background on TL classifiers. These sections provide an overview of common approaches to EMDISM, from the nascent days of ML NLP research, through newer applications of DL approaches, and finally state-of-the-art TL approaches. Understanding the similarities and differences of these approaches to NLP classification and emotion detection provides insight into how a strength of one approach may offset a weakness of another approach, and also supports the development of ensembles. Understanding how each model processes NLP data, creates and stores models, and generates predictions is key to creating ensembles that work well together. My research includes assessment of representative algorithms from ML, DL, and TL and their inclusion and efficacy in various iterations of my ensemble EMDISM research. Section 2.6 discusses related work on ensemble approaches to classification problems, sentiment analysis, and emotion detection, and also identifies commonalities between approaches.

2.1 Common Tasks in NLP Research

In this section, I discuss common tasks in NLP research and discuss how each step aligns to my research framework. NLP research for text usually requires selecting some subset of common tasks within several steps in the research process, commonly referred to as an NLP pipeline [36, 52], including:

1. **Data acquisition** - either acquiring common datasets from online resources or assembling new datasets through some form of programmatic or manual web-scraping and labeling. I identified and employed existing datasets for my research, as discussed in Chapter 3, rather than creating new datasets. The data acquisition step feeds data to Framework Section A in Figure 1.1.

2. **Pre-processing** - applying techniques to reduce noise in the dataset and to prepare the data for use in classification algorithms. Randomly partitioning datasets into testing and training sets is also considered a pre-processing task. This task aligns with Section A of my research framework in Figure 1.1 and RQ1, wherein I examine using TL embedders in the input layer of DL algorithms.

3. **Training classifiers** - selecting state of the art algorithms to train with a subset of the data, and then training those classifiers using training data. This task aligns with Section B of my research framework in Figure 1.1. It applies to RQ2 for initial development and assessment of classifiers for inclusion in my ensembles. It also applies to RQ3, in that cascading and switching ensembles require training new classifiers to predict super-classes and sub-classes within the super-classes.

4. **Predicting results** - applying trained classifiers to testing data to generate predictions about the testing data. This task aligns with Section B of my research framework in Figure 1.1. It aligns with RQ2, in that the initial classifiers create predictions for assessment in the next task in order to assess whether they should be included in my ensembles, and the singleton prediction results are maintained for comparison to ensembles from RQ3. It also aligns with RQ3 in that simple voting, weighted voting, cascading, and cascading/switching ensembles all must generate predictions for assessment of ensemble efficacy.

5. **Assessing predictions** - analyzing the predictions to produce useful metrics and determining the most effective classification algorithms. This task aligns with Section C of my research framework in Figure 1.1 and applies to both RQ2 and RQ3 as predictions from both singleton and ensemble classifiers must be compared to assess which have the best accuracy, precision, recall, and f-measure.

Variations on these common tasks are typical throughout the NLP literature. I have adapted my previously published approach [37], in which I extended the framework of Rocha et al. [36] to investigation of forensic authorship attribution. I discuss the pre-processing phase (Framework A) of these common tasks in 2.2. The training and predicting tasks (Framework B) for singleton classifiers are discussed in 2.3, 2.4, and 2.5, and the application of ensemble classification approaches are discussed in 2.6. The assessing process for my research is described in 2.7 (Framework C).

2.2 Pre-processing NLP Data

The research in this section is relevant to RQ1, wherein I assess novel applications of pre-trained embedders for inclusion in my DL algorithms, and to Section A of my research framework (Figure 1.1). Collobert et al. [53] described several pre-processing tasks in their research (Section A in Figure 1.1), including "Part-Of-Speech tagging (POS), chunking (CHUNK), Named Entity Recognition (NER) and Semantic Role Labeling (SRL)." Whereas these elements are common within the NLP research space, Smetanin's more recent work [54] provided a fine-grained listing of the pre-processing methods used in their research, including: 1) replacing URLs, emails, the date and time, usernames, percentages, currencies and numbers with tags, 2) annotating repeated, censored, elongated, and capitalized terms with tags, 3) correcting elongated words via corpus, 4) hashtag and contraction unpacking, and 5) reducing the variety of emotions via dictionary corpus.

The Smetanin approach is more fine-grained than Collobert et al., but Symeonidis, Effrosynidis, and Arampatzis went even further [41], evaluating the interactivity of seventeen pre-processing techniques and their net effect on classification accuracy, as well as recommending an order for pre-processing. Their recommended order was designed to reduce conflicts between approaches, and I adopted their recommendations as part of the pre-processing steps in my own research. For example, the logical approach to pre-processing is to remove noise from recognizable tokens, like hyperlinks,

user tags, and hashtags, before more basic processing like punctuation processing, contraction replacement, and lemmatization, as the act of pre-processing these discrete tokens could introduce additional noise, whereas removing discrete tokens first reduces the noise in the dataset before more general cleansing and reduces the overall quantity of data requiring further pre-processing. In general, they recommended lemmatization, replacing repeated punctuation and contractions, and removing numbers from social media data prior to classification. They also recommended avoiding punctuation removal, markup of capitalized words, slang replacement, spelling correction, and replacing emotion negations with antonyms. For more specific applications, Symeonidis et al. note [41]:

> *"Depending on the classifier, the results vary, and if we combine these techniques we may get different results. A winning combination if someone wants to preprocess text for a classic machine learning Sentiment Analysis is: replace URLs and user mentions, replace Contractions, remove Numbers, replace repetitions of punctuation, and lemmatizing. If we choose a Neural Network approach the above combination, without the technique of removing numbers, is the best."*

In the Pre-processing section of Chapter 4, I describe the pre-processing techniques I applied to my available research datasets, which align closely with the techniques described above. In subsequent sections, I describe various classification methods used in previous research in support of my research framework, Section B in Figure 1.1 and RQ2.

2.3 Machine Learning Sentiment Classification

Next I describe research into traditional machine learning classifiers. A wide variety of machine learning classification algorithms exist, and the broad categories below include many variants. The methods I cover below are the most commonly applied

to the tasks of sentiment analysis and emotion detection. The research in this section is relevant to RQ2, wherein I assess different classification algorithms for inclusion in my ensemble algorithms, and to Section B of my research framework (Figure 1.1).

In 1993, Quinlan introduced decision trees [55], one of the earliest machine learning algorithms for classification. Decision trees classification has been adapted and applied to numerous classification problems, and they are still in common usage today and applied specifically to the domain of social media emotion detection [44]. Decision trees (DT) are a supervised learning method which work by building classification structures which partition data into subsets of similar values. These structures resemble trees and they can be easily rendered to graphs to make them easier for humans to understand. In a decision tree, each branch represents a decision point where information gain is maximized, and each leaf on the tree is a class label. In a Twitter dataset of 200,000 samples with five emotions automatically labeled using a combination of the NRC lexicons for emoticons, hashtags, and word-sense annotation, Ranganathan et al. reported classification accuracies with multiple DT implementations of 88% to 96% [44]. Larger trees tend to be more accurate but can suffer from overfitting. In considering decision trees for ensembles, it may also be that DT classifiers based on smaller sample sizes may be more accurate and may offer better accuracy for the least represented classes in unbalanced datasets.

Naïve Bayes (NB) [56] is a popular classification algorithm which is based on the idea that a given term's probability of appearing in a text does not depend on its position or context [57]. This means that the emotion of a selected text can be predicted by selecting the highest probability that a term will belong to one class among many. Whereas NB is a popular classification algorithm, it tends to perform with less accuracy than other machine learning classifiers, especially SVM when used in the domains of sentiment analysis and emotion detection [48, 51].

Support Vector Machine (SVM) classifiers [58] attempt to define a multi-dimensional

hyperplane that segregates large vectors of data into discrete clusters with the goal of maximizing the margins between clusters. This approach maps well to the sparse input vectors resulting from mapping text tokens into input vectors in classifiers, and as such SVM is a popular and accurate algorithm for sentiment polarity detection [48, 51]. Many different kernels have been developed for use with SVM, including linear kernels and sigmoid kernels, among others, and SVM has been extended to include a one-versus-rest (OneVR) approach to multi-class categorization [59]. In OneVR, the probability that a text will belong to one class is compared to the probability that it will belong to all the other classes together. This process is repeated to compare each class against the group of all other classes, and the final selection is determined by the highest probability for any one class.

K-nearest neighbor (KNN) is a clustering approach [60] wherein a variety of distance measures may be used to determine the distance between a given text sample's vector representation and its nearest neighbors, including cosine similarity, the Jaccard coefficient, Manhattan, Minkowski, and Hamming Distances, and others. The variety of distance measures makes KNN classification highly flexible, and it also works well for multi-class problems. The approach also benefits from improved classification as datasets grow larger. As such, KNN has been successfully applied to the domain of sentiment analysis [61, 62].

Maximum entropy (ME) uses labeled training data to create an estimate of the word count distribution for given classes within a document. Text samples are represented as word counts, and the word counts are used to assign the text sample to a class based on the probability estimated from the training data. Maximum entropy was first applied to text classification by Nigam, Lafferty, and McCallum [63], who specifically compared the performance of ME to NB and found that the method outperformed Naive Bayes in some cases for text classification of a paper's creator. Rao et al. [64] applied topic-level maximum entropy (TME) to achieve a top accuracy of

approximately 86%. One interesting feature of their approach is that they combined the mapping of topics, emotion labels, and emotion valence in their TME algorithm to improve their results. Xie et al. [65] applied ME to the task of classifying words by negative or positive emotion and achieved comparable accuracy to LSTM, NB, and SVM.

2.4 Deep Learning Emotion Detection

The research in this section is relevant to RQ2, wherein I assess different classification algorithms for inclusion in my ensemble algorithms, and to Section B of my research framework (Figure 1.1). I begin by summarizing the core concepts of deep learning, following the survey overview in [66]. Neural networks are assembled as layers of decision nodes, with an input layer, where source data enters the network to help train it, one or many hidden layers which contain decision neurons with state and activation, and an output layer. In the output layer, predictions are evaluated for error against the original input, and weighted adjustments are propagated backwards through the layers to adjust the decision thresholds at each of the decision nodes (called neurons). The network is trained iteratively through cycles (called epochs) in which input is processed, evaluated, and small error corrections are propagated backwards to minimize the prediction error. See Figure 2.1 for a representation of a simple neural network.

The activation function is important for the proper performance of a neural network. Sigmoid activation is common for binary classification, softmax for multi-class classification, and other activation functions include the hyperbolic tangent function (tanh) and the rectified linear function (ReLU). Many hidden layers are possible, each with different activation functions. For multi-class classification, the neural network must correct errors based on validated categorical accuracy, instead of the accuracy of any one node.

A neural network's input layer is not well-suited for the simple input of each word

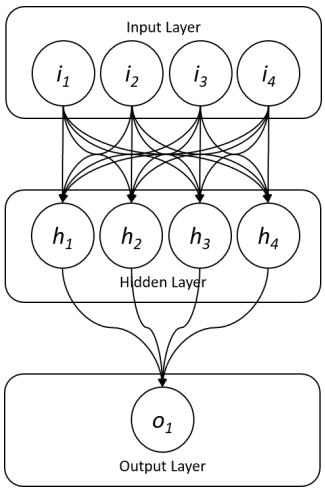

Figure 2.1: Simple Neural Network.

of a sentence into a node. For emotion detection from text, the input text (a tweet, phrase, sentence, n-gram, or character n-gram) is converted from text into a vector of numbers using word embedding [53]. Zhang et al. [66] note that "Word embedding is a technique for language modelling and feature learning, which transforms words in a vocabulary to vectors of continuous real numbers." Word2Vec [42] and GloVe [43] are two popular embedding functions. Word2Vec uses a Skip-Gram (SG) model or Continuous Bag-of-Words (CBOW) model to create word embeddings from text input and saves them for reuse as source input for deep learning models. SG and CBOW take opposite approaches to context, with SG working from the word out to context words and CBOW working from the context inward. GloVe creates embeddings by training on a sparse word co-occurrence matrix focusing on non-zero entries. Embeddings can also be manually generated through use of a tokenizer, like the one included with the Keras library for deep learning [67], and embedders from more recent libraries may also be used, including those based on Google's WordPiece [68], like the ones included in the HuggingFace Transformer libraries [69].

More complex neural networks have also been developed. The design in Figure 2.1 was extended to create the Convolutional Neural Network (CNN) [70], originally dedicated to the task of image recognition. CNNs establish smaller and smaller filters

on sections of an image while mapping the features from the larger selection to each smaller filter. The last layer of a CNN is usually a softmax layer which maps to the categories of the classification task. This approach to image recognition conceptually maps to text classification in that the context of the surrounding nodes is important to the classification task. Consider the phrase, "I can't say that I love this movie." If one segments this phrase to just "...I love this movie," the emotional classification changes.

Recurrent Neural Networks (RNNs) employ an internal memory of previous steps. RNNs are well-suited to the task of emotion detection, as the output of a step for each word is applied to the next word in the sequence, which captures the context of the entire sequence of words. To help address the vanishing or exploding gradient problem limiting RNNs to memory of a small number of steps, researchers developed variants of RNNs, including Bidirectional RNNs (B-RNN) [71] and Long Short Term Memory (LSTM). B-RNNs are designed to consider not only the previous steps in a sequence but also the next steps in a sequence. The design of a B-RNN uses two stacked RNNs (an ensemble of DL classifiers), wherein one RNN processes the input in the original order, and the other reverses the order of the input. The hidden state of each RNN is then used to compute the output.

LSTM is a type of neural network that bypasses the vanishing/exploding gradient problem with RNN memory [72]. LSTM uses 4 layers per time step to complete its computations, including a forget gate, an input gate, a hidden memory layer, and an output gate. Other variants of LSTM include Tree-structured LSTM [73] and the Gated Recurrent Unit (GRU) [33]. GRU simplifies LSTM by combining the forget and input gates and merging the hidden and cell states. Bidirectional LSTM (BiLSTM) and bidirectional GRU (BiGRU) simply add a bidirectional layer to each algorithm.

Convolutional LSTM (C-LSTM) adapts the LSTM model to add memory of the class to each gate in the LSTM layer. Ghosh et al. [32] applied this approach in a

topic modeling context and improved upon the performance of the LSTM classifier in that application by approximately 20%. Given the high number of topics contained within a text dataset, this performance boost makes C-LSTM a good candidate for application to the problem of fine-grained emotion detection.

2.5 Transformer Emotion Detection

The research in this section is relevant to RQ2, wherein I assess different classification algorithms for inclusion in my ensemble algorithms, and to Section B of my research framework (Figure 1.1). Transformers were first proposed by Vaswani et al. [74] as a new type of neural network designed to replace the sequential computational models of then state-of-the-art deep learning networks with faster, parallelizable networks using attention mechanisms to replace the recurrent and convolutional architectures common in approaches like LSTM, C-LSTM, CNN, and RNN. Figure 2.2 shows the Vaswani et al. representation of their transformer architecture. The attention mechanism is the most important concept in transformers as it maps the relationships between words in a sentence with differential weights to determine which provide the most context for any given word in the sentence. Consider the sentence, "The car would not fit in the garage because *it* was too small." A human reading this sentence would most likely infer from their inherent contextual knowledge of cars and garages that the word "it" in this case referenced the garage, not the car; however, a neural network without a self-attention mechanism would likely have trouble mapping out which subject in the sentence was the target of the word "it." A transformer classifier, on the other hand, would map these relationships and others like it in the training data to develop a weighting indicating that the garage was the most likely target of the "it" reference given other contextual clues in the sentence.

Devlin et al. [34] created BERT (Bidirectional Encoder Representations from Transformers) to capture the bidirectional contextual information inherent in text. BERT was pre-trained using the 800M words of the BooksCorpus [75] and 2.5B words

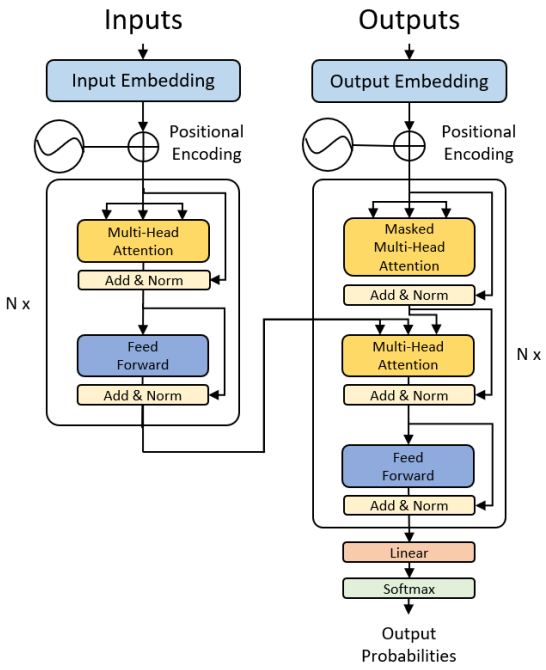

Figure 2.2: Architecture of a Transformer neural network [74].

of English Wikipedia. BERT was developed using a 30K token vocabulary based on WordPiece embeddings [68]. Part 1 of BERT's pre-training was unsupervised training, conducted by randomly masking 15% of the input tokens and then predicting the masked tokens – a so-called masked language model (MLM) approach. Part 2 of pre-training was meant to enable an understanding of sentence relationships using the unsupervised training of a next sentence prediction task. BERT generated pre-trained embeddings based on the sum of token, segmentation, and position embeddings, and thus captures information from each pre-training task in the final embedding vector. BERT was trained with batches of 256 tokens. The original BERT approach showed substantial improvement on the General Language Understanding Evaluation (GLUE) [76] score by 7.7% absolute improvement, and 91.6% for binary (positive/negative) sentiment prediction when combined with BiLSTM. [22]

Liu et al. [35] extended the concepts of BERT transformers in RoBERTa (Robustly Optimized BERT Pretraining Approach). RoBERTa was trained with the CC-NEWS dataset, a derivation compiled from Nagel's CommonCrawl News dataset [77]. The

RoBERTa approach matched or improved on BERT results in 6 of the 9 GLUE tasks. Liu et al. increased several training hyperparameters for BERT, including the length of training and the batch sizes used. RoBERTa was trained with maximum batch sizes of 512. For ablation testing, Liu et al. removed the next sentence prediction task, adapted the masking pattern dynamically, and trained on longer sequences. In evaluating RoBERTa, the team reported a best accuracy in binary sentiment analysis (the SST-2 task from GLUE) of 92.9%.

Yang et al. [40] compared their XLNet algorithm to BERT, and differentiated their approach from BERT in several key ways. The XLNet team indicated that BERT introduces noise in its pretraining approach by adding artificial symbols for masks and separators into the data and also assumes each predicted token is independent of another token considering the presence of masked tokens. XLNet's training method considers permutations of factorization orders to capture the bidirectional context of tokens within text and maximize the logarithmic likelihood of a token sequence in regards to the permutations. As an autoregressive approach to pre-training, XLNet uses the product rule to factor join probabilities of predicted tokens, thus avoiding the token/mask independence discrepancy in BERT. Finally, XLNet further differentiates its approach from BERT in not introducing tagging noise to the dataset. XLNet was trained on many of the same or similar datasets as BERT and RoBERTa, including CommonCrawl, BooksCorpus, and English Wikipedia, while also training with the ClueWeb 2012-B (extended from [78]) and the Giga5 [79] datasets. I note that Yang et al. intentionally filtered their training data to remove short, low-quality articles, arguably representative of social media type data. XLNet was trained with 512 token training sequences and, like RoBERTa, dropped the next sentence prediction task when compared to BERT. XLNet reported 94.4% accuracy in the SST-2 binary sentiment prediction task, higher than both BERT and RoBERTa.

Cross-lingual language models (XLM) [80] were developed to extend the success

of pretraining approaches like BERT to multiple languages. XLM was trained using the XNLI dataset [81] with 7500 human-annotated samples from 15 languages. XLM differs from BERT by considering text streams truncated at 256 tokens, composed of an arbitrary number of sentences, instead of sentence pairs. XLM attempts to address the imbalance between frequent and rare tokens by sampling the tokens in a stream based on their multinomially distributed weight in proportion to the square root of their inverted frequencies. For translation language modeling, XLM extended the MLM approach to consider pairs of parallel translated sentences, thus generating predictions based on contextual clues from either language.

XLM-RoBERTa (XLM-R), developed by Conneau et al. [39], applied concepts presented by both BERT and XLM. Specifically, XLM-R was trained with MLM using monolingual sample streams and a larger vocabulary than BERT, with 250K tokens, compared to 30K with BERT. 100 different languages were sampled with the same distribution used in XLM, with $\alpha = 0.3$ instead of $\alpha = 0.5$, and without language embeddings. A clean derivation of the CommonCrawl Corpus was used in pretraining XLM-R, with one English version and twelve versions inclusive of other languages. Conneau et al. reported 95.0% accuracy on the SST binary sentiment classification task.

Clark et al. [38] developed ELECTRA (Efficiently Learning an Encoder that Classifies Token Replacements Accurately) to offset a perceived weakness of BERT, namely, that BERT has an imbalance caused by the introduction of masking tokens during the pre-training phase, but not during the fine-tuning phase of training. While the pre-training corpus is not specifically noted in [38], the appendix notes running an MLM comparison between ELECTRA-Base and BERT-Base using a combined Wikipedia and BooksCorpus dataset.

BERT was actually more accurate in the MLM comparison, with a 77.9% accuracy compared to ELECTRA's 75.5%. ELECTRA attempts to offset the BERT imbal-

ance between training and fine-tuning by replacing some tokens with samples from a proposed distribution created by a small MLM. Pre-training is then conducted to predict every token, whether an original token or a replaced token. This differs from BERT, wherein pre-training is only applied to the masked 15% of all tokens, and enables ELECTRA to be pre-trained faster. ELECTRA reported a top SST accuracy between 89.1% to 96.7%, with accuracy variations determined by the fine-tuning datasets used and the duration of training.

2.5.1 Hyperparameter Optimization

Hyperparameter optimization refers to the process whereby researchers attempt to identify the optimal set of parameters for use with training classification models to develop models with sufficient accuracy while avoiding overfitting the model to a specific dataset. Elshawi et al. [82] note that the process of hyperparameter optimization is one of the key challenges in developing accurate models specific to a given domain. Murray et al. [83] examined auto-sizing various components of the transformer architecture, including gradients, attention heads, and feed forward networks, in order to streamline the hyperparameter optimization process. Yang and Shami [84] completed a comprehensive survey of hyperparameter optimization approaches for traditional machine learning and deep learning models and identified the learning rate, dropout rate, batch size, and number of epochs as key hyperparameters in need of tuning for deep learning algorithms. They describe numerous optimization approaches, including trial and error [85], grid search [86], and random search [87], as well as various libraries designed to assist in their implementation.

It is notable that while each of the various transformer algorithms I considered provided some insight into how their models were developed, details on hyperparameter selection explored were limited. For example, in [34], Appendix A.3 notes a constant dropout of 0.1, batch sizes of 16 or 32, Adam learning rates of 5e-5, 3e-5, or 2e-5, and either 2, 3, or 4 learning epochs, stating that these values were found to "...work well

across all tasks...". In addition, it is noted that datasets with more than 100k training samples were less sensitive to hyperparameter choices, and that fine-tuning was generally fast enough that it was reasonable to simply iterate through the different hyperparameter options and pick the model with the best results. I interpret this as a recommendation that a basic grid search [86] approach is expected to be sufficient for hyperparameter optimization, so I applied a grid search approach to hyperparameter optimization for the transformer models I developed.

2.6 Ensemble Text Classification

The research in this section is relevant to RQ2, wherein I assess different classification algorithms for inclusion in my ensemble algorithms, and to Section B of my research framework (Figure 1.1). Ensemble approaches to machine learning problems have a long history, from their earliest references [25, 46, 88, 45], to an overview of the many ways to create ensembles [26], to ensembles specifically designed to classify sentiments and emotions [89, 90, 50, 51, 64, 91, 92, 93, 47, 49, 94, 95, 96, 97, 98, 99, 100, 101, 27, 28, 29]. In my research, the ultimate goal is the analysis of ensembles combining ML, DL, and TL algorithms to offset the shortcomings of various approaches with the different coverage afforded by other approaches. For example, some algorithms seem to perform better with smaller sample sizes and/or imbalanced data, so I investigate their performance to determine the potential for ensemble inclusion. Research into overall ensemble design is included in this section to provide insight into the utility and logic behind ensembles, and to assess how they may best be assembled.

2.6.1 Foundation of Ensembles

In 1990, Hansen and Salamon [25] proposed ensembles of neural networks to further reduce the residual error in trained neural networks. They noted that their approach trained multiple neural networks to classify data based on the same dataset and

then applied a simple consensus voting scheme to predict the classification of the data. They also noted two key points for ensemble classifiers: (1) "The conclusion is that the ensemble can be far less fallible than any one network," and (2) "An accurate model of performance with individual proficiencies can provide a criterion for screening the networks for membership in the ensemble." To simplify their second point, in a majority voting ensemble, even with only 3 classifiers in the ensemble, in order for a prediction to be wrong, 2 of the 3 classifiers must provide incorrect predictions. Furthermore, the second key point implies that an effective approach for building ensembles is to individually assess the accuracy of candidate classifiers and then select the most accurate among those classifiers for use in ensembles.

Boosting, first proposed for machine learning by Schapire in 1990 [46], is the process used to turn a weak learning algorithm into a strong learning algorithm through iterative training and adjusting weights to focus on incorrectly classified training data. When the training begins, all base learners are weighted the same. In the next iterations, incorrectly classified outputs are increased in weight so that more attention is given to the incorrectly classified data. This step is repeated until either a higher accuracy is achieved or the improvement in accuracy reaches a plateau. AdaBoost [88], one of the first popular boosting algorithms, uses a weighted voting ensemble and is still popular today.

Bootstrap aggregating (Bagging) predictors were ensemble approaches first proposed by Breiman in 1996 [45]. In bagging, multiple randomly selected bootstrapped replicas of data used for training are drawn with replacement from the training data, and different base learners are trained on the bootstrapped replicas. A simple majority vote is used to select the prediction with bagging, as is common in ensemble classifiers from multiple domains, including machine learning classifiers [88], recommender systems [26], sentiment analysis [48, 93], and emotion detection [95].

2.6.2 Overview of Ensemble Approaches

Robin Burke characterized hybrid (ensemble) approaches in the context of recommender systems [26]. Burke noted several general types of ensemble structures, including:

- **Weighted** - Several systems present their votes and make recommendations based on the most votes for any given prediction. The most trusted system's vote has more weight than others.
- **Mixed** - Recommendations from each approach are presented to the user concurrently.
- **Cascading** - Recommendations from one algorithm are used to refine the recommendations from the next algorithm.
- **Switching** - Conditions within the data dictate switching from one technique to another dynamically during processing.
- **Feature combination** - Features from different recommendation sources are combined into one source.
- **Feature augmentation** - Output from one recommender is used as the input to the next.
- **Meta-level** - The trained model from one system is used as input for the next.

Burke's ensemble structures were presented in the context of recommender systems, but the general categories above are useful in understanding the flexibility inherent in building ensembles more broadly, and his categories suggest useful ways to assemble the disparate pieces of an ensemble into an aggregate approach which is more effective than each part can achieve alone. I adopt Burke's characterization as a guide for the ensemble structures developed in this research.

2.6.3 Ensembles for Sentiment Analysis and Emotion Detection

Cao and Zukerman [89] evaluated an ensemble classifier combining supervised and unsupervised approaches to "Multi-way Sentiment Analysis," which I refer to as emotion detection. In their research, the ensemble was comprised of a lexicon approach, which built an estimation of sentiment by building from word, to phrase, to sentence, and finally review sentiment, and two unsupervised classifiers, consisting of NB and another ensemble decision tree and SVM-based classifier called minimum cost spanning tree SVM (MCST-SVM). In the complex MCST-SVM ensemble, Bickerstaffe and Zukerman [90] were applying Burke's concepts of feature augmentation and cascading, as each decision node in the decision tree of MCST-SVM was decided through the application of SVM to assign samples to their left and right sub-classes. Cao and Zukerman found that their ensemble performed comparably to a pure Naïve Bayes approach for the task of predicting star ratings from 1 to 5 stars, achieving best accuracy scores of approximately 70%-75%. As Cao and Zukerman built their ensemble approach on top of another ensemble, the ubiquity of ensembles in sentiment analysis and emotion detection is obvious, but these examples are only two among many. Duppada, Jain, and Hiray [50] used stacked (cascading) ensembles of XG Boost and Random Forest classifiers to assess the intensity of emotions and to detect 4 classes of emotions (anger, fear, joy, and sadness) with 83.6% accuracy. Burnap et al. [51] used random forests and maximum probability classifiers to reach an F-measure of 0.728 in multi-class detection for text specifically relating to suicidal ideation. Rao et al. [64] applied a feature aggregation ensemble approach in their TME algorithm to achieve best accuracy scores of approximately 86%. Oussous, Lahcen, and Belfkih [48] tested an ensemble including NB, SVM, and ME in a simple majority voting ensemble, and achieved accuracies in the range of 84% to 89% for Arabic sentiment polarity. These examples demonstrate the value in creating ensembles which include techniques with varying approaches to classification problems. Here, the ensembles

combine lexical and traditional machine learning classifiers to render predictions, concepts which I apply in my ensemble algorithms. Note also that the best accuracies reported here are 75% for sentiment valence (star ratings), 83.6% for classifying four discrete emotions, and 86% for sentiment polarity.

Liaw and Wiener [91] first applied random forest for text classification. A random forest (RF) is essentially an ensemble of decision trees, wherein the decision at each node is not simply the best among the available predictors at a node, but are instead a random sampling of the best predictors at the node. The random sampling in random forests works well to offset the overfitting problem, wherein a classifier will be very effective at predictions on the original dataset but perform poorly when applied to other datasets in the same domain. Fang and Zhan [92] compared the performance of the random forest ensemble against SVM and NB for categorizing sentiment on 5-star rating scales and sentiment polarity, and found that RF outperformed SVM and NB. Random forests are an example of creating an ensemble of the same classifier to create an approach that works better than a single application of the algorithm. This approach is reflected in my application of these concepts where I repeatedly apply differently tuned models of BERT tailored to detect super-classes and sub-classes in cascading ensembles. The comparison of Random Forests to SVM and NB is reflected in my comparison of numerous classifiers to singleton classifiers to determine if my ensembles are more accurate than singleton classifiers.

With the recent surge in research applying deep learning and transformer algorithms to the problem of sentiment analysis and emotion classification, some researchers have created ensembles of deep learning algorithms. For example, Baziotis et al. [93] created an ensemble of two RNNs, specifically Bi-directional LSTMs, each with separate tasks, to detect ironic Tweets for the SemEval 2018 Task 3 competition [102]. One model captured semantic information of the tweet at the word-level and the second captured semantic information at the character-level. Whereas they

used pre-trained word embeddings, they created the character-level embeddings from scratch. They evaluated two different ensemble methods, majority voting and unweighted averaging, and the unweighted averaging approach was most accurate for detecting irony, with approximately 78.5% accuracy. The combination of multiple deep learning approaches is reflected in my ensemble algorithms.

The approach applied by Araque et al. could be described as building an ensemble of ensembles [47]. This approach evaluated two overall ensemble approaches, meta-level and switching. At the baseline level, the researchers applied 6 classifiers to the dataset, and many of these 6 classifiers were in themselves ensembles. Within the ensembles, NB, ME, SVM, RNN, and lexicon-based approaches were all present. Then the output of the 6 classifiers were unified through rule-based switching for one evaluation case, and were unified at the meta-level as input for an RF classifier for the other case. In this approach, the ensemble approaches performed better than their component approaches 50% of the time (3 out of 6 comparisons), with accuracy ranging from 85% to 94%. This example is reflected in my application of switching techniques in my ensemble algorithms. Note that sentiment polarity accuracy in this case is as high as 94%.

Perikos and Hatzilygeroudis [49] built a cascading ensemble classifier wherein they combined NB, maximum entropy, and a knowledge-based tool for sentence structure analysis with the ISEAR and Affective Text datasets. This ensemble determined whether a sentence was emotional or neutral, then further classified only the emotional text in the next step as positive or negative. In applying this ensemble to each dataset, they achieved higher accuracy scores with the ensemble approach. Da Silva et al. [94] combined NB, SVM, Random Forest, and a knowledge-based lexicon for binary emotion prediction, and this ensemble was most accurate in its application to every dataset. Xia et al. [95] used an ensemble of NB, maximum entropy and SVM for emotion polarity classification with 85.58% accuracy. Of the ensemble approaches

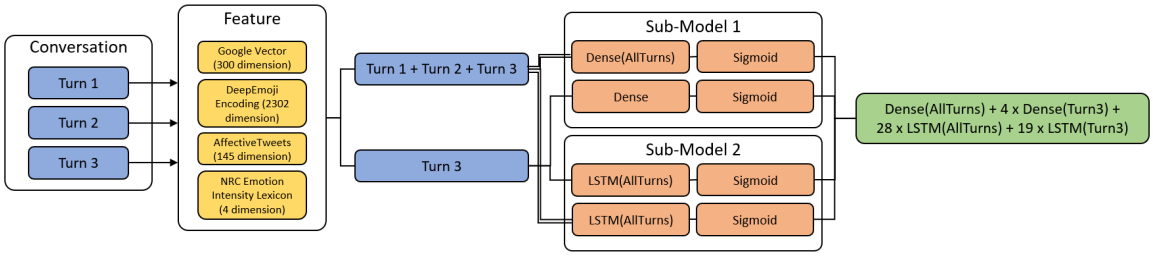

Figure 2.3: EmoDet ensemble architecture used by Al-Omari, Abdullah, and Bassam for SemEval-2019, Task 3.

I have found thus far, only the team of Araque et al. included any combination of ML and DL classifiers, with a heavy weighting (5:1) of ML to DL classifiers. Both Araque et al. and Xia et. al. confirm that ensembles can yield better performance when complex feature sets are involved. Finally, Wang et al. [96] applied bagging, boosting, and random subspace [103] (a variant of bagging wherein the feature space instead of the instance space is modified) in the field of sentiment classification and found that an ensemble of random subspace with SVM had the best average accuracy for classifying binary sentiments, which strongly supported additional research into ensemble classifiers for emotion detection. These examples inform my inclusion of cross-domain approaches combining variations of ML, DL, and TL algorithms, as shown in numerous ensembles I create which include decision trees.

Al-Omari, Abdullah, and Bassam [97] applied an ensemble called EmoDet, combining Word2Vec embeddings, a fully connected neural network architecture, and LSTM to the task of detecting Happy, Sad, Angry, and Other classes of emotions from context in social media conversations, Task 3 of SemEval-2019 [98]. Figure 2.3 shows the architecture of the EmoDet solution. EmoDet achieved an f-measure score of 0.67 for the task, better than the baseline provided by the SemEval-2019 organizers. This example is reflected in my research, wherein I examine using TL embedders as the input layer to five different DL algorithms and compare the results to the accuracy obtained using an embedder customized to my primary research dataset.

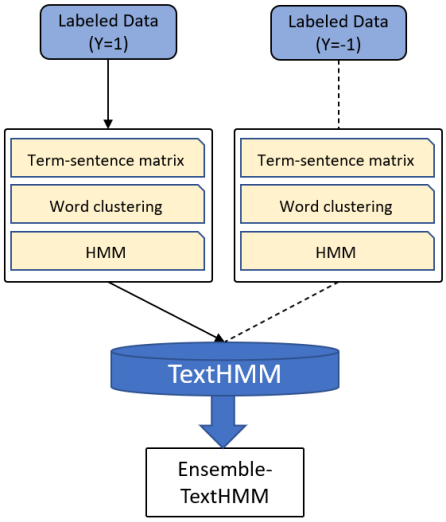

Figure 2.4: Architecture of Kang et al. ensemble of Hidden Markov Models using text clustering.

Yue et al. [99] used an ensemble of an English-Chinese sentiment dictionary they created and 3 DL algorithms detecting emotions in code-switching posts with happiness, sadness, fear, anger, and surprise labels. The DL algorithms assessed included RCNN (a combination of recurrent and convolutional neural networks), CNN, and LSTM with attention. Their ensemble DUTIR_938, which combined their sentiment dictionary and all DL classifiers, achieved a higher f-measure score than other single and ensemble classifiers they assessed. The other ensembles consisted of smaller pieces of the combined ensemble (e.g. CNN + RCNN without LSTM). These concepts are applied in my ensembles combining DL and TL algorithms.

Kang et al. [100] applied an ensemble of Hidden Markov Models based on text clustering to the GLUE SST2 task and achieved an accuracy of 86.1, similar to the reported accuracy of 88.1% for CNNs developed by Kim [104] and applied to the same task. This example reinforces the idea that ensembles consisting of differently trained versions of the same models can present more accurate results than a single application of the model, as is reflected in my cascading ensemble algorithms. See Figure 2.4 for the architecture implemented by Kang.

Thavareesan and Mahesan [101] combined Word2vec [42], fastText from Facebook [105], and rule-based sentiment analysis methods to automate the process of expanding Tamil sentiment lexicons. Their method was 88% accurate in predicting binary sentiment classes from Tamil texts after using their automated process to expand their sentiment lexicons from 2,951 positive and 5,598 negative words to 10,537 positive and 12,664 negative words. Lim and Madabushi [27] created an ensemble of BERT and a simple neural network with an input layer using TF-IDF features for SemEval 2020 sub-task A, a binary text classification problem. They reported a weighted f-measure score of 0.8128 on the Dev dataset after training their ensemble with 10% of the available training data. This f-measure score exceeded the BERT score of 0.8085 for sub-task A. These examples illustrate how other researchers have, in a limited fashion, explored combining DL and TL classifiers as well as using pieces of alternate libraries integrated in novel ways with ensembles, key concepts which are reflected in my embedding research supporting RQ1 as well as my ensemble creation and assessment in RQ3.

Babu and Eswari [28] built an ensemble of a BERT variant called CT-BERT (COVID Twitter BERT) [106], RoBERTa, and TF-IDF SVM. See Figure 2.5 for the architecture this research team used for binary text classification for WNUT-2020 Task 2. This ensemble had lower f-measure scores as compared to the singleton CT-BERT classifier they used for the same task. CT-BERT alone had a reported f-measure of 88.7% compared to 88.52% for their ensemble. Perrio and Madabushi [107] created three ensembles with RoBERTa as the primary model combined with variations including TF-IDF limitations on the considered tokens and a percentage metric predicting the probability of a character being numeric. Their ensemble achieved an f-measure score of 0.8910 for the same binary text classification task. These examples show limited examples wherein TL algorithms are included with ensembles, a concept which I extend in my widespread integration of TLs with larger ensembles in

my research.

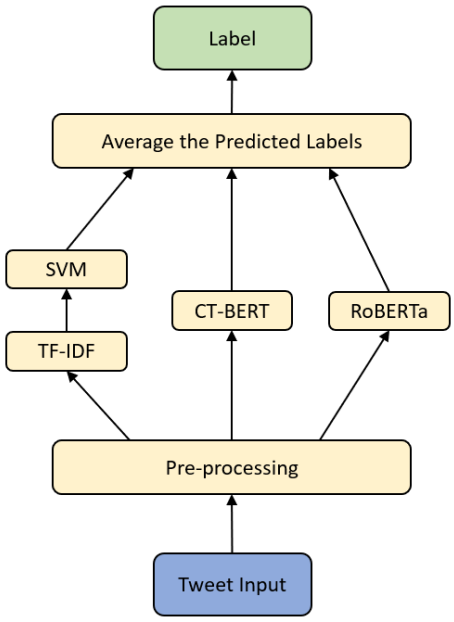

Figure 2.5: Ensemble architecture model used by Babu and Eswari for binary text classification.

Lai, Chan, and Chin [29] created and assessed cascading ensembles combining four classifiers, including kNN, SVM, Naìve Bayes, and an unspecified DL classifier. In their approach, they used 3 ensembles to generate initial predictions and a subsequent meta-classifier for the final prediction (see Figure 2.6). This is a promising approach to creating ensembles combining ML and DL algorithms, and this architecture inspired the design of some of my ensemble algorithms. They reported that the ensemble which used the DL classifier for the final meta-classifier outperformed any of the individual classifiers they assessed.

2.7 Evaluation

Sokolova and Lapalme [108] note that classification problems may fall into one of several categories, including binary (positive/negative), multi-class (each sample is labeled as belonging to one of multiple classes), multi-labeled (each sample may belong to several classes), or hierarchical (each sample has a proper place within a hiearchy of labels). Table 2.1 outlines common performance measures, per [108].

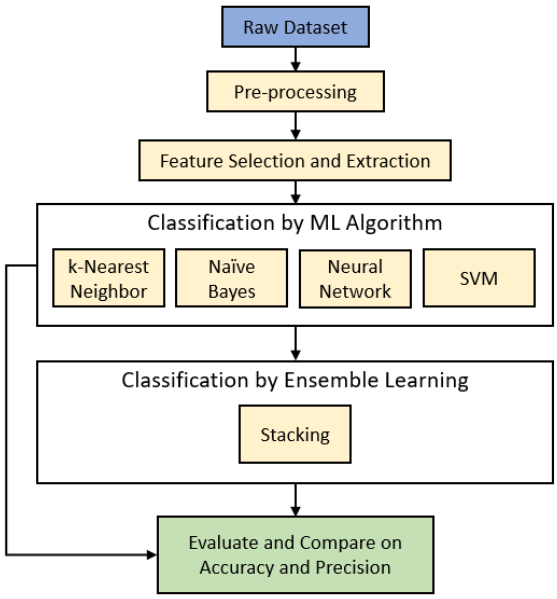

Figure 2.6: Ensemble architecture assessed by Lai, Chan, and Chin for sarcasm detection.

Table 2.1: Definitions of classification metrics from Sokolova and Lapalme.

Measure	**Evaluation focus**
Accuracy	Overall effectiveness of a classifier
Precision	Class agreement of the data labels with the positive labels given by the classifier
Recall (Sensitivity)	Effectiveness of a classifier to identify positive labels
F-score	Relations between data's positive labels and those given by a classifier
Specificity	How effectively a classifier identifies negative labels
AUC	Classifier's ability to avoid false classification

2.8 Review Summary

In this chapter, I discussed common tasks in the NLP pipeline and various algorithms for text classification using traditional machine learning, deep learning, transformer learning, and ensemble approaches to sentiment analysis and emotion detection. Within these areas, I have reached the following conclusions about the current state of emotion detection research:

- Binary and trinary emotion polarity detection approaches are numerous, robust, and reasonably accurate [47].

- Traditional ML classification is in broad use for binary and trinary emotion polarity detection [44, 95, 64], but multi-class emotion detection is a challenging problem wherein accuracy remains an issue.

- A recurring theme in ensemble classifiers is the idea that a weakness in one approach can be offset by iteration [88, 45], by combining multiple results in cascading ensembles [49, 90], and by using majority/plurality voting among multiple approaches to achieve greater prediction accuracy [48, 25, 88, 45].

- DL classification is in common use [72, 54, 32] and has recently been eclipsed by research into TL classifiers [34, 35, 40], and concurrently some limited examinations of ensemble approaches are also common [28, 107]; however, a robust examination of ensembles combining ML, DL, and TL approaches is needed to assess the viability of ensemble approaches and improve the overall classification accuracy for EMDISM. This work offers valuable research and insights regarding my robust examination of ensemble classifiers for EMDISM.

CHAPTER 3: DATASETS

4.88 billion people, approximately 62% of the world's population, uses the Internet, and the number of active social media users grew to 4.55 billion people in 2021. The average time spent browsing social media in 2021 reached nearly 2.5 hours per person. The user's ability to interact with other users via micro-blogging or direct text messaging is a key feature social media applications, and Twitter is the 15th most popular social media application, with more than 363 million users [109]. As of March 16, 2022, more than 500 million tweets were being sent every day, for a total of roughly 200 billion tweets per year, at a rate of approximately 6,000 Tweets per second, and 46% of Twitter users indicate using the platform has increased their political awareness [110]. 26% of users look at advertisements longer on Twitter than on other social media platforms and advertising revenue was expected to increase by 22% from the previous year, indicating the value of understanding how people are engaging with the platform and how they feel about those interactions. Regarding security considerations, researchers have examined how to track and identify violence using Twitter, with applications such as measuring the level of violence in Mexico through tweets [111, 112] and identifying criminal activity including gun violence, drug use, and drug dealing [113]. Given the ubiquity of Twitter, accessibility of large datasets, business opportunities in advertising and customer sentiment analysis, and the tendencies of some to use it to promote violence, I selected datasets relevant to Twitter for my research in building ensembles for fine-grained emotion detection in social media.

I chose the Harnessing Twitter (HT) dataset for my primary research. The portion of the HT dataset which was still publicly available for hydration from Twitter consists

of 1.2M tweets labeled with seven emotions, including **joy**, **sadness**, **anger**, **love**, **thankfulness**, **fear**, and **surprise**. Note that these emotions align closely with Ekman's six [19], varying only by including thankfulness and love and excluding disgust.

I considered other datasets for my primary research, including the dataset for SemEval-2019-Task 3 [98], Crowdflower [114], and the dataset used by Ranganathan [1]. The SemEval dataset was limited to approximately 30,000 samples with four categories of emotion, angry, happy, sad, and others. In my view, this dataset lacked samples to accurately reflect the breadth of data available, and it did not reflect granular coverage close to Ekman's six basic emotions. The Crowdflower dataset was interesting, with 40,000 samples and 13 emotions, and it provided a useful starting point for exploration. However, provenance for the associated research was more limited, and the larger number of emotion categories across a comparatively smaller number of samples gave rise to data sparsity considerations. I switched from this dataset to the HT dataset after using Crowdflower to test algorithms and explore the nuances of NLP for EMDISM. Later in my research, I found the availability of the dataset used in [1], with 184,471 samples. However, the repository did not contain documentation on the data organization and the original researcher was not responsive to clarification requests, so it did not seem practicable for use. Given these challenges and the strict Twitter limitations on sharing anything more than Tweet IDs with other researchers, I focused on the HT dataset.

3.1 Pilot Study Dataset - Crowdflower

For my pilot comparisons of traditional machine learning algorithms and explorations of deep learning algorithms, I selected the Crowdflower dataset [114], which consists of 40,000 tweets with non-standard emotion labels (more emotions than Ekman's six) annotated by crowdsourcing. This datset was among several used at the First AAAI Conference on Human Computation and Crowdsourcing [115] and the

dataset is part of the Cortana Intelligence Gallery at Microsoft. The Crowdflower data was processed by Liu, Kang, and Ken [116] to consolidate 13 emotion categories into five combined categories, including **neutral**, **happy**, **sad**, **anger**, and **hate**, using the same approach as Bouazizi and Ohtsuki [117]. However, I selected a more robust dataset to confirm my pilot research results and complete the remainder of my DL and TL research. I provide details for this dataset's origin and composition in the next section.

3.2 Primary Experimental Dataset - Harnessing Twitter

Wang et al. [118] presented a paper describing a Twitter dataset they assembled containing more than 2.5 million tweets, wherein each tweet was labeled with one of seven discrete emotions, including **joy**, **sadness**, **anger**, **love**, **thankfulness**, **fear**, and **surprise**, using emotion-related hashtags in each tweet to derive the annotations. They describe an ensemble approach for classification using unigrams, bigrams, emotion-bearing words, and POS tagging wherein they were able to achieve a highest classification accuracy of 65.57%. I refer to this dataset as the HarnessingTwitter (HT) dataset. Armin Seyeditabari [119] used the HT tweets in his dissertation research and was able to share the list of tweet identifiers and their sentiment labels, in accordance with Twitter's developer license agreement. I then used Hydrator [120] to scrape the tweet contents and metadata from Twitter and appended the sentiment labels to complete reassembly of the dataset. At the time I acquired the dataset through the Twitter developer API using Hydrator, 1.2M of the original 2.5M Tweets were available. See Table 3.1 for metadata about the distribution of the unbalanced classes within the HT dataset. I provide more details about the methodology I applied in building and assessing models based on the HT dataset in Chapter 4.

In the next chapter, I provide details of my research methodology, including software libraries used, pre-processing the HT dataset, creating models with ML, DL, and TL algorithms, creating and assessing ensembles of the most accurate algorithms, and

Table 3.1: Sample sizes for emotions in HT dataset

joy	349,419	thankfulness	72,505
sadness	299,412	fear	65,010
anger	261,806	surprise	11,978
love	153,017		

discuss the analytical metrics and approaches I used in evaluation and presention of my results.

CHAPTER 4: METHODOLOGY

In this chapter, I discuss the methodology I used for my primary research, as aligned with the research framework in Figure 1.1. This covers general aspects of the research approach, with details more specific to individual experiments in subsequent chapters. First, I describe the software libraries employed in my research. Next, in the Pre-processing section, I describe the methods I used to prepare my datasets for use in training ML, DL, and TL emotion classification models (research framework section A), including the tools and methods used in partitioning the dataset into training and testing sets and the tokenization and embedding techniques applied. In the model creation section, I provide specific details for how each ML, DL, and TL model was created, and I also describe the approaches used in comparing embedders and hyperparameters for use with DL and TL classifiers, respectively. In the Ensemble Creation section, I describe the various ensembles developed and tested. In the Analysis section, I discuss the various metrics used for assessing the performance of the base classification models and the ensembles I developed.

4.1 Software Libraries and Research Hardware

My experiments employed standardized platform implementations of different ML, DL, and TL algorithms based on the following common Python libraries:

- *scikit-learn* - train-test splits and analytics [121]
- *HuggingFace's Transformers* - basis of Simple Transformers [69]
- *Simple Transformers* - implements transformers commonly used for multi-label text classification [122]

- *Keras Tensorflow* - basis of HuggingFace's Transformers [123]
- *Pandas* - dataframe manipulation [124]
- *NLTK* - preprocessing text [125]
- *Numpy* - array-based math [126].

My primary research was conducted on a Micro-star International Z390 Gaming Infinite X Plus 9 desktop computer, with 48GB of RAM, an Intel(R) Core(TM) i7-9700K CPU, and a single NVIDIA GeForce RTX 2080 GPU.

4.2 Pre-processing

Following Symeonidis et al. [41], the following pre-processing techniques were applied to prepare the tweet messages in the HT dataset for use in training and analysis.

- Removed URLS.
- Removed username references (e.g. "@POTUS" was replaced with "").
- Removed hashtags (e.g. "#happy" was replaced with "").
- Removed numbers (e.g. "134" was replaced with "").
- Cast all remaining text to lowercase.
- Un-escaped html escape strings.
- Removed punctuation duplicates (e.g. '!!!!!' was replaced with '!').
- Replaced contractions with proper English phrases (e.g. "what's" was replaced with "what is, "'ve" was replaced with " have," and "n't" was replaced with " not.")
- Stripped extra whitespace from the beginning and end of each string.

- Lemmatized the verbs in each tweet.

Hashtags were removed because they had been directly employed as part of the original emotion labeling process for the HT dataset [118]. Wang et al. used 131 emotion hashtags as keywords to collect tweets relevant to seven emotion categories within Shaver's emotion hierarchy [18] and closely related to Ekman's six basic emotion [19]. Since hashtag comparisons to sentiment lexicons were used in labeling HT, removing the hashtags assured that the tweets in question would need to rely on the specific user text and textual context without specifically labeling the tweet via hashtag.

Given the size of the dataset, the overall time required to pre-process the dataset was lengthy, taking more than 1 hour just to iterate through the steps above across 1.2M samples. For this reason, and to avoid the possibility of introducing unintended errors while duplicating the pre-processing steps multiple times across different versions of training, testing, and assembling ensemble algorithms, I saved a clean, labelled version of the entire dataset for reuse across the various algorithms I developed in my research.

For each model in my research, the HT dataset was partitioned in a 70/30 train/test split using a random seed of 21. Each prediction model was trained with the training data, and the final model was saved for reuse in 10-fold cross validation testing. As part of the train/test partitioning, incorporating a specified random seed and data sample shuffling caused data splits to be stratified in a manner that preserved the same class balance ratios across all folds. Each model was then used to generate and save predictions and accuracy scores across 10 random slices of 30% of the data using random seeds 1-10 to ensure that datasets were never the same for validating accuracy of results.

4.3 Model Creation

My research is focused on investigating ensemble classifiers for detecting specific emotions from short social media texts. As Hansen and Salamon observed [25], the

correct place to start when building ensembles is to identify which algorithms are most proficient at a given classification task and include the most proficient in ensembles designed to help the strengths of one algorithm offset the weaknesses of another algorithm. To investigate this kind of balance in blended ensembles, I created models using ML, DL, and TL algorithms and trained each model using the same random slice of 70% of the HT dataset, with the other 30% used for evaluating the initial model as part of the model training process. Upon completion of building and accuracy testing each singleton model described in my research, the comprehensive set of predictions from each cross-validation slice of the dataset was cached for efficiency purposes to streamline inclusion of the predictions in my ensemble simple voting and weighted voting algorithms.

4.4 Ensemble Creation

In exploring ensembles for EMDISM, I followed the compositional structures outlined by Burke [26] as a guide for my work in ensemble algorithm creation. I chose to create and assess voting ensembles (both simple voting and weighted voting ensembles), cascading ensembles, and a cascading/switching ensemble. The following sections provide a general overview for these types of ensemble structures.

4.4.1 Voting Ensembles

Simple voting ensembles are relatively straightforward. In simple voting ensembles, one trains multiple classification algorithms to predict classes of samples within the dataset, then compiles the predictions into a scheme wherein the prediction with the most votes is presented as the prediction from the greater ensemble. *Weighted voting ensembles* are more complex, as they require development of some selection criteria to give more weight to certain algorithms based on an observed tendency within the prediction algorithms. For example, if a researcher were attempting to assign a topic classification to news headlines and observed that one algorithm was more accurate

in predicting politics or entertainment classes, the votes from that algorithm could be assigned to have greater weight than the predictions of others.

4.4.2 Cascading Ensembles

Cascading ensembles are created by using the output of one step in an ensemble algorithm as the input for the next step in an ensemble algorithm. I created several cascading ensembles, wherein the first phase of the ensemble predicted whether a sample was one of two or more super sets, wherein multiple final categories were combined to create larger sets of super classes in the first stage of the cascading ensemble. In the second stage of these ensembles, classifiers trained to predict classes within the super classes were used to predict the final classes.

4.4.3 Switching Ensembles

Switching ensembles are created when conditions within the data dictate switching from one algorithm to another to predict the class of a given sample. I implemented a cascading switching ensemble wherein the first stage predicted whether a sample would belong to the most robustly represented super class or a super-class with the least represented emotions.

4.5 Evaluation and Analysis

4.5.1 Classification Metrics

Accuracy, precision, recall, and f-score (or f-measure) are some of the most commonly considered measures to use in understanding how well a classifier is performing. In deciding which metrics to use in identifying the most appropriate members to include in my ensembles, I focused primarily on accuracy and used weighted f-measure as a critical second metric to help capture how effectively my classifiers were performing in regards to imbalanced classes in the HT dataset. As part of experimental analysis in Chapters 6 - 9, I provide detailed scores for the average accuracy for each singleton classifier and each ensemble, and I also compare the weighted precision,

weighted recall, and weighted f-measure scores for the top 5 ensemble algorithms to the same scores for BERT, the most accurate single classification algorithm among those I tested.

4.5.2 Overfitting Assessment

Overfitting is a common problem among classification algorithms, wherein a model is found to generally work well for predicting classes within the dataset upon which it was trained but suffers performance degradation on other datsets. I considered previous research [127, 128, 129] comparing validation loss and accuracy to assess how well a transformer model generalizes and avoid overfitting, as well as others who posit that sufficiently large datasets generate models wherein the flat part of a power-law learning curve describes a region of irreducible error [130]. To assess how well my models generalize, I performed 10-fold cross validation with a 70/30 split to provide robust sampling across the entire dataset and determined that the accuracy scores were stable across folds, with maximum deviations from the average generally within ± 0.00065 for ML models, ± 0.00038 for DL models, and ± 0.00062 for TL models. Chapter 8 provides comprehensive details of the accuracy comparisons across algorithms and ensembles and also provides evidence supporting my hyperparameter optimization findings via comparisons of accuracy and validation loss curves.

4.5.3 Summary

In this chapter, I provided details regarding the software libraries and hardware used in my research, described how my datasets for pilot and primary research were pre-processed for use with classifiers, as well as general model creation for individual ML, DL, and TL models. I also provided an overview of the ensemble approaches employed in this research. Finally, I discussed metrics for assessing individual and ensemble algorithms. In the following chapters, I describe the experiments and results I conducted to assess each type of algorithm and provide information supporting the

answer to the question of whether ensemble classifiers are more effective than singleton classifiers in emotion detection for short social media texts.

CHAPTER 5: PILOT RESEARCH AND RESULTS

This chapter covers two threads of pilot research that shaped the foundation of my EMDISM approach and investigation. I began my overall research program investigating authorship attribution, a different but related topic to EMDISM in NLP research [37]. Investigation of forensic authorship attribution in the social media context, however, was constrained by the limited scope and availability of reliable datasets. As emotion detection is a supporting factor of interest for FAA, I shifted my research focus to sentiment analysis and multi-class emotion detection. After changing task focus to EMDISM, I conducted pilot studies using the CrowdFlower dataset to understand the EMDISM research space.

In this chapter, I first introduce my preliminary research in authorship attribution. Second, I discuss the development and application of my research framwork (Figure 1.1) for pilot EMDISM research, examine the tools and commonalities between approaches, discuss the research I have completed using machine learning classifiers and deep learning classifiers, and discuss my initial efforts in creating voting ensembles.

5.1 Authorship Attribution

I began my research in natural language processing with an examination of authorship attribution techniques [37]. Authorship attribution, sometimes referred to as stylometry, is a classification problem wherein we attempt to identify an author based on characteristics of their writing styles. Forensic authorship attribution (FAA) is an extension of authorship attribution with a focus on creating authorship attribution techniques and refining the science to include the necessary accuracy and rigor

as to be applicable in law enforcement activities and criminal cases. During my research, I improved on the common n-gram (CNG) work of Potthast et al. to achieve an attribution accuracy of 87.8% through a revised implementation of CNG which automated the process of identifying the best profile size to use for classification. As part of my research, I adapted the NLP pipeline by Rocha et al. [36] for forensic authorship attribution. In my pilot EMDISM research, I extended this pipeline into a research framework for emotion detection (see Figure 1.1).

5.2 Pilot EMDISM Study

As an initial exploration in the EMDISM space, I conducted a pilot study with the more limited CrowdFlower dataset. This provided initial experience in working through the stages of the research framework for EMDISM and helped to shape the design and directions of the primary studies. In this section, I describe the research tools, data, component classifiers, initial ensemble methods, and evaluation in my pilot experiments for EMDISM.

5.2.1 Research Tools

All classifiers and utility programs were created in PyCharm Community Edition, executed in an Anaconda virtual environment with Python 3.7, and used the following libraries:

- **Keras Tensorflow** [67] - used for deep learning algorithms, including LSTM, C-LSTM, BiLSTM, GRU, BiGRU.
- **NLTK** [131] - used for data cleansing activities including lemmatization, stopword removal, etc.
- **Pandas** [124] - used for CSV file loading and saving, as well as dataframe manipulation for train/test splits, appending results, etc.
- **SciKit Learn** [121] - used for machine learning classifier pipelines.

5.2.2 Dataset Selection

I selected the CrowdFlower [114] dataset for pilot investigations, to provide a reasonable balance between dataset size and more agile performance considerations for exploring different approaches. Even for the CrowdFlower dataset, more complex deep learning algorithms still required a shift from laptop to a more powerful desktop computer for training of the BiLSTM model. More specifically, pilot experiments employed a derivation of the CrowdFlower data prepared by Liu, Kang, and Ken [116], referred to henceforth as CrowdflowerLKK, which enabled comparison with their reported results. For CrowdflowerLKK, the authors combined classes from the original CrowdFlower data using the same approach as Bouazizi and Ohtsuki [117]. Liu, Kang, and Ken consolidated classes into **neutral**, **happy**, **sad**, **anger**, and **hate**. Fun and love were combined to the happy class, as compared with the dataset used by Bouazizi and Ohtsuki.

5.3 Pilot Study Methodology

The pilot study methodology is organized according to the stages in my research framework (Figure 1.1): pre-processing, predicting, and assessing.

5.3.1 Framework A: Pre-processing CrowdflowerLKK

For framework part A, I cleansed the text of the tweets in the CrowdFlowerLKK dataset [114] with a simple pre-processing function in PyCharm which cast all text to lowercase, converted common English contractions to regular text, and stripped extra whitespaces from the tokenized text. Next, I used the NLTK [131] lemmatizer function to lemmatize the words of each tweet.

I experimented with removing stopwords from the text to remove additional noise from the dataset, but stopword removal reduced the accuracy of my results, so I elected to only lemmatize and perform basic cleansing on the text for the pilot research. Stopword removal is common in the field of natural language processing,

especially for text classification problems, so I was initially surprised that stopwords appeared to hold a significant value in the domain of EMDISM. However, this effect was demonstrated in Smetanin [54] as well as dos Santos and Ladeira [132], who also indicated stopword removal negatively affected their results. Given that the results of my pilot analysis followed those reported in [54, 132], I chose to retain stopwords in my datasets when training the models I assessed and used in my own ensembles.

5.3.2 Framework B: Predicting

After pre-processing, I developed a Python application to assess the performance of various ML and DL classifiers from the SciKit Learn library [121], as well as three custom ensembles. I began this research with the intuition that an ensemble of traditional machine learning and deep learning techniques may improve the state of the art in EMDISM by introducing novel predictions from ML and DL. I began by identifying representative best-performing classifiers of each type. To do so, I performed 10 fold cross-validation testing for individual ML and DL classifiers, as well as the ensemble approaches described in the following sections. After cross-validation testing, I averaged the accuracy for each algorithm across all test cycles to determine the highest average accuracy across all 10 testing folds.

5.3.3 Framework C: Assessing

This section provides testing and assessment results specific to each classifier type investigated, including ML, DL, and ensembles of ML and DL classifiers.

5.3.3.1 Comparing Machine Learning Classifiers

I compared the following ML algorithms, using their implementations in SciKit Learn:

- **SVM** - Support Vector Machine with stochastic gradient descent classifier (SGDC) model updating during training [58].

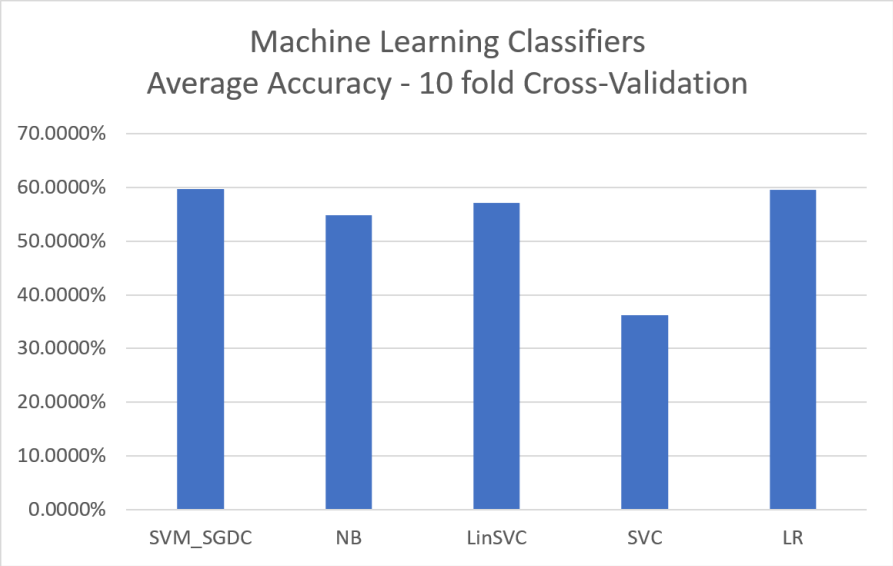

Figure 5.1: ML classifier accuracy chart.

- **NB** - Naïve Bayes [56].
- **SVC** - support vector classification [133], used for processing high dimensional sparse vectors by "...reducing the number of objects in the training set that are used for defining the classifier."
- **LinearSVC** - a variant of SVC designed to scale better to larger datasets [134].
- **LR** - logistic regression [135].

These algorithms were selected because each is well-suited to the task of classifying sparse vectors of tokenized word sequences for sentiment polarity, and I reasoned that their inclusion in a one versus rest pipeline using the SciKit learn library would provide a good comparison of their performance as applied to the multi-class emotion detection task.

Table 5.1 and Figure 5.1 show results of the comparison testing between machine learning algorithms. Both Logistic Regression and SVM_SGDC performed comparably, with both having approximately 59% to 60% accuracy. In comparison to Zainuddin and Selamat [136], who achieved **binary** sentiment prediction accuracy of

approximately 70% using SVM, or to Ramadhan et al. [137], who achieved **binary** sentiment prediction accuracy of approximately 74% using Logistic Regression, my accuracy is lower, but is comparable when considering the **multi-class** nature of the CrowdflowerLKK dataset used in my pilot research.

Table 5.1: Comparison of accuracy for ML and DL classifiers with CrowdFlowerLKK dataset.

ML Classifier	**Accuracy**
SVM_SGDC	**59.6758%**
NB	54.8573%
LinSVC	57.2108%
SVC	36.1979%
LR	59.5779%

DL Classifier	**Accuracy**
C-LSTM	**84.1320%**
LSTM	78.4655%
BiLSTM	81.6952%
GRU	73.7133%
BiGRU	78.8782%

5.3.3.2 Comparing Deep Learning Classifiers

With the lower overall accuracy for selected ML approaches, I proceeded to consider DL EMDISM approaches. I began my comparison of deep learning classifiers by implementing 5 neural network classifiers. For DL classifiers, multiple researchers have reported accurate text classification results using convolutional neural networks [138, 139, 140] or recurrent neural networks, so I focused on a selection of those including C-LSTM [32], LSTM [72], BiLSTM [54], GRU [141], and BiGRU [33].

To discover the maximum length for any token sequence in my pilot research, I tokenized and iterated across the entire CrowdflowerLKK dataset to determine a maximum token sequence length of 40. I created input layers through the application of a flexible embedding layer, with accuracies exceeding those of other researchers for the same CrowdFlowerLKK dataset [116, 117]. Whereas Bouazizi and Ohtsuki were only able to reach a classification accuracy of 60.2% in their approach to multiclass emotion detection using the CrowdFlowerLKK dataset, I achieved a 10-fold cross

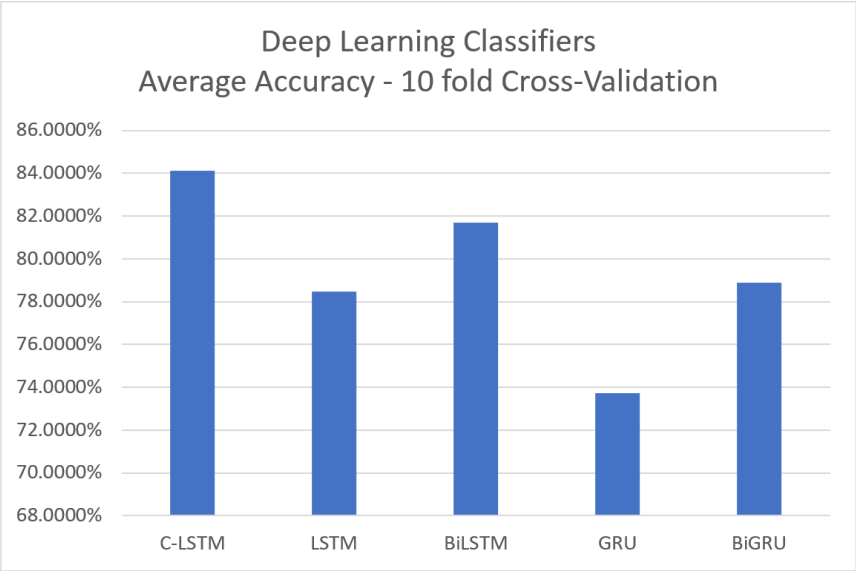

Figure 5.2: Deep learning classifier accuracy chart.

validation accuracy of 84.13% with C-LSTM using the data as prepared by Liu, Kang, and Ken. I used custom embeddings created using the Keras Embedding utility, with higher accuracy than other researchers, and this seemed to indicate that an approach tailored to the specific domain and dataset may be more effective than applying pre-developed embeddings generalized to other tasks. This problem required further study to confirm; hence, I examined this in my primary research in my assessments related to RQ1.

Table 5.1 and Figure 5.2 show a comparison of the performance of the selected deep learning algorithms. I noticed that C-LSTM and BiLSTM were the most accurate deep learning classifiers among those tested, with accuracies of 84.13% and 81.69% respectively. Given performance constraints, I tested an ensemble with C-LSTM for ensemble testing; however, I note the accuracy of the approach was comparable, and also improved on basic LSTM by more than 3%, so in applications where performance is not a consideration, BiLSTM could be a viable alternative for inclusion in an ensemble classifier. I further noted that the bidirectional approach for LSTM does seem to offer an accuracy improvement over LSTM, confirming that both preceding

and following text is important for emotion detection. This observation supports my decision to include TL algorithms in my research, given that the self-attention mechanism and interactions between the encoders and decoders are specifically designed to map the context of specific texts to other tokens within the text vector for a sample. Hence, I include TL embedders for assessment in RQ1 and several TL algorithms for assessment in RQ2 and RQ3 in my primary research.

5.3.3.3 Pilot Research in Ensemble Classifiers

Having considered baseline individual ML and DL classifiers, I proceeded to investigate initial potential for ensemble approaches. Guided by Burke's general ensemble structures [26], as well as previous ensemble work in EMDISM [47, 48, 49], I developed 3 ensemble approaches to emotion detection for the CrowdFlowerLKK dataset.

The *first ensemble* is a simple voting ensemble (EnSV), wherein I compared the results of 6 classifiers, SVM_SGDC, C-LSTM, LSTM, BiLSTM, GRU, and BiGRU and selected the prediction which had the most votes. C-LSTM was given the tie-breaking vote, in case no clear winner was identified. I selected the SVM_SGDC classifier based on its achieving the highest accuracy among the ML classifiers in a one versus rest implementation for multi-class emotion detection. By including SVM_SGDC, I hoped to introduce novel predictions from outside the DL group of classifiers. The *second ensemble* is a simple voting ensemble (EnSV), wherein I compared the results of only the 5 deep learning classifiers, C-LSTM, LSTM, BiLSTM, GRU, and BiGRU and selected the prediction which had the most votes. For the *third ensemble*, EnWgCLSTM, when there was no clear voting winner, I selected C-LSTM in a simplified weighting scheme which gave the tie breaking vote to the most accurate individual classifier.

Table 5.2 and Figure 5.3 show a comparison of the performance of my ensemble EMDISM algorithms. Overall results showed that the pilot ensemble methods did not perform as well in terms of accuracy as C-LSTM alone. However, EnSV - the

Ens. Classifier	Accuracy
EnSV	**82.4607%**
EnSVDL	82.2908%
EnWgCLSTM	82.3381%

Table 5.2: Ensemble classifier accuracy comparison. Note that EnSV, the voting ensemble which included SVM was more accurate than EnSVDL, the voting ensemble which only used deep learning algorithms.

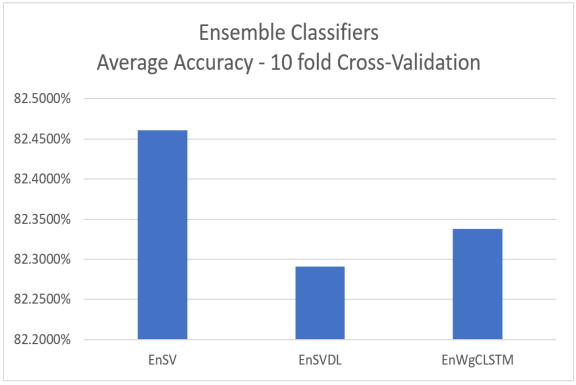

Figure 5.3: Ensemble classifier accuracy graph. EnSV had a higher accuracy than either EnWgCLSTM or EnSVDL, indictating inclusion of traditional machine learning algorithms may add value.

ensemble method which included votes from SVM and the entire set of deep learning algorithms - was more accurate than EnSVDL, the ensemble method with only votes from the deep learning algorithms. This indicates that the inclusion of traditional machine learning techniques has the potential to improve the accuracy of blended ensemble EMDISM algorithms.

The above observations support my decision to consider one or more DL algorithms in my ensemble research, given that including SVM, an ML algorithm, in the ensemble with DL algorithms was more accurate than an ensemble with DL algorithms alone. Hence, I include an assessment of ML, DL, and TL algorithms for inclusion in my ensembles as part of RQ2, in which I assess singleton classifiers as suggested by Hansen and Salamon [25], and as part of RQ3, in which I assess ensemble classifiers as suggested by Xia, Zong, and Li [95], in my primary research.

5.4 Pilot Study Insights

The pilot study provided numerous valuable insights, which ultimately were applied in my primary research. First, my own experimentation with stopword removal supported the findings of Smetanin [54] as well as dos Santos and Ladeira [132], as my

classifier accuracy was reduced when stopwords were removed. Thus I chose to retain stopwords and otherwise followed the general pre-processing guidelines of Symeonidis et al. [41].

Next, I gained valuable experience in adapting the Rocha NLP pipeline [36] I used in my FAA research [37] to the common tasks of NLP research [52]. As my research questions align closely with each phase of my research framework (see Figure 1.1), the pilot research provided a proving ground of sorts to apply the framework and identify questions relevant to each part of the framework.

Furthermore, I gained valuable insight into the process of building ensembles, including initial explorations of simple and weighted voting schemes, as well as ideas about how to assemble and assess more complex ensembles, which ultimately led to the switching and cascading/switching ensembles described in my primary research.

Because my DL classifiers using custom embedding layers were more accurate than other developers examining the CrowdflowerLKK dataset [116], I understood that the embedding approach used with DL algorithms may affect the accuracy of their predictions, so I integrated this intuition into my examination of embedding approaches in RQ1, as aligned with Framework Section A.

Finally, my confirmation that bidirectional DL algorithms were generally more accurate than their unidirectional counterparts demonstrated the importance of contextual relationships between tokens in text. This insight and the illustrated relevance of embedding algorithms combined to lead me to include TL embedders and TL classifiers in my primary research, and I describe the results of including these elements in 6, 7, and 9.

The pilot study analysis provided a foundation for experimental development and for considering the primary research questions in the context of a substantially larger scale dataset. I needed to clarify which embedder to use for pre-processing my DL input layers EMDISM (RQ1), I needed to identify the best candidate algorithms from

ML and DL classifiers (RQ2), and I needed to perform additional evaluation of ensemble approaches to EMDISM (RQ3) based on those results. As part of the development of the primary research studies, RQ3 was expanded to include TL algorithms based on embedder comparisons in addressing RQ1.

CHAPTER 6: ML CLASSIFIER EXPERIMENTS AND RESULTS

The research described in this chapter focused on identifying the average accuracy with 10-fold cross validation for individual candidate ML models, with the ultimate goal of including one or more of the most accurate models in my ensemble creation and evaluation research. Based on experience from the pilot studies and further literature review, I began a more in-depth primary investigation of the potential for ensemble approaches in EMDISM. This involved broader analysis of individual component classifier models, as well as deeper development of blended ensembles to consider tradeoffs in individual component performance, following my framework approach. This chapter considers component ML approaches, Chapters 7 and 8 consider component DL and TL approaches. And Chapter 9 considers blended ensemble approaches.

6.1 Traditional Machine Learning Model Creation

In order to identify candidate ML approaches for ensemble participation, I investigated a representative set of widely used supervised ML approaches. Of the classifiers discussed in 2.3, the following ML approaches were selected, and I created ML models for each using the Scikit-Learn and NLTK libraries for Python:

- SVM
- Naïve Bayes
- Linear SVC
- Logistic Regression
- Decision Trees

I considered including KNN [60] and maximum entropy [63] in my ML classifier assessments as well; however, each of these algorithms presented unique difficulties in model creation. For KNN, selecting the correct k number of nearest neighbors to assess in a cluster is critical to achieving high accuracy in text classification, and that estimation was considered to be out of scope for the overall ensemble analysis. In regards to maximum entropy, Rao et al. [64] noted that combining topic mapping, emotional valence (strength of emotion), and emotion labels yielded the most accurate results; however, neither the Crowdflower, CrowdflowerLKK, nor the HT data included topic and emotional valence labeling, rendering the task of achieving high prediction accuracy problematic.

I created and saved models for each of the above algorithms trained on the HT dataset. ML models employed TF-IDF tokenization from the SciKit-Learn library as part of pre-processing to focus the attention of the ML pipelines used on the words with the most information.

A one versus rest pipeline was created for each model, the model was trained, and the final model was saved for reuse in 10-fold cross validation testing. Each model was then used to generate and save predictions and accuracy scores across 10 random slices of 30% of the data using random seeds 1-10 to ensure that datasets were never the same for validating accuracy of results.

6.2 Experiment 1 - ML Classifier Assessment

The experiments in this section align with my research framework section B as shown in Figure 1.1 and address RQ2 for the ML classifier assessment. All the ML models described in this section were created as described in Section 6.1. For this experiment, I created the following hypotheses to establish an average accuracy threshold for a fair comparison between algorithms.

- $H_{1.0}$: All ML algorithms assessed have an average accuracy with no statistically significant difference.

Table 6.1: Average accuracy of ML models with 10-fold cross-validation.

ML Models	**Avg Accuracy**
Decision tree	81.05%
Linear SVC	64.20%
Logistic Regression	61.55%
Naïve Bayes	57.97%
Support Vector Machine	54.13%

Table 6.2: 10-fold cross-validation accuracy scores for ML models.

Seed	**Dec. Trees**	**Lin. SVC**	**Log. Reg.**	**Naïve Bayes**	**SVM**
1	81.12%	64.23%	61.58%	57.99%	54.20%
2	80.93%	64.10%	61.46%	57.93%	54.02%
3	81.04%	64.28%	61.37%	58.06%	54.16%
4	81.07%	64.18%	61.51%	57.94%	54.14%
5	81.07%	64.12%	61.51%	57.89%	54.08%
6	81.07%	64.18%	61.51%	57.97%	54.23%
7	81.05%	64.25%	61.61%	58.00%	54.13%
8	81.10%	64.16%	61.59%	58.02%	54.15%
9	80.99%	64.26%	61.59%	57.99%	54.12%
10	80.98%	64.23%	61.52%	57.95%	54.11%

- $H_{1.1}$: At least one ML algorithm is more accurate than the rest by a statistically significant margin.

See Table 6.1 for the average accuracy of my DL models with 10-fold cross-validation. Experiment 1 identified Decision Trees as the most accurate classifier among the traditional ML algorithms I tested for EMDISM, with an average accuracy of 81.05%. The largest average accuracy for all other ML classifiers was for Linear SVC, with 64.20% average accuracy. I completed a single-factor analysis of accuracy variance between decision trees and linear SVC and determined that the variance is statistically significant, with a p-value of 1.62e-40. Thus I must reject the $H_{1.0}$ hypothesis and accept the $H_{1.1}$ hypothesis, and **Decision Trees is therefore appropriate for inclusion in my ensemble algorithm experiments.**

For verification purposes, I have provided the accuracy scores for each random seed of the most accurate classifier of the ML classifiers. Table 6.2 provides the

comprehensive set of accuracy scores for the Decision Tree model used in my research and included as part of my ensemble algorithms. As a reminder, each model was trained with a random slice of the HT dataset generated by Scikit-Learn's train/test split utility seeded with 21 as the random seed. The standard deviation from mean for the cross-validation accuracy scores for C-LSTM with custom embedder was within ±0.00059, indicating the accuracy was stable across all folds.

6.3 Discussion

In this chapter, I described my comparison of various ML text classification algorithms. This research supported RQ2, in that I identified the most accurate algorithm for inclusion in my ensemble research supporting RQ3. I note that the decision to include decision trees in the analysis based on the research of Ranganathan et al. [44] was validated, based on the much higher DT average accuracy (81.05%) over its closest ML competitor, Linear SVC (64.20%). Furthermore, I note the value in confirming the results of my pilot research with models trained on the robust HT dataset, given that my pilot research identifed SVM as the most accurate ML algorithm, whereas SVM was the least accurate of the algorithms I compared with HT. This implies that differences in dataset metadata may influence the accuracy of component algorithms selected for inclusion in ensembles. More specifically, the key differences in CrowdflowerLKK and the HT dataset include the number of samples (40K vs. 1.2M respectively) and the number of emotions (5 vs. 7 respectively). As I expect research in EMDISM to grow to consider both more emotions and larger datasets, I suggest future researchers limit the use of SVM in ensembles to datasets more similar to CrowdflowerLKK than the HT dataset, both in number of classes and number of samples.

When I began the ML comparisons, I had hopes that several of the selected algorithms would be sufficiently accurate to include in my ensembles; however, given DT's much higher accuracy, I focused on this algorithm for inclusion in many of my

ensembles, as I discuss in greater detail in 9. For the sake of completeness, I ultimately assessed one ensemble with all classifiers included, but only those using DTs yielded better results.

CHAPTER 7: DL CLASSIFIER EXPERIMENTS AND RESULTS

7.1 Deep Learning Models

Having analyzed candidate ML models for EMDISM ensembles, I proceeded to investigate candidate DL models. The research described in this chapter was designed to identify the average accuracy with 10-fold cross validation for my implementations of each selected DL model as well as to determine if there may be some value in alternate methods for creating embedding layers using tokenizers from pre-trained TL libraries, with the ultimate goal of including one or more of the most accurate models in my ensemble creation and evaluation research. My DL experiments employed standard platform implementations of components, embedders, and deep learning approaches using the following common Python libraries:

- scikit-learn [121]
- HuggingFace's Transformers [69]
- Keras Tensorflow [123]
- Pandas [124]
- NLTK [125]
- Numpy [126]

I created DL models using the algorithms and embedders shown in Table 7.1 using the Keras TensorFlow library. I created and saved new models for each of the approaches. The following subsection describes the methods used to compare embedders between an embedder customized for the dataset and the embedders specific to the Keras

Tensorflow libraries for BERT, RoBERTa, ELECTRA, XLM-RoBERTa, and XLNet. To be clear, in this portion of my research, I was testing the **embedders** for the TL algorithms above, **not** the TL base models or models fine-tuned from the base models.

7.1.1 Embedder Comparisons

For the DL models, I compared the effects of using different embedders on the average 10-fold cross-validation accuracy for each combination of embedder and algorithm. Table 7.1 lists the models and embedder variants compared. In total, there were 6 embedding approaches and 5 DL models compared with 10-fold cross-validation testing for a total of 300 iterations across all combinations of model, embedder, and seed. See Section 7.2 in this chapter for more details about the interactions between the embedders and DL models and the effects of using different embedders from the transformer libraries with DL models.

Table 7.1: Models and embedders compared for use with DL models.

Models	**Embedders**
BiLSTM	Custom
C-LSTM	BERT
LSTM	ELECTRA
GRU	RoBERTa
BiGRU	XLM-RoBERTa
	XLNet

The general process for each embedding approach to convert a given tweet to an embedded numeric vector is as follows:

1. Pre-process as described in section 4.2.
2. Use the tokenizer for the specified embedding approach to convert each text sample to a tokenized vector.
3. Pad each vector to a uniform length for use in the input layer of the selected deep learning algorithms.

I chose 40 as the padded vector length, which was based on the assumption that the average word length for English words is 4 to 7 characters, yielding an estimated maximum range of 35 tokens in a 140 character tweet, as was the restriction for all tweets in the HT dataset. As 40 was programmatically confirmed as the maximum length for any token sequence in my pilot research with the Crowdflower dataset, and as the tweets in HT were restricted to the same length as those in the Crowdflower dataset, it was reasonable to use the same padded vector length when working with the HT dataset.

I developed a straightforward baseline embedder that is trained on short social-media texts, in order to make an initial comparison with modern, commonly available, out-of-the-box embedders. My customized embedding approach employs the baseline Keras Tensorflow tokenizer [123] as a generic equivalent to tokenizers for embedders from the HuggingFace Transformers library. To create the vocabulary for the customized embedding approach, I initialized the tokenizer vocabulary by fitting on the total text content of the HT dataset. This yielded a vocabulary size of 179,181 tokens for the custom embedding approach. I used the BERT [34], ELECTRA [38], RoBERTa [35], XLM-RoBERTa [39], and XLNet [40] tokenizers as a representative sampling of modern embedders from pre-trained TL classifiers.

7.1.2 Deep Learning Model Creation

Each DL model was trained with the same hyperparameters, as follows:

- **Embedding dimensions:** 100
- **Vocabulary size:** Determined by tokenization approach. For my custom embedding approach, the vocabulary size was 179,181, and this vocabulary size was used in the models analyzed for inclusion in my ensembles.
- **Batch size:** 2500
- **Maximum embedding length:** 40

- **Dropout:** 0.25
- **Recurrent dropout:** 0.0
- **Number of classes:** 7
- **Activation:** softmax
- **Reduce learning rate on plateau monitoring:** val_categorical_accuracy
- **Minimum learning rate:** 0.001
- **Training epochs:** 50 for LSTM-based models, 100 for GRU-based models.

As noted above, the only deviations from standard were in the vocabulary size, as this was determined based on the vocabulary size of the embedder used, and in the number of training epochs. The GRU-based models took longer to reach the bottom of the error correction plateau, so they were trained for additional iterations.

Table 7.2: Average accuracy for DL models by embedder used in input layer.

DL Model	**BERT**	**Custom**	**ELECTRA**	**XLM-R**	**XLNet**	**RoBERTa**
BiGRU	74.76%	**76.94%**	74.72%	73.62%	73.68%	75.26%
BiLSTM	74.63%	**76.66%**	74.71%	74.06%	73.87%	74.86%
C-LSTM	77.60%	**78.85%**	78.13%	77.49%	76.84%	78.29%
GRU	66.17%	**69.61%**	66.13%	65.63%	65.40%	66.58%
LSTM	71.94%	**74.30%**	72.05%	71.31%	70.96%	72.45%

7.2 Experiment 2 - DL Classifier Assessment

The embedder experiments in this section align with my research framework section A, as shown in Figure 1.1, and address RQ1 for the DL classifier assessment. The overall DL classifier assessment aligns with research framework section B and RQ2. The vocabulary size for each model varied according to the embedder used in creating the model. The number of training epochs varied, where the models based on LSTM variants used 50 epochs and the models based on GRU used 100 epochs. All other parameters were the same across all model/embedder variants created and assessed.

Table 7.3: 10-fold cross-validation accuracy for C-LSTM model using selected TL embedders (not algorithms) in input layer.

Seed	Custom	BERT	ELECTRA	RoBERTa	XLM-R	XLNet
1	**78.89%**	77.68%	78.17%	78.42%	77.59%	76.93%
2	**78.76%**	77.50%	78.10%	78.22%	77.41%	76.83%
3	**78.86%**	77.61%	78.09%	78.26%	77.47%	76.89%
4	**78.85%**	77.59%	78.12%	78.29%	77.52%	76.80%
5	**78.82%**	77.57%	78.10%	78.16%	77.40%	76.74%
6	**78.83%**	77.61%	78.16%	78.34%	77.49%	76.83%
7	**78.87%**	77.60%	78.16%	78.33%	77.44%	76.84%
8	**78.89%**	77.63%	78.15%	78.32%	77.52%	76.83%
9	**78.85%**	77.60%	78.16%	78.29%	77.50%	76.88%
10	**78.87%**	77.60%	78.13%	78.29%	77.51%	76.84%

7.2.1 Experiment 2.1: Embedder Comparison with DL Models

For this experiment, I created the following hypotheses to establish a threshold for determining which embedder to use in my DL classifiers:

- $H_{2.1.0}$: There is no statistically significant difference in average accuracy resulting from using a custom embedder or a TL library embedder from BERT, ELECTRA, RoBERTa, XLM-R, or XLNet.
- $H_{2.1.1}$: At least one embedder will generate predictions with a statistically significantly greater average accuracy than other embedders assessed.

Table 7.2 shows the average accuracy of each embedding method per deep learning model. Note that the custom embedding approach yields a higher average accuracy across every deep learning model tested, improving on the accuracy with RoBERTa using CLSTM by 0.56%. Also, I found that CLSTM was the most accurate of the models, improving on the accuracy of BiLSTM by 2.19% when using the custom embedding approach, and delivering a combined average accuracy across all approaches of 77.87%, more than 3% more accurate than BiLSTM.

To determine whether the variance in accuracy between embedding types was statistically significant, I first performed a single factor analysis of variance between the

results generated by each embedding approach within a given deep learning model type. In every case, the F value was much higher than the F-crit value, indicating that the variances between embedding types were statistically significant. For example, in my analysis of the accuracy variance for CLSTM across all embedding types, the F value was 1988.789, whereas the F-crit value was 2.38607, confirming the significance of the variance in accuracy between approaches.

In addition, I performed a series of two sample t-tests assuming equal variances to assess the variances in average accuracy between the customized embedding approaches and each of the pre-trained approaches. Using a two-tailed t-test assuming equal variances between my accuracy results comparing each embedding approach with the customized approach, with 95% confidence and 18 degrees of freedom, I calculated t-values between 21.92 (RoBERTA embedding with CLSTM) and 173.40 (XLNet with LSTM) and t critical values of 2.10. In every case comparing the custom approach to each other embedding approach across all deep learning algorithms, the t Stat score exceeds the t Critical score and the largest p-value score across all comparisons was $1.964e-14$. Given that the custom embedding approach yielded statistically significantly greater average accuracy than the other pre-trained embedders assessed, I must reject the $H_{2.1.0}$ hypothesis and accept the $H_{2.1.1}$ hypothesis. **Thus I confirm that the accuracy increase when using the custom embedding approach is statistically significant.**

Note that the custom embedder tailored to the HT dataset yielded the most accurate results in every case, with the most accurate DL model and embedder combination delivering an average 10-fold cross-validation accuracy of 78.85% for C-LSTM and the custom embedder. For verification purposes, I have provided the accuracy scores for each random seed of the most accurate classifier of the DL classifiers. See Table 7.3 for 10-fold cross-validation accuracy scores of the C-LSTM algorithm and all embedder combinations. As a reminder, each model was trained with a random

Table 7.4: Average accuracy difference for DL models by embedder used in input layer. Values reflect accuracy increase when using the custom embedder.

	BERT	**ELECTRA**	**XLM-R**	**XLNet**	**RoBERTa**	**Avg by Model**
BiGRU	2.19%	2.23%	3.33%	3.27%	1.68%	**2.54%**
BiLSTM	2.03%	1.95%	2.60%	2.79%	1.79%	**2.23%**
C-LSTM	1.25%	0.72%	1.36%	2.01%	0.56%	**1.18%**
GRU	3.45%	3.48%	3.99%	4.21%	3.03%	**3.63%**
LSTM	2.37%	2.25%	3.00%	3.34%	1.85%	**2.56%**
Avg. inc. per Emb.	**2.26%**	**2.12%**	**2.85%**	**3.12%**	**1.78%**	

slice of the HT dataset generated by Scikit-Learn's train/test split utility seeded with 21 as the random seed. Please note that I calculated the standard deviation from mean for the cross-validation accuracy scores for C-LSTM with custom embedder, and found the score was within ±0.00038, indicating the accuracy was stable across all folds. To highlight the impact of embedder choice on accuracy, the calculated average accuracy increase by model and by embedder is available in Table 7.4. The custom embedder was 1.18%-3.63% more accurate across all DL models.

7.3 Discussion

The experiments described in this chapter provided several interesting insights. This research supported RQ1, as I examined creating input layers using alternative embedders for DL algorithms, and RQ2, as I identified the most accurate combination of DL and embedding approach for inclusion in RQ3. First, I confirmed that the customized embedding approach I used in my pilot research, tailored to the dataset upon which the model was trained, yielded better prediction accuracy for all DL algorithms. Thus, I selected the models based on the custom embedding approach for use in further ensembles. Second, I found that the DL models were generally more accurate than the ML models, except for the DT model. I also was led to consider the TL classifiers in greater detail, based on my work with their embedding tokenizers.

Finally, the research described in this chapter eventually led to the exclusion of

most of the DL models in my ensemble research, given that the best DL models were less accurate than DT models or TL models. In order to validate whether any DL algorithms were appropriate for ensembles, I did create 4 simple voting ensembles using DL models, including one using only the most accurate DL model (CLSTM) and several with all models; however, none of these ensembles were more accurate than the BERT baseline model. See Chapter 9 for details about the ensembles I created and assessed. In Chapter 8 I discuss my TL model comparisons research.

CHAPTER 8: TL CLASSIFIER EXPERIMENTS AND RESULTS

8.1 Transformer Models

The research described in this chapter was designed to identify the average accuracy with 10-fold cross validation for my implementations of each selected TL model, with the ultimate goal of including one or more of the most accurate models in my ensemble creation and evaluation research. This research supported RQ1 and RQ2, as after examining using the TL embedding approaches in the previous chapter, I assessed which TL algorithms to include in my ensemble research supporting RQ3, if any. As part of this research, I found ambiguous guidance on the best hyperparameter optimization approaches, so I extended my research to discover the best combination of basic TL hyperparameters which yielded the best accuracy for EMDISM while avoiding overfitting. My TL experiments employed standard platform implementations of components, embedders, and transformer classification approaches using the following common Python libraries:

- scikit-learn [121]
- SimpleTransformers [122]
- HuggingFace's Transformers [69]
- Keras Tensorflow [123]
- Pandas [124]
- NLTK [125]
- Numpy [126]

I created TL models using the algorithms and base models defined in Table 8.1 using the SimpleTransformers library.

8.1.1 Transformer Model Creation

I fine-tuned new models for each of the referenced algorithms using base models developed by the original authors of the papers describing each algorithm, and accessed from the models saved on the HuggingFace Transformers library site. I acknowledge that there are some models provided by subsequent researchers which are fine-tuned for text classification tasks using Twitter data; however, the intent of my research was to assess the effectiveness and accuracy of ensemble approaches based on core algorithms, hence the decision to fine-tune base models with the HT dataset. For example, Barbieri et al. [142] have numerous variants of their "cardiffnlp/twitter-roberta-base-sentiment" models available for use; however, I note that they specify in their supporting paper for the model that their base models are trained on learning rates which are half those recommended by Devlin et al. [34], their own paper suggests additional research regarding how they handle emojis, and and their models were trained using only 60,000 tweets, which I suggest is unlikely to adequately capture the diverse nature of tweets in the greater landscape of Twitter posts. Other models, like "digitalepidemiologylab/covid-twitter-bert-v2-mnli," are based on the work of other researchers [143] and focused on specific topics (COVID in this example). So to reiterate, to avoid the many variables involved in variant Twitter-specific models, I used the base models specific to the original authors for my research.

Table 8.1: Algorithms and base models used for TL models.

Algorithm	**Base Model**
BERT	bert-base-uncased
ELECTRA	google/electra-base-discriminator
RoBERTa	roberta-base
XLM-RoBERTa	xlm-roberta-base
XLNet	xlnet-base-cased

8.1.2 Transformer Hyperparameter Optimization

When working with TL algorithms, we must answer key questions regarding hyperparameters, as follows:

- How do we know when to stop fine-tuning (this may be restated as how many fine-tuning epochs should we complete)?
- Which learning rate should we use?
- Which batch size should we use?

It is notable that whereas each of the various transformer algorithms I considered provided some insight into how their models were developed, details on hyperparameter selections explored were limited. For example, in [34], Appendix A.3 notes a constant dropout of 0.1, batch sizes of 16 or 32, Adam learning rates of 2e-5, 3e-5, or 5e-5, and either 2, 3, or 4 learning epochs, stating that these values were found to "...work well across all tasks...". In addition, it is noted that datasets with more than 100k training samples were less sensitive to hyperparameter choices, and that fine-tuning was generally fast enough that it was reasonable to simply iterate through the different hyperparameter options and pick the model with the best results. I interpret this as a recommendation that a basic grid search [86] approach is expected to be sufficient for hyperparameter optimization. [22]

Given the incomplete answers to the above questions, I chose to start my assessment by completing a grid search across parameters recommended by Devlin et al. [34], including a comparison of learning rates, learning epochs, and batch sizes.

8.2 Experiment 3 - TL Classifier Assessment

The experiments in this section align with my research framework section B as shown in Figure 1.1 and address RQ2 for the TL classifier assessment. TL models described in this section were created as described in Section 8.1.1. Reference Ta-

ble 8.1 for the specific models selected for fine-tuning of each TL algorithm. Before I could compare transformers, I needed to answer key questions about how to build the most accurate models without overfitting. See 8.1.2 for hyperparameter optimization questions assessed in experiments 3.1 and 3.2.

8.2.1 Experiment 3.1 and 3.2: Hyperparameter Optimization

I executed a hyperparameter grid search for BERT, ELECTRA, RoBERTa, XLM-RoBERTa, and XLNet with the following parameters:

- **Learning rates:** 2e-5, 3e-5, 5e-5
- **Fine-tuning epochs (Experiment 3.1):** 2, 3, 4
- **Fine-tuning epochs (Experiment 3.2):** 5, 6, 7, 8, 9, 10
- **Batch size:** 16 (attempted), 8 (final)

After initially trying batch sizes of 16, I used batch sizes of 8 for all permutations of models, epochs, and learning rates. The relatively smaller batch size of 8 (vs. 16 or 32 as recommended by Devlin et al.) was used because some models gave rise to buffer overrun errors with the Nvidia Geforce RTX 2080 GPU used in experimentation. Moreover, the batch size selection follows Masters and Luschi [144], who found "...that using small batch sizes for training provides benefits both in terms of range of learning rates that provide stable convergence and achieved test performance for a given number of epochs."

In Experiment 3.1, I performed 10-fold cross-validation testing across the models I fine-tuned with a grid search using the BERT-recommended parameters in order to establish baselines from the BERT recommendations. Results for the 5e-5 learning rate are shown in Table 8.2. In the domain of fine-grained emotion detection from social media texts, I found that the higher learning rate, 5e-5, yielded more accurate results than the smaller 2e-5 and 3e-5 rates. This trend held across all transformer

Table 8.2: Experiment 3.1 - Accuracy per epoch with BERT recommended fine-tuning epochs.

Model	**2eps**	**3eps**	**4eps**
BERT	**74.14%**	**78.13%**	**81.43%**
ELECTRA	72.76%	76.44%	79.38%
RoBERTa	72.49%	75.07%	77.73%
XLM-RoBERTa	70.43%	72.43%	75.16%
XLNet	71.67%	75.13%	77.79%

models, with BERT yielding the most accurate results after 4 epochs at an average accuracy of 81.43%. I also determined that the increase in accuracy was most prevalent in the BERT models, with a difference of approximately 3-4% increase for BERT and ELECTRA and 2-3% for other models.

I also noted that accuracy continued to show substantial increases across training epochs. For example, BERT at the 5e-5 learning rate improved over 3% in accuracy from the 3rd to 4th training epoch. Given the rate of increase was still decreasing relatively slowly between the 3rd and 4th epochs, I performed Experiment 3.2 with additional grid search testing to determine where the rate of increase was low enough that the increase in accuracy was prohibitively expensive when compared to the increased training time. Results showed that the rate of accuracy increase across all models and epochs tended to flatten significantly at about the 8-10 epoch range, as shown in Figure 8.1. When I extended the number of fine-tuning epochs to 10, accuracy increased 0.21% between the 9th and 10th epoch to 88.06%, whereas the accuracy increased over twice as much between the 8th and 9th epoch, with an increase of 0.46%. Furthermore, I performed a 2 factor analysis of accuracy variance between models and epochs and found that the variance between the recommended number of 4 fine-tuning epochs, as compared to my recommended number of 9 epochs was statistically significant, with a p-value of 9.1e-134 between models and 1.8e-167 between epochs.

Table 8.3: Experiment 3.2 - Accuracy per epoch with additional fine-tuning epochs.

Model	5eps	6eps	7eps	8eps	9eps	10eps
BERT	**83.81%**	**85.43%**	**86.67%**	**87.39%**	**87.85%**	**88.06%**
ELECTRA	81.90%	83.78%	85.17%	86.11%	86.72%	86.96%
RoBERTa	79.71%	82.05%	83.34%	84.73%	85.61%	85.96%
XLM-RoBERTa	77.56%	79.31%	81.25%	82.67%	83.64%	84.04%
XLNet	80.20%	82.18%	83.76%	85.01%	85.80%	86.09%

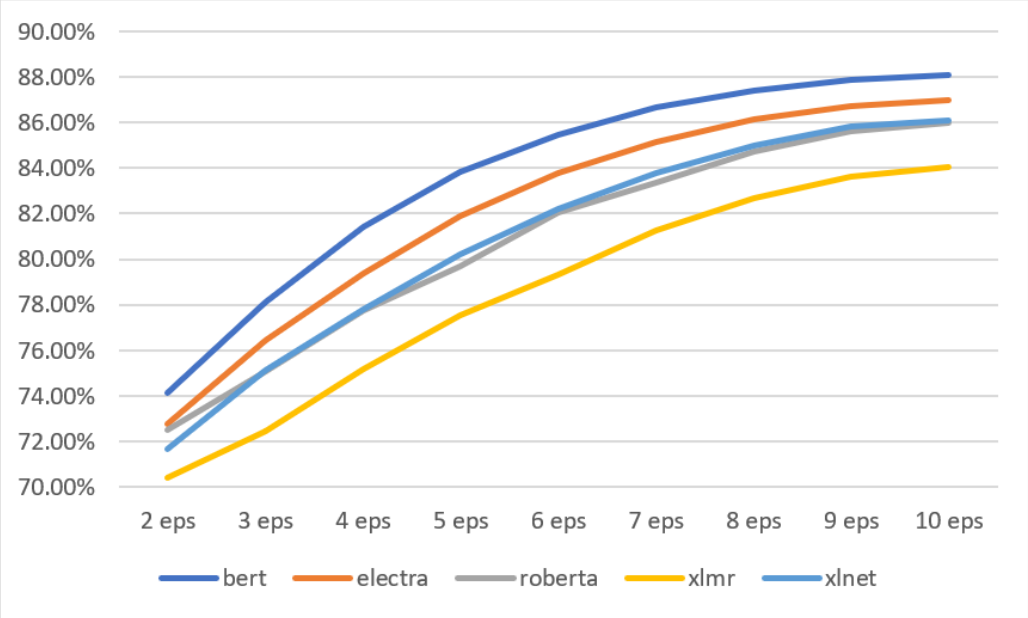

Figure 8.1: TL accuracy increase curve flattens at 8-10 fine-tuning epochs.

Table 8.4: Experiment 3.3 - Validation loss assessments for TL models.

EPOCH	BERT	ELECTRA	RoBERTa	XLNet	XLM-RoBERTa
2	0.17681	0.18367	0.18548	0.19001	0.19749
3	0.15511	0.16370	0.17077	0.17128	0.18447
4	0.13773	0.14828	0.15680	0.15644	0.17119
5	0.12642	0.13618	0.14576	0.14460	0.15827
6	0.12191	0.12787	0.13411	0.13459	0.14936
7	0.11936	0.12628	0.12931	0.12872	0.13939
8	**0.12031**	**0.12177**	0.12327	0.12355	0.13452
9	0.12362	0.12371	**0.12131**	**0.12278**	0.13217
10	0.12716	0.12589	0.12283	0.12529	**0.13142**

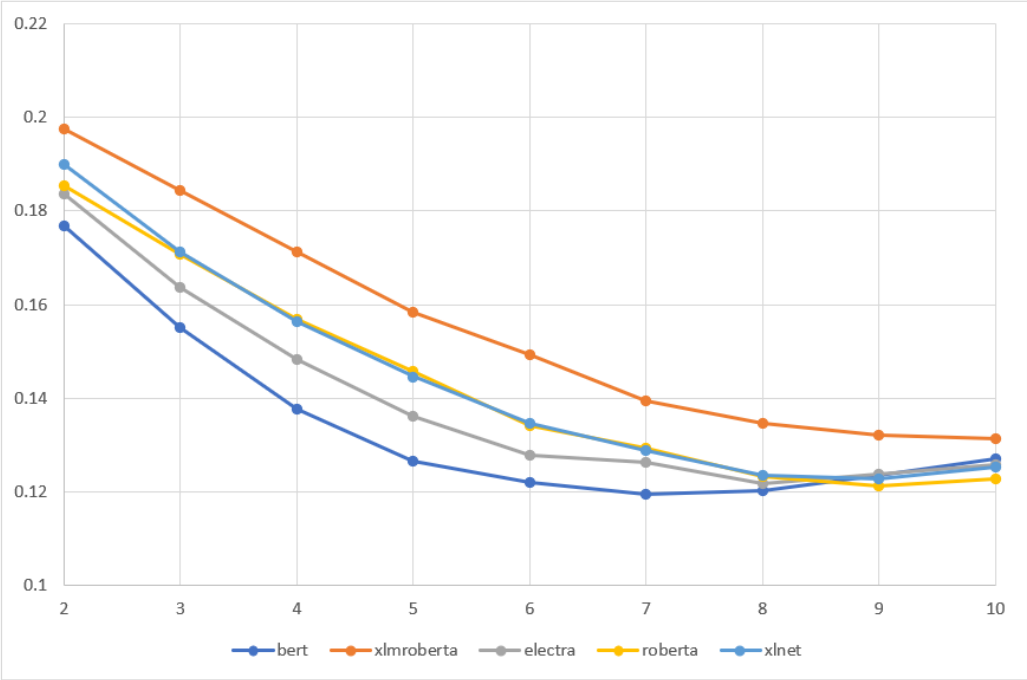

Figure 8.2: TL validation loss curve minimizes at 8-10 fine-tuning epochs.

8.2.2 Experiment 3.3: TL Overfitting Assessment

In experiment 3.3, I analyzed accuracy and validation loss for my TL model hyperparameter grid search, as suggested by [127, 128, 129] to assess how well my TL models generalized and avoided overfitting. Figures 8.1 and 8.2 show the accuracy and validation loss results. See Tables 8.2, 8.3, and 8.4 for the data supporting Experiment 3.3. I calculated the standard deviation from mean for the cross-validation accuracy scores for BERT at the 9th epoch, and found the score was within ± 0.00062, indicating the accuracy was stable across all folds and generalizes well without overfitting. These results indicate that the TL models I created with the 5e-5 learning rate and fine-tuned for 9 epochs were likely to generalize well to other datasets of similar size and class composition, as reported in [22]. Moreover, given that my TL models were the most accurate of all models assessed, they became the backbone of my ensemble research, with the most accurate ensembles being those created with TL algorithms, as discussed in Chapter 9.

8.3 Discussion

The results in this chapter complete the answer to RQ2, in that I identified the most accurate ML classifiers for EMDISM in Chapter 6, I identifed the most accurate DL classifiers for EMDISM in Chapter 7, and I identifed the most accurate TL algorithms for EMDISM in this chapter. As a result of the experiments described in this chapter, I gained several valuable insights and also contributed to the overall body of knowledge for EMDISM researchers. First, I found that BERT was well-suited to the task of EMDISM, consistently yielding the most accurate results for models fine-tuned on the HT dataset, across all cross-validation folds and across all hyperparameter combinations considered in my research. Thus, BERT became the baseline for comparison for my ensemble algorithms, and even formed the backbone of my cascading and cascading/switching ensembles.

I also found that TL algorithms in general are well-suited to EMDISM applications, yielding comparable results, and also aligning closely in their behavior across a hyperparameter grid search. All five TL models required approximately the same number of fine-tuning epochs (8-10) to reach a plateau where the increase in accuracy was offset by overfitting to the dataset. All five TL models, also provided more accurate results with the highler 5e-5 learning rate. All five models were more accurate than any other classifiers I assessed, and thus these models formed the core of many of my ensembles, with the three most accurate, BERT, ELECTRA, and RoBERTa leveraged most extensively in my ensembles.

CHAPTER 9: ENSEMBLE EXPERIMENTS AND RESULTS

9.1 Singleton Model Comparison Summary

In order to investigate ensemble approaches for EMDISM, I first investigated individual candidate ML, DL, and TL components. To summarize the findings of Experiments 1, 2, and 3, in support of my research framework sections A and B, as well as RQ1 and RQ2, I observed the following:

- **Experiment 1:** The Decision Tree classifier was the most accurate ML classifier, with an average accuracy of 81.05%. As this is the most accurate algorithm after the TLs, I included this algorithm in voting, weighted voting, and cascading/switching ensembles.

- **Experiment 2:** After determining that the custom embedder created using the entire HT dataset was the most accurate embedder, I found that the C-LSTM neural network was the most accurate DL classifier, with an average accuracy of 78.85% using the custom embedder. As C-LSTM was less accurate than decision trees, I excluded it from most of my ensembles, although I did use C-LSTM in one simple voting ensemble and the complete set of DL algorithms for several other ensembles for completeness and to assess whether they may offer some unexpected boost in accuracy.

- **Experiment 3:** In 3.1, I determined that the Devlin et al. [34] recommendations were conservative and fine-tuning the base models to 8-10 training epochs was a better solution, as determined in 3.2. In 3.3, I found that the comparison of accuracy and validation loss curves provided high confidence that my models fine-tuned on the HT dataset were not overfitted and should generalize

well to other Twitter datasets with the same emotion classes. Furthermore, I found that the BERT models were the most accurate at every epoch, with an average accuracy of 87.85% at the 9th epoch. Given that BERT was the most accurate TL algorithm, it served as a baseline against which to compare my ensemble algorithms, and BERT, alone as well as with other selections of TLs, also formed the core of numerous ensembles. BERT was only excluded in ensembles where TL algorithms were excluded for the sake of completeness of testing, like All_DLs and All_MLs.

9.2 Experiment 4: Ensemble Comparisons

The ensembles explored in this section align with my research framework sections B and C as shown in Figure 1.1 and address RQ3 for the ensemble classifier assembly and Evaluation. I created 21 ensembles, including simple and weighted voting variants, cascading and cascading/switching variants. Implementation details for each of the ensembles are provided in the following sections. Ensembles are explained in more detail in the relevant categorical sections in 9.2.1, 9.2.3, 9.2.4, and 9.2.5.

9.2.1 Ensembles 4.1-4.9: Simple Voting Ensembles

Simple voting ensembles are perhaps the easiest to implement, as they require identifying appropriate classifiers to include in the ensemble, creating predictions with those classifiers, then pooling the predictions for each sample and assigning the prediction as the class with the most votes for each sample. For all simple voting algorithms, I used the following pooling/prediction process:

1. Stage all predictions for each seed in one unified file by tweet ID, with each algorithm having its own column for predictions.

2. Iterate through each test sample's ID and append votes from each selected algorithm to a list.

3. Select the item with the most votes as the ensemble's prediction. In case of ties, the first item in the list was accepted as the prediction.

Tie-breaking choices were based on classifier analysis ordering — the prediction first added to the list would win in case of tie. I argue that this approach aligns well with the spirit of ensembles, given my approach to assembling the list of predictions. As broad categories of predictions, the ML algorithms were generally less accurate than the DL algorithms, which were generally less accurate than the TL algorithms. Given that my prediction pooling approach is ordered to always append the predictions in order from ML to DL to TL, this implies that the predictions generated should always favor the "novel" selections of the less accurate classifiers. Future work could assess whether a random tie-breaker from the top n tied predictions is more accurate.

I created the simple voting ensembles in the following list and compared their accuracy to BERT, the most accurate singleton transformer:

- **Ensemble 1:** All Models described within Experiments 1, 2, and 3 of this chapter. Used as a baseline to confirm code was working as expected and predictions were within valid classes.

- **Ensemble 2:** BERT_XLMR_XLNET - the most accurate and two least accurate transformers. Intended to assess whether the least accurate TLs combined with BERT would offer more accurate predictions than BERT alone.

- **Ensemble 3:** CLSTM_All_TLs - the most accurate DL and all TLs. Included to assess whether the most accurate DL classifier was a valid choice for ensemble inclusion.

- **Ensemble 4:** Dectree_All_TLs - the most accurate classifier outside of the TLs. Combined all TLs and most accurate algorithm from outside the TL category.

- **Ensemble 5:** Dectree_BERT_ELECTRA - the most accurate non-TL and the two most accurate TLs. Combined best in class across ML and TL.
- **Ensemble 6:** All_DLs - all DL classifiers. Included to assess whether a DL ensemble would be more accurate than C-LSTM alone.
- **Ensemble 7:** All_DLs_All_TLs - all DL classifiers and all TL classifiers. Combines all classifiers from 2 most accurate categories of algorithms, DL and TL.
- **Ensemble 8:** All_MLs - all ML classifiers. Included to assess whether a ML ensemble would be more accurate than decision tree alone.
- **Ensemble 9:** All_TLs - all TL classifiers. Combines all algorithms in most accurate category, TL.

Table 9.1 shows the average accuracy of each of the simple voting ensembles I created above, with the BERT baseline included for comparison. Of the 9 simple voting ensembles compared, 4 outperformed the BERT baseline, especially those which focused on the most accurate TL classifiers. Furthermore, Dectree_All_TLs was the most accurate of the simple voting classifiers, indicating the value of including different types of classifiers in one ensemble. See Table 9.7 and Table 9.8 for descriptions of the components for each ensemble.

9.2.2 Component Analysis for Ensemble Weighting

In support of weighted ensembles I created in Experiment 4, I developed the following pair of hypotheses to establish an acceptable accuracy threshold for including an algorithm in my weighted voting ensembles:

- $H_{4.0}$: All DL and ML algorithms assessed are no more accurate than TL algorithms for predicting the fear and surprise classes in the HT dataset.

Table 9.1: Ensembles 1-9 - Comparing simple voting ensembles, with those above baseline emphasized.

Ensemble Type	**Composition**	**Avg. Accuracy**
Simple Voting	All Models	83.17%
	BERT_XLMR_XLNET	**87.90%**
	CLSTM_All_TLs	81.00%
	Dectree_All_TLs	**89.37%**
	Dectree_BERT_ELECTRA	**88.11%**
	All_DLs	77.83%
	All_DLs_All_TLs	85.28%
	All_MLs	65.26%
	All_TLs	**88.46%**
Baseline	*BERT*	*87.85%*

- $H_{4.1}$: At least one ML or DL algorithm has average accuracy greater than or equal to the average prediction accuracy of the TL algorithms for the least represented classes in the HT dataset.

Note that the average prediction accuracy for the TL algorithms for the **fear** class is 72.33%, and the average prediction accuracy for the TL algorithms for the **surprise** class is 55.96%.

Table 9.2 describes the distribution of samples across each emotion in the HT dataset. See the heatmap in Figure 9.1 for a visual aid to help assess the accuracy within classes for each algorithm. The color scheme is as follows:

- Green: Accuracy higher than 80%.
- Yellow: Accuracy between 50-80%.
- Red: Accuracy below 50%.

Table 9.2: Distribution of emotions in HT dataset

joy	349,419	thankfulness	72,505
sadness	299,412	fear	65,010
anger	261,806	surprise	11,978
love	153,017		

Emotion	Decision Trees	Linear SVC	Logistic Regression	SVM	BiGRU	BiLSTM	C-LSTM	GRU	LSTM	BERT	ELECTRA	RoBERTa	XLM -RoBERTa	XLNet
joy	82.45%	79.35%	78.60%	82.01%	84.89%	85.15%	86.26%	79.33%	83.44%	90.86%	90.57%	90.33%	88.24%	89.92%
sadness	79.87%	64.48%	62.92%	41.76%	76.20%	78.99%	79.98%	71.70%	73.92%	89.50%	87.89%	87.86%	85.80%	87.42%
anger	83.71%	72.14%	68.71%	72.33%	82.45%	79.44%	83.20%	74.62%	82.00%	90.36%	89.89%	88.38%	87.20%	89.26%
love	77.79%	41.31%	35.98%	12.06%	66.24%	63.70%	68.11%	55.03%	60.96%	82.33%	80.53%	78.22%	76.13%	77.97%
thankfulness	81.64%	50.96%	46.46%	44.01%	69.06%	69.74%	71.30%	59.35%	64.47%	84.03%	82.39%	79.13%	77.64%	80.60%
fear	76.45%	28.38%	22.57%	8.45%	57.17%	55.25%	57.17%	42.84%	45.92%	76.94%	74.20%	70.40%	66.89%	73.24%
surprise	81.87%	9.66%	3.06%	1.78%	39.91%	42.61%	43.16%	18.38%	37.43%	64.49%	58.54%	55.22%	46.25%	55.28%

Figure 9.1: Experiment 4.10 - Heatmap of emotion prediction.

Of all ML and DL algorithms assessed in my research, only the Decision Trees classifier exceeds the threshold specified in the $H_{4.1}$ hypothesis. Thus, I must **reject** the $H_{4.0}$ hypothesis and **accept** the $H_{4.1}$ hypothesis, and **the Decision Trees algorithm is therefore appropriate to include in my weighted voting ensembles**.

Notice that most of the algorithms struggle to correctly predict the least sampled emotions in the dataset, especially fear and surprise. With this insight in mind, I designed the weighted ensembles to take advantage of the strengths of the most accurate classifiers overall, BERT, ELECTRA, and RoBERTa, while also considering predictions from Decision Trees, the one classifier with relatively strong accuracy across the most difficult to predict classes.

9.2.3 Ensembles 10-17: Weighted Voting Ensembles

Weighted voting ensembles are more complex to implement than simple voting, as they require not only identifying appropriate classifiers to include in the ensemble but also a weighting scheme developed using insight about the strengths and weaknesses of each algorithm.

For all weighted voting algorithms, I used the following pooling/prediction process:

1. Stage all predictions for each seed in one unified file by tweet ID, with each algorithm having its own column for predictions.

2. Iterate through each test sample's ID and append votes from each selected algorithm to a list, as described in the next list, wherein I discuss how the

weighted votes were added.

3. Select the item with the most votes as the ensemble's prediction. In case of ties, the first item in the list was accepted as the prediction.

I applied insights gained from the heatmap analysis (see Figure 9.1) for the weighting approaches as described in Ensembles 10-17 in the list below:

- **Ensemble 10:** BE_DS - BERT, ELECTRA, and a vote for surprise is added only when Decision Trees classifier predicts surprise. Leverages higher accuracy for surprise from decision tree with two TLs with highest average accuracy.

- **Ensemble 11:** BE_DS2 - BERT, ELECTRA, and two votes for surprise are added only when Decision Trees classifier predicts surprise. Leverages higher accuracy for surprise from decision tree with two TLs with highest average accuracy, while giving additional weight to decision trees for least accurate surprise category.

- **Ensemble 12:** BE_DFS - BERT, ELECTRA, and a vote for fear or surprise is added only when Decision Trees classifier predicts fear or surprise. Leverages higher accuracy for fear and surprise from decision tree with two TLs with highest average accuracy.

- **Ensemble 13:** BE_DFS2 - BE_DFS - BERT, ELECTRA, and two votes for fear or surprise are added only when Decision Trees classifier predicts fear or surprise. Leverages higher accuracy for fear and surprise from decision tree with two TLs with highest average accuracy, while giving additional weight to decision trees for least accurate fear and surprise category.

- **Ensemble 14:** BER_DS - BERT, ELECTRA, RoBERTa and a vote for surprise is added only when Decision Trees classifier predicts surprise. Leverages

higher accuracy for surprise from decision tree with three TLs with highest average accuracy.

- **Ensemble 15:** BER_DS2 - BERT, ELECTRA, RoBERTa and two votes for surprise are added only when Decision Trees classifier predicts surprise. Leverages higher accuracy for surprise from decision tree with three TLs with highest average accuracy, while giving additional weight to decision trees for least accurate surprise category.

- **Ensemble 16:** BER_DFS - BERT, ELECTRA, RoBERTa and a vote for fear or surprise is added only when Decision Trees classifier predicts fear or surprise. Leverages higher accuracy for fear and surprise from decision tree with three TLs with highest average accuracy.

- **Ensemble 17:** BER_DFS2 - BERT, ELECTRA, RoBERTa and two votes for fear or surprise are added only when Decision Trees classifier predicts fear or surprise. Leverages higher accuracy for fear and surprise from decision tree with three TLs with highest average accuracy, while giving additional weight to decision trees for least accurate fear and surprise category.

Table 9.3 shows the average accuracy of each of the weighted voting ensembles I created. The BERT baseline is included as a point of reference. Of the 8 weighted voting ensembles compared, 6 of them outperformed the BERT baseline. Furthermore, BER_DS and BER_DFS were the most accurate of the weighted voting classifiers, with the same accuracy score, indicating that weighting votes for fear may add little value. See Table 9.7 and Table 9.8 for descriptions of the components for each ensemble.

9.2.4 Ensembles 18-19: Cascading Ensembles

Cascading voting ensembles are created by using the output of one or more classifiers to feed the input of one or more additional classifiers. Cascading ensembles

Table 9.3: Ensembles 10-17 - Comparing weighted voting ensembles, with those above baseline emphasized.

Ensemble Type	Composition	Avg. Accuracy
Weighted	**BE_DS**	**88.11%**
	BE_DS2	83.80%
	BE_DFS	**88.11%**
	BE_DFS2	83.49%
	BER_DS	**89.42%**
	BER_DS2	**88.06%**
	BER_DFS	**89.42%**
	BER_DFS2	**88.04%**
Baseline	*BERT*	*87.85%*

require training additional classification models at each node in the cascade. Details on the cascading ensembles I investigated are as follows:

- **Ensemble 18: BERT 4,3** - appended new super-class label to the HT dataset, wherein joy, sadness, anger, and love, the 4 classes with the highest number of samples, were included in one class, and thankfulness, fear, and surprise, the 3 least represented classes were included in a second class. A super-class BERT model was created to classify samples as belonging to one or the other of the 2 super-classes. Predictions from each class were passed to 2 additional BERT models, one trained to predict within the first super-class, and the other trained to predict within the second super-class. Predictions from both classes were then reassembled as a set of comprehensive predictions and were assessed for accuracy. See Figure 9.2 for the architecture of the BERT 4,3 ensemble. See Table 9.4 for the accuracy results for ensembles 18 and 19. The intuition here is that partitioning into smaller groups and training models specific to those groups may yield more accurate results than one model covering every class. This attempts to address the imbalance issue.

- **Ensemble 19: BERT 2,2,3** - appended new super-class label to the HT dataset, wherein joy and sadness, the 2 classes with the highest number of

Table 9.4: Ensembles 18 and 19 - Comparing cascading voting ensembles, with the ensemble outperforming the baseline emphasized.

Ensemble Type	**Composition**	**Avg. Accuracy**
Cascading	**BERT 4,3**	**88.23%**
	BERT 2,2,3	87.54%
Baseline	*BERT*	*87.85%*

samples, were included in one class, the next 2 classes ranked by number of samples, anger and love, were included in a second class, and thankfulness, fear, and surprise, the 3 least represented classes were included in a third class. A super-class BERT model was created to classify samples as belonging to one of the 3 super-classes. Predictions from each class were passed to 3 additional BERT models, one trained to predict within the first super-class, one trained to predict within the second super-class, and the third trained to predict within the third super-class. Predictions from all classes were then reassembled as a set of comprehensive predictions and were assessed for accuracy. See Figure 9.3 for the architecture of the BERT 2,2,3 ensemble. See Table 9.4 for the accuracy results of ensembles 18 and 19. The intuition here is that partitioning into smaller groups and training models specific to those groups may yield more accurate results than one model covering every class. This attempts to address the imbalance issue.

Please note that the BERT 4,3 cascading ensemble outperformed the baseline BERT model for average accuracy, indicating there is value in assembling ensemble classifiers wherein the most accurate model overall is retrained to predict super-classes and sub-classes, rather than treating each class equally from the start. This approach appears to offer modest improvement for the overall classification task when the classes are imbalanced, as they are within the HT dataset. See Table 9.7 and Table 9.8 for descriptions of the components for each ensemble.

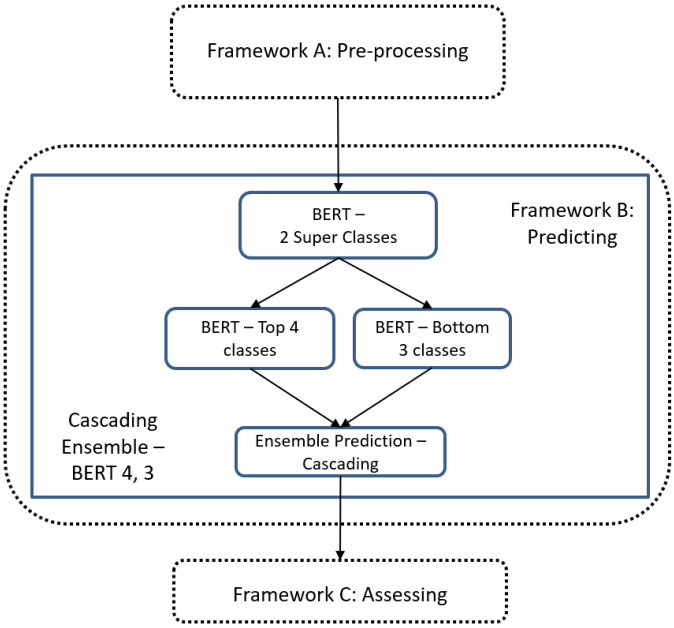

Figure 9.2: Ensemble 18: BERT 4,3 architecture.

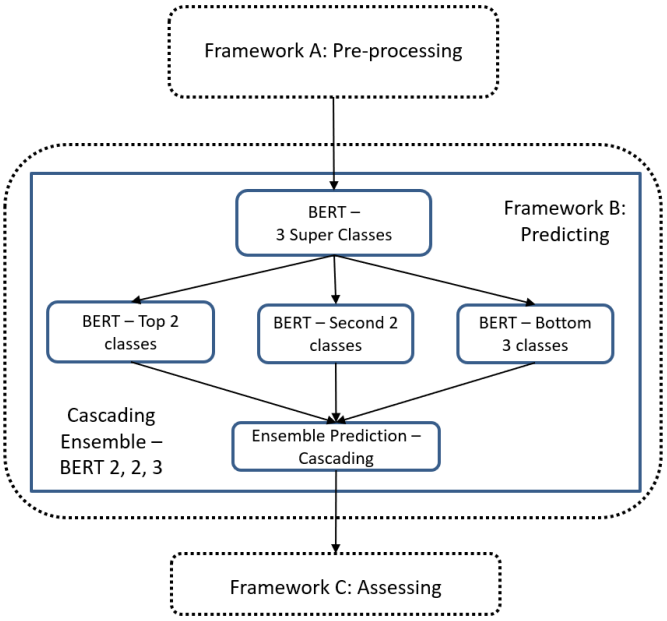

Figure 9.3: Ensemble 19: BERT 2,2,3 architecture.

9.2.5 Ensembles 20-21: Cascading/Switching Ensembles

Cascading voting ensembles are created by using the output of one or more classifiers to feed the input of one or more additional classifiers. Cascading ensembles require training additional classification models at each node in the cascade. Details on the cascading / switching ensembles I investigated are as follows:

- **Ensemble 20: BERT 3, Dectree 4** - appended new super-class label to the HT dataset, wherein joy, sadness, and anger, the 3 classes with the highest number of samples, were included in one class, and love, thankfulness, fear, and surprise, the 4 least represented classes were included in a second class. A super-class BERT model was created to classify samples as belonging to one or the other of the 2 super-classes. Predictions from each class were passed to a BERT model trained to predict within the first super-class and a Decision Trees model trained to predict within the second super-class. Predictions from both classes were then reassembled as a set of comprehensive predictions and were assessed for accuracy. See Figure 9.4 for the architecture of the BERT 3, Dectree 4 cascading/switching ensemble. See Table 9.5 for the accuracy results for ensembles 20 and 21. The intuition here is that partitioning into smaller groups and training models specific to those groups may yield more accurate results than one model covering every class. Switching to decision trees for the least represented classes was informed by the analysis of the heatmap in 9.1, indicating decision tree is the most accurate algorithm for predicting the four least represented classes, fear and surprise and may offer some benefit for love and thankfulness as well.

- **Ensemble 21: BERT 5, Dectree 2** - appended new super-class label to the HT dataset, wherein joy, sadness, anger, love, and thankfulness, the 5 classes with the highest number of samples, were included in one class, and fear and

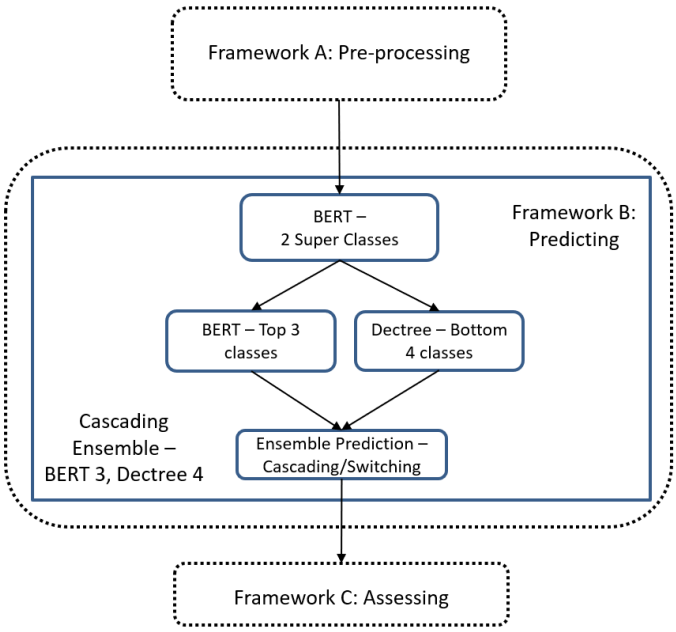

Figure 9.4: Ensemble 20: BERT 3, Dectree 4 architecture.

surprise, the 2 least represented classes were included in a second class. A super-class BERT model was created to classify samples as belonging to one or the other of the 2 super-classes. Predictions from each class were passed to a BERT model trained to predict within the first super-class and a Decision Trees model trained to predict within the second super-class. Predictions from both classes were then reassembled as a set of comprehensive predictions and were assessed for accuracy. See Figure 9.5 for the architecture of the BERT 5, Dectree 2 cascading/switching ensemble. See Table 9.5 for the accuracy results for ensembles 20 and 21. The intuition here is that partitioning into smaller groups and training models specific to those groups may yield more accurate results than one model covering every class. Switching to decision trees for the least represented classes was informed by the analysis of the heatmap in 9.1, indicating decision tree is the most accurate algorithm for predicting the two least represented classes, fear and surprise.

The BERT 5, Dectree 2 cascading/switching ensemble outperformed the BERT

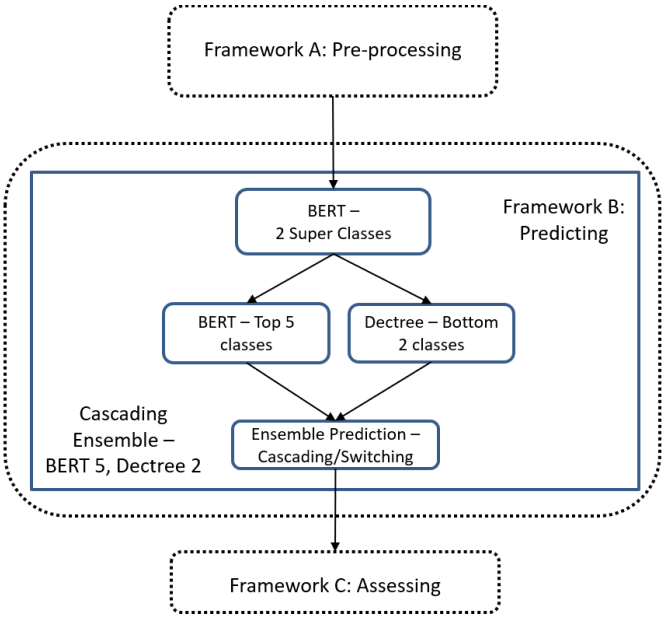

Figure 9.5: Ensemble 21: BERT 5, Dectree 2.

Table 9.5: Ensembles 20 and 21 - Comparing cascading voting ensembles, with the ensemble outperforming the baseline emphasized.

Ensemble Type	**Composition**	**Avg. Accuracy**
Cascading/Switching	BERT 3, Dectree 4	86.37%
	BERT 5, Dectree 2	**88.06%**
Baseline	*BERT*	*87.85%*

baseline, indicating there is value in training ensemble classifiers to segment the dataset into super-classes and sub-classes and switching to a different algorithm with better performance for specific sub-classes. See Table 9.7 and Table 9.8 for descriptions of the components for each ensemble.

9.3 Experiment 5: Comparing ensembles to BERT Baseline

For Experiment 5, I created and assessed the following two hypotheses:

- $H_{5.0}$: Ensemble emotion detection classifiers are no more accurate, no more precise, have no greater precision, and no greater f-measure score than BERT, the most accurate singleton classifier I evaluated for EMDISM.

- $H_{5.1}$: At least one ensemble classifier is more accurate, more precise, has greater recall, or has greater f-measure scores than BERT, the most accurate singleton classifier I evaluated for EMDISM.

Initial assessments compared the average accuracy of various algorithms to each other and to the ensembles I created. To verify accuracy results, I also calculated and compared the weighted precision, recall, and f-measure of the top 5 most accurate ensembles to BERT, the most accurate singleton classifier used as a reference baseline. See Table 9.6 for a comparison across these metrics, ranked in order by weighted f-measure. Note that the 5 most accurate ensembles all had better precision, recall, and f-measure scores than BERT alone, a trend that held across multiple ensemble types. I conducted a single-factor analysis of variance in accuracy between BERT and my five most accurate ensembles, with 5 degrees of freedom, and found that the variance was statistically significant with a p-value of 9.92e-59, and thus I must reject the $H_{5.0}$ hypothesis and accept the $H_{5.1}$ hypothesis. **There exist numerous ensemble classifiers I created and evaluated which exceed the average accuracy, weighted precision, weighted recall, and weighted f-measure scores of BERT, including at least BER_DFS, BER_DS, Dec-**

Table 9.6: Experiment 5 - Comparing ensembles with the BERT baseline. Top 5 ensembles outperform BERT in avg. accuracy as well as weighted precision, recall, and f-measure.

	Average Accuracy	**Weighted Precision**	**Weighted Recall**	**Weighted F-measure**
BER_DFS	89.423%	0.89535	0.89423	0.89441
BER_DS	89.423%	0.89535	0.89423	0.89441
Dectree_All_TLs	89.370%	0.89332	0.89370	0.89311
All_TLs	88.456%	0.88416	0.88456	0.88375
BERT 4,3	88.225%	0.88184	0.88225	0.88175
BERT	87.851%	0.87796	0.87851	0.87804

tree_All_TLs, All_TLs, and BERT 4,3, and the variance in accuracy is statistically significant.

9.4 Discussion

Table 9.7 and Table 9.8 provide a comprehensive list of each ensemble abbreviation and describes the components of that ensemble, and the ensembles are ranked by accuracy.

I completed a detailed summary comparing multiple ensemble classifiers to singleton classifiers. Of the 21 discrete ensembles I created, 12 outperformed BERT, the most accurate singleton classifier I analyzed. See Figure 9.6 to see how the various ensembles compare to BERT. As part of my research, I answered questions about how various embedders interact with DL classifiers, improved hyperparameter optimization recommendations, identified numerous ensembles appropriate to granular EMDISM, and validated that my models and ensembles should generalize well to other short text social media datasets. In the next chapter, I discuss in more detail the conclusions and contributions of my research, then end with proposed topics for future work.

Table 9.7: List of ensembles tested with accuracy above baseline, ranked by accuracy. SV-simple voting, WV-weighted voting, C-cascading, CS-cascading/switching, BL-baseline.

Ensemble	**Accuracy**	**Type**	**Description**
BER_DS	**89.42%**	WV	BERT, ELECTRA, RoBERTa with Decision Trees predictions counted only for surprise
BER_DFS	**89.42%**	WV	BERT, ELECTRA, RoBERTa with Decision Trees predictions counted only for fear and surprise
Dectree _All_TLs	**89.37%**	SV	Decision Trees and all TL predictions get equal votes
All_TLs	**88.46%**	SV	All TL predictions get equal votes
BERT 4,3	**88.23%**	C	BERT superclass to separate BERT models trained with top 4 and bottom 3 classes
Dectree _BERT _ELECTRA	**88.11%**	SV	Decision Trees, BERT, and ELECTRA get equal votes
BE_DS	**88.11%**	WV	BERT, ELECTRA, with Decision Trees predictions counted only for surprise
BE_DFS	**88.11%**	WV	BERT, ELECTRA, with Decision Trees predictions counted only for fear and surprise
BER_DS2	**88.06%**	WV	BERT, ELECTRA, RoBERTa, with Decision Trees predictions given 2 votes for surprise
BERT 5, Dectree 2	**88.06%**	CS	BERT superclass to BERT model trained for top 5 and Decision Trees model trained for fear and surprise
BER_DFS2	**88.04%**	WV	BERT, ELECTRA, with Decision Trees predictions given 2 votes for fear and surprise
BERT _XLMR _XLNET	**87.90%**	SV	BERT, XLM-RoBERTa, and XLNet get equal votes
BERT	*87.85%*	BL	Baseline BERT singleton

Table 9.8: List of ensembles tested with accuracy below baseline, ranked by accuracy. SV-simple voting, WV-weighted voting, C-cascading, CS-cascading/switching, BL-baseline.

Ensemble	**Accuracy**	**Type**	**Description**
BERT	*87.85%*	BL	Baseline BERT singleton
BERT 2, Second 2, Bottom 3	87.54%	C	BERT superclass to separate BERT models trained with top 2, middle 2, and bottom 3 classes
BERT 3, Dectree 4	86.37%	CS	BERT superclass to BERT model trained with top 3 and Decision Trees model trained for bottom 4
All_DLs _All_TLs	85.28%	SV	All DL models and all TL models
BE_DS2	83.80%	WV	BERT, ELECTRA, with Decision Trees predictions given 2 votes for surprise
BE_DFS2	83.49%	WV	BERT, ELECTRA, with Decision Trees predictions given 2 votes for fear and surprise
All Models	83.17%	SV	All models described in this research get equal votes
CLSTM _All_TLs	81.00%	SV	CLSTM and all TL models get equal votes
All_DLs	77.83%	SV	All DL models get equal votes
All_MLs	65.26%	SV	All ML models get equal votes

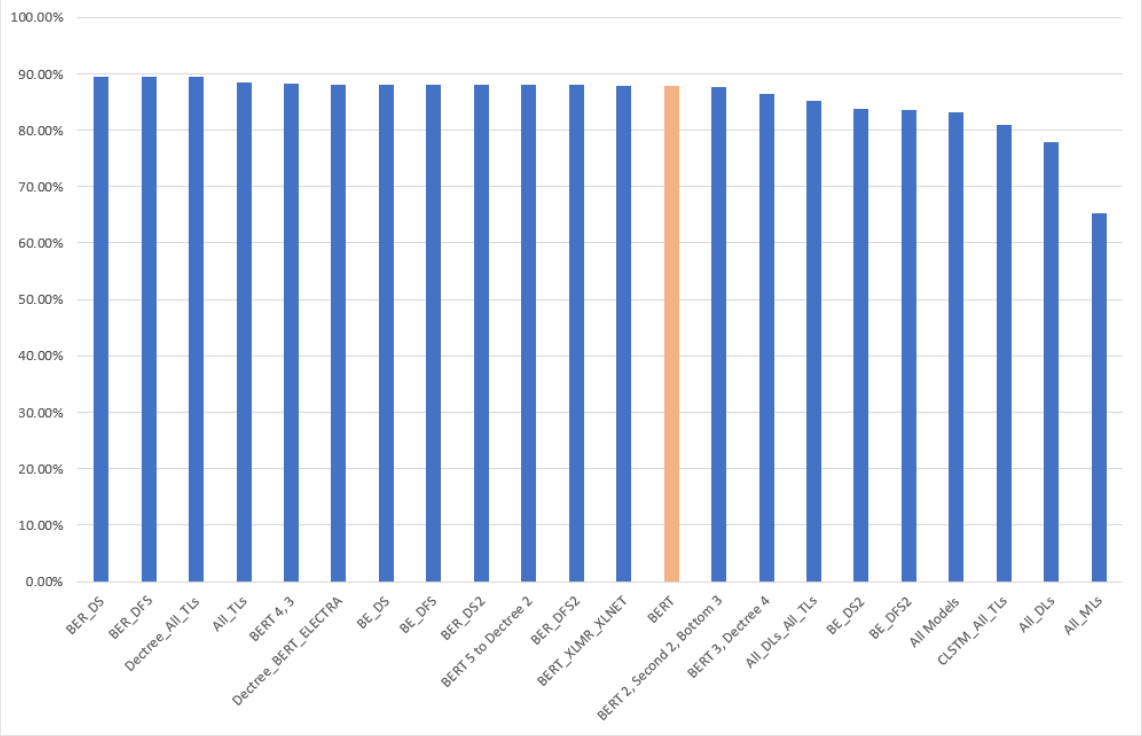

Figure 9.6: Summary: Comparing BERT baseline to ensembles described in this paper.

CHAPTER 10: CONCLUSIONS

Granular emotion detection is a field of research with numerous applications in business, security, health care, social science, and many other fields [1, 5, 6, 7, 3, 4, 8, 10]. Given the massive increase in the use of social media for communication, advertising, politics, and news consumption [145], we need to improve our methods for accurately predicting emotions from short social media texts. Ensemble classifiers provide distinct advantages to offset the weakness of one classifier with the strengths of another, as I demonstrated in my experimental results. In this chapter, I discuss my distinct research contributions, including my research into the efficacy of leveraging embedders from TL libraries with DL classifiers, establish better hyperparameter baselines for EMDISM with TL classifiers, and strongly support the case for creating ensembles for EMDISM, with details for how to build numerous ensembles which outperform state-of-the-art singleton classifiers. This chapter first summarizes conclusions across the different stages of my research, where each stage may connect with multiple research questions. I then frame the conclusions specifically in the context of each research question, and discuss limitations of the research. The chapter concludes with an overview of my research contributions.

10.1 DL Embedder Conclusions

In research, scientists often leverage existing tools in novel ways, an insight that led me to consider applying embedders developed for TL algorithms as the input layer for DL neural networks. I compared BiLSTM, C-LSTM, BiGRU, GRU, and LSTM neural networks using a custom embedder developed using the Keras Tensorflow tokenizer and 179,181 unique tokens from the HT dataset with the standard embedders

included with the BERT, ELECTRA, RoBERTa, XLM-RoBERTa, and XLNet implementations from the HuggingFace Transformers library. The custom embedder was significantly more accurate across all DL models (1.18%-3.63%). Thus, I confirm that the custom embedder created using the complete list of tokens from the dataset yields more accurate DL models than using embedders from BERT, ELECTRA, RoBERTa, XLM-RoBERTA, and XLNet. I recommend DL researchers create a customized embedding layer based on a complete tokenization library from the target dataset, whenever development time and dataset time renders such an approach feasible.

Considering the DL models were less accurate than both ML decision trees and all TL classifiers, I suggest developers will obtain more accurate classifiers by focusing their research on ensembles using decision tree classifiers and TL classifiers. The experimental data in section 9 provide a comprehensive breakdown of the ensembles I created and analyzed and supports this conclusion. Overall, given the accuracy sacrificed and lack of novel strengths of the DL models I compared, ensembles with decision trees and TL classifiers are more likely to yield better accuracy than models with DL classifiers.

These conclusions about DL embedders answer RQ1 in part, by determining that there appears to be little value in leveraging TL embedders to create the input layer for DL algorithms. Instead, a custom embedding layer created from a tokenization vocabulary specific to the dataset examined appears to yield models which more accurately predict emotional classes in EMDISM.

10.2 Transformer Hyperparameter Conclusions

My research required a detailed examination of the recommended hyperparameters suggested for TL models. Devlin et al. [34] provided parameters that they suggested would "...work well across all tasks..."; however, I found that the recommended parameters were suboptimal for EMDISM using the HT dataset. Instead, I found that

the best overall combination of hyperparameters is as follows:

- **Learning rate:** 5e-5
- **Fine-tuning epochs:** 8-10
- **Batch size:** 8 (for single NVIDIA GPU systems)

When comparing my models fine-tuned to 9 epochs to models fine-tuned to 4 epochs, the 9 epochs models were significantly more accurate. For BERT and ELECTRA, 8 or 9 epochs are sufficient to minimize validation loss without overfitting. For RoBERTa, XLM-RoBERTa, and XLNet, the minimum validation loss occurs at 9 or 10 epochs. Given these findings, I suggest 9 epochs offers a satisficing compromise to enable reusability of parameters between TL models.

Regarding batch sizes, Masters and Luchi [144] suggest that the benefits of using smaller batch sizes outweigh increases in processing time caused by using smaller batch sizes. Hence, for developers working with single GPU systems, I suggest there is very little downside to using a batch size of 8 across all TL models, other than modest increases in fine-tuning times across each epoch.

Finally, in the broader landscape of available classification algorithms, I reiterate that TL models, especially ensembles including TL models, are likely to produce the most accurate classifiers. The experimental data in Sections 8.2, 9, and 9.3 support this conclusion.

These conclusions about transformer hyperparameters complete the answer to RQ1 and also answer RQ2. Specifically, whereas using TL embedders with DL algorithms yields little value, fine-tuned TL models are highly appropriate for inclusion in ensemble algorithms. My experiments comparing ML, DL, and TL algorithms in their entirety provide the complete answer to RQ2, in that TL algorithms are the most accurate, with BERT being the most accurate within the TL category. Decision

trees is the next most accurate algorithm and is appropriate to include in ensemble algorithms as well.

10.3 Ensemble Conclusions

My research created and compared 21 ensemble classifiers. Of these ensemble classifiers, 12 were more accurate than the BERT TL classifier, and the top 5 improved on every metric of BERT, including average accuracy and weighted precision, recall, and f-measure. 6 of 12 were weighted voting algorithms, wherein votes for the least represented classes were added from the Decision Trees classifier, the classifier with the best success rate in predicting the least represented categories. 4 of the top 12 ensembles were simple voting ensembles, wherein I focused on the most accurate classifiers overall. 2 of 12 were cascading or cascading/switching ensembles. In other words, of every broad category of ensemble I tested, each category yielded at least one ensemble that worked better than BERT, the most accurate singleton classifier.

From these results, I conclude that ensemble classifiers offer advantages in granular emotion detection in social media. My most accurate ensembles used weighted voting or simple voting to provide more accurate predictions than any single classifier, and the success of the BERT 4,3 classifier indicates that there is even value in creating ensembles of the same classifier to separate predictions into super-classes prior to making final predictions with more granular models of the same primary algorithm. In short, ensembles offer statistically significant advantages in EMDISM accuracy, with modest investments of additional time spent training more granular models.

My findings about ensemble classifiers, indicating that there are numerous ensembles which are more accurate and with better precision, recall, and f-measure, answers RQ3 in its entirety. Ensembles combining the most accurate single classifiers consistently provide more accurate results than one algorithm alone, as demonstrated with my ensembles such as BER_DS and Dectree_All_TLs, and cascading ensembles of the same algorithm can outperform singletons as well, as shown in BERT 4,3. In

short, properly designed ensembles often do perform better than singleton classifiers for multi-class emotion detection, and I have provided numerous examples and detailed guidelines for their creation.

10.4 Research Question Summary

The previous sections framed conclusions across the different stages of my research, where each stage may connect to multiple research questions. This section presents the conclusions framed more specifically to my overall research questions.

RQ1: How can we best integrate transformers, like BERT [34], ELECTRA [38], RoBERTa [35], XLM-R [39], and XLNet [40], and their embedders in our ensemble algorithms? I found there was no advantage to using TL *embedders* to create the input layer for my DL algorithms, whereas TL *algorithms*, on the other hand, were far more accurate than all DL algorithms, and eventually formed the backbone for my ensemble research. BERT was the most accurate single classifier I examined, and I was able to establish and share better hyperparameters for my peers to use in EMDISM research when fine-tuning TL base models.

RQ2: Which ML, DL, and TL classifiers are most accurate for use in a voting ensemble classification approach for EMDISM? After extensive comparisons with 10-fold cross validation across all algorithms, I determined the following:

- Decision trees was significantly more accurate than all other ML algorithms, and was even more accurate than all DL algorithms. Decision trees was also the most accurate algorithm for predicting fear, the least represented class, and was second only to BERT in predicting surprise. Thus, Decision trees was included in many of my ensembles.

- DL classifiers were more accurate than all ML classifiers except decision trees, but were less accurate than all TL algorithms. Thus, DL classifiers were excluded from most ensembles, with only a few references included to assess

whether some unexpected benefit may be gained.

- As previously stated, TL algorithms were the most accurate of all classifiers I compared and formed the backbone of my ensemble research, with BERT and ELECTRA being the most accurate. And ensembles with decision trees were frequently more accurate than BERT as well.

- As part of my examination of TL algorithms, I discovered and shared better hyperparameters for use in EMDISM for all TL classifiers I examined.

RQ3: Do ensemble classifiers perform better than singleton classifiers for multi-class emotion detection, and if so, which ensembles are most effective? I created and compared 21 ensemble classifiers using simple voting, weighted voting, cascading, and cascading switching ensembles. Of these ensembles, more than half were more accurate than BERT, the most accurate singleton classifier, and at least the five most accurate improved on BERT's predicitions by a statistically significant margin. **In short, ensembles can frequently offer more accurate predictions than singleton classifiers for EMDISM.**

10.5 Limitations

There are a number of limitations or additional considerations that scope and contextualize this research. This includes potential exploration of different or related techniques and architectures, ethical considerations, and evaluation considerations. Recent research by Chowdhery et al., for example, describes the Google Pathways Language Model (PaLM) system, a transformer-based language model trained on more than 6,000 tensor processing unit (TPU) pods with 540 billion parameters [146]. The authors describe numerous benchmarks with which they evaluated PaLM's performance, and discuss how the model offers performance either equal or exceeding human performance for numerous complex text-analysis tasks, including question answering, common-sense reasoning, reading comprehension, natural language infer-

ence, and other tasks described in the SuperGLUE benchmarks [147] of complex NLP tasks. A careful examination of the PaLM paper reveals there is no mention of any emotion detection or sentiment analysis applications for this research. Future research could examine how well the PaLM models generalize to EMDISM tasks, although a comparison in Table 34, Appendix F.2 of the PaLM paper illustrates how the model completes prompts can vary widely in emotional tone from the actual completion by human participants, suggesting from inference that PaLM may struggle to understand emotional nuances, especially in tone and toxicity. Regardless, a comparison of classification performance between PaLM and my EMDISM models using identical datasets would be required to understand which method is more accurate for EMDISM. Furthermore, my ensemble approach could be restructured as a monolithic neural network or transformer architecture which may present some advantages in training new models but could likely suffer from the explainability problem often present in neural networks.

The PaLM paper also describes several ethical considerations which are relevant and bear consideration in applying NLP research for EMDISM, including the possibility of reinforcing or aggravating stereotypes in text generation algorithms [148], unintentionally retaining and disclosing private information through model memorization [149], or leading to negative consequences through careless or malicious application of knowledge gained via NLP machine learning [150]. Weidinger et al. [150] go even further, describing six broad risk categories of ethical concerns, including:

1. Discrimination, Exclusion and Toxicity
2. Information Hazards
3. Misinformation Harms
4. Malicious Uses
5. Human-Computer Interaction Harms

6. Automation, Access, and Environmental Harms

In light of these concerns, I acknowledge that EMDISM techniques described herein have the potential to be used for good (e.g. detecting violent emotions and intervening before harm may come to the post originator or others) or ill (e.g. creating posts which are designed to aggravate discord between users based on assessment of emotions inspired by prior posts). Furthermore, the potential for abuse of these kinds of techniques is another concern. Consider a hypothetical case wherein a user's posts are flagged as suggestive of violence and actions are taken to confront the user, which may then provoke a violent response leading to harm to the user or law enforcement personnel. In this case, can we really know for certain whether the user would have become violent on their own or if their violence was a reaction to an imposition of consequences before a harmful action was committed? At the very least, there is an ethical imperative to consider carefully how to apply EMDISM technology on a case-by-case basis to ensure applications are based on sound scientific principles and tailored to avoid harm to every person with whom the technology is engaged. As discussed in [37], before scientific techniques are acceptable for inclusion in US court cases as evidence, they must meet 5 criteria, including *calculation of a known or potential error rate*, *empirical testing methods*, *standardized procedures*, *general acceptance by the scientific community*, and *peer-reviewed publication*. Overall, it is important to give careful consideration for how best to leverage the techniques discussed to prevent harm, intentional or otherwise, to the general public.

In terms of evaluation, there are a number of overall sensitivity analysis aspects that can be considered. Accuracy is an important consideration, and as noted previously can form the basis for inclusion of results as legal evidence. Testing for statistical significance does not equate to being completely certain that the results are significant. For example, a p-value threshold of 0.05 indicates that there is a 5% chance of a type 1 error, wherein we reject the null hypothesis in error. However, given that my

calculated p-values were all much smaller than 0.05, I suggest that my conclusions are highly unlikely to represent a type 1 error. Furthermore, given the relatively small increase in accuracy of my ensembles as compared to the BERT baseline (0.38% to 1.57% for the 5 most accurate ensembles), one may argue that the gain in accuracy is not worth the added effort to create an ensemble. Whereas I acknowledge that there are circumstances when speed is more important than accuracy (e.g. knowing that a Tweet indicates an 87.85% likelihood of violence is barely less urgent than an 89.42% likelihood of violence), the work for building an ensemble is highly reusable and does not appreciably add to the processing time for assessing any 1 sample, and conversely, there are instances wherein accuracy is more important than speed. For example, in my hypothetical situation described above, if confronting someone with violent tendencies risks inciting a violent confrontation, we are compelled to be as confident as possible that action is required before acting to mitigate the risk. A complementary consideration is also possible. Specifically, if an emotion detection algorithm or ensemble outperforms random selection by even a slight margin, there are clearly cases wherein early recognition and warning authorities of violent emotions may save lives; hence, even with some small risk of error, an ensemble with near 90% accuracy applied in good faith warrants integration in social media platforms to assist in preventing catastrophic acts of violence.

My research was limited to the HT dataset, and was further limited in that only approximately half of the original dataset was available for hydration when I obtained the relevant tweets from Twitter. Furthermore, Twitter datasets which are commonly available are often inconsistently labeled, in that both the labeling methodology and emotions labeled can vary widely from one dataset to the next. To offset the limitation of working with one primary dataset, I performed 10-fold cross-validation testing and assessed the standard deviation from mean across all folds. I also compared the validation loss and accuracy curves for the predictions with the TL classifiers

to identify the point at which overfitting was most likely to occur, using models at about the point where both curves flattened and the validation loss began to climb (at approximately the ninth fine-tuning epoch). Given these considerations, I am confident my results should generalize well to other datasets; however, further testing with datasets using a similar labeling technique for the same emotions is required to confirm the generalization between datasets.

This research focused on the context of single-label classification in fairly short social media texts, which represents a constraint on the context of the results. The HT dataset is limited to one specific label for each tweet and the original 140 character constraint on the Tweet content included. Naturally, however, there are cases where a sample could potentially represent more than one emotion. For example, the same circumstances which frighten a person may also make that person angry. Similarly, joy, love, and thankfulness could also be closely intertwined. In addition, longer text samples could be considered within the overall short-text social media context. As a straightforward example, Tweet length has been doubled to 240 characters. Future research could follow the same analysis process presented here in order to explore the effectiveness of ensemble classification methods for multi-label datasets with longer social media text samples. For example, this could investigate combinations of multi-label [151, 152] and ensemble [101, 27] approaches. This would also require research to develop a a large dataset with longer text samples (e.g., content after the Tweet length limit was doubled to 280 characters) in order to understand potential differences with longer text content and to confirm the efficacy of ensembles for longer samples. Further research could also be conducted to confirm the present findings based on other differing factors between tweets, such as sample topic, ethnicity of the sample author, or code-switching between languages.

I also recognize that I focused on hyperparameter optimization for the TL algorithms, and did not perform rigorous testing for the hyperparameters across the ML

and DL algorithms. In my research, I focus primarily on highly generalizable applications of baseline algorithms to use in ensemble classifiers, rather than a rigorous examination of the best way to fine-tune every individual algorithm. I focused on optimization of the TL algorithm hyperparameters due to the steadily increasing accuracy I noticed in my initial grid search with these algorithms, and in an effort to provide future researchers with a unified set of base hyperparameters which applied broadly to EMDISM using the TL models I fine-tuned. In a similar fashion, my examination of embedders for DL algorithms was designed to establish a generalized approach to initializing the input layers of DL networks and also discover if the tokenization vocabularies of the various TL algorithms may offer some benefit over the customized embedding approach tailored to the complete vocabulary of the HT dataset. Finally, the input process for ML algorithms used the TF-IDF tokenization approach, well understood and frequently applied to ensemble text classification tasks [27, 28, 107]. Given that many of my ensembles performed better than BERT, the most accurate single classifier, my research in ensembles offers advantages as shown by my evaluation metrics; however further research is warranted to examine whether detailed hyperparameter optimization across all algorithms would affect classification accuracy for each individual algorithm, as well as ensembles based on those algorithms.

For my TL algorithms, I used a batch size of 8, as opposed to the batch sizes of 16 or 32 proposed by Devlin et al. [34], due to hardware limitations causing buffer overruns during fine-tuning. Further research is needed to determine whether smaller batch sizes significantly affect the accuracy, precision, recall, and f-measure for TL algorithms alone as well as for ensembles based on TL algorithms.

Finally, some specialized deviations from the original baseline TL models have been studied and made available for inclusion through the HuggingFace Transformer library. These derivations include specialized applications, such as Twitter-specific BERT-based variants [153], detecting hate or irony [142], focusing on COVID-related

tweets [106], or stance detection from political tweets [154]. It is possible that some of these variants may perform even better than the BERT, ELECTRA, RoBERTa, XLM-RoBERTa, and XLNet models which formed the core of my TL algorithm evaluations; however, my intuition is that the accuracy of these models will depend more on the hyperparameters and datasets used for fine-tuning than on the origin base models. Further research can establish the validity of this intuition.

10.6 Research Contributions

My contributions to the field of EMDISM are as follows:

- **Pre-processing:** I created and assessed DL models built using TL embedders and compared these to a custom embedder built specifically on the HT dataset. My results demonstrate the custom embedder yields more accurate results than the TL embedders as the input layer of DL algorithms.

- **Transformer learning - hyperparameters:** I conducted a rigorous examination of TL hyperparameters to determine which provide the most accurate results without overfitting and presented these recommendations. Outcomes provided a more effective combination of learning rates and fine-tuning epochs for the domain of EMDISM, beyond the original researcher's recommendations.

- **Transformer learning - ensembles:** I created and assessed multi-TL ensemble classifiers and reported my findings. Results demonstrate ensembles of TLs can provide more accurate predictions than singleton TL classifiers.

- **Multi-discipline ensembles:** I conducted and reported a rigorous assessment of fifteen different individual classifiers across three different classifier disciplines (ML, DL, and TL) and provided results for individual accuracy as well as for ensembles leveraging all components and combinations thereof, including ensembles combining ML and TL classifiers. Outcomes show that ensembles com-

bining TLs and the most accurate ML classifier are more accurate than the most accurate singleton classifiers.

- **Ensembles:** I conducted and reported a rigorous assessment of EMDISM ensembles including up to 15 different component classifiers, in 21 unique combinations, across 4 ensemble categories. Results demonstrate ensembles can often provide more accurate results than the best singleton classifier, with better precision, recall, and f-measure.

My comprehensive ensemble research has answered valuable questions in the field of granular emotion detection in short texts from social media. I expanded the body of knowledge regarding the use of transformer learning for EMDISM and provided valuable guidance into the most appropriate hyperparameters for fine-tuning TL models, as supported by an extensive grid search of fundamental hyperparameters. The research community can create more accurate models using the most effective combination of learning rate and number of fine-tuning epochs.

Furthermore, I completed the most extensive research to date into using ensembles for more accurate prediction of emotions in short text from social media. I assessed 21 different ensembles and found 12 that worked better than the baseline. Finally, in the next chapter, I suggest future work that should expand our knowledge even more, leading to better business, mental health, and security outcomes.

CHAPTER 11: FUTURE WORK

11.1 Identifying Switching Thresholds

In my research, I found that cascading/switching ensembles may be more accurate than singleton classifiers in some circumstances. When I limited my switching mechanism to the least represented samples in the HT dataset, the accuracy improved by 0.21% points above the baseline BERT classifier. I also found that my weighted voting ensembles which were weighted to consider predictions from Decision Trees for the least represented samples were 0.19% to 1.57% more accurate than the baseline BERT classifier. Given these observations, additional research is warranted to determine if there exists a ratio or percentage beyond which a switching ensemble or weighted voting ensemble is recommended. Additional research is required to identify this switching/weighting threshold.

11.2 Confirming Batch Size Assumptions

Hardware limitations dictated the selection of smaller batch sizes for fine-tuning my TL models, in order to avoid buffer overflows with my NVIDIA RTX 2080 GPU. Bearing in mind that Masters and Luchi [144] suggest the benefits of using smaller batch sizes outweigh increases in processing time caused by using smaller batch sizes, additional research is required to confirm whether batch size selection has a significant impact on common evaluation metrics, including accuracy, precision, recall, and f-measure. A study to conduct a grid search across batch sizes with more robust hardware resources available is warranted to determine whether the batch size has any impact at all on these key metrics in EMDISM. Future experiments may also discover by what percentage smaller or larger batch sizes affect the overall training

time per epoch for transformers.

11.3 Black Box Ensembles and Applied Ensembles

My current research has primarily focused on creating ensembles which integrate separate classifiers by aggregating predictions from separate models into one summarized set of predictions, but this is not the only way to assemble ensembles. Alternate approaches could reassemble selected ensembles into larger black box algorithms, with multiple transformers training their models in parallel and unifying their output in the final layers. This approach would maintain the advantages of ensembles while simplifying their implementation for developers.

Furthermore, one possible next logical step for applying my current research would be to develop multi-use bots which can detect negative emotions and take meaningful action to influence the users exhibiting those emotions in a positive manner. For example, consider the theoretical case wherein an angry consumer is using inflammatory rhetoric on Twitter or other social media sites to incite boycotting a company. A bot could detect this rhetoric, inform live representatives of a company to reach out to the consumer, and also interact with the consumer with automated responses to gain insight into the source of the problem and present the live representative with deescalation techniques. A similar approach could be applied in law enforcement and security settings to try to intervene, defuse, or prevent problematic interactions from escalating to the point where violent acts are committed.

11.4 Hashtag Removal, Sample Length, and Additional Classes

When pre-processing the HT dataset, I chose to remove all hashtags because they were used by Wang et al. [118] to assist in class labeling. Future research could consist of several parts to determine how the hashtag removal affected the performance of each individual classifier and the ensembles I examined. First, new models could be trained and compared using the HT dataset without removing hashtags for each

individual classifier and for ensembles of these classifiers. A comparison of metrics between the models built without hashtags and with hashtags will inform future research regarding whether or not to exclude hashtags in training new models.

Second, a new dataset consisting of tweets hydrated after the character limit for Twitter was doubled to 280 characters could be created and labeled using a similar lexicon-based approach as used for the original HT dataset. The original models I built for my research can be assessed for accuracy after applying the same pre-processing steps described in 4.2 to the new dataset. If we achieve comparable accuracy with the existing models, this will indicate that the models I created generalize well to other datasets and also answer the question of how well the models generalize to longer social media samples, both from Twitter and by extrapolation to other social media sites without a character limit.

Future research could also adapt the HT dataset or create a new dataset with more emotion classes. To complete this work, the lexicon-based approach described by Wang et al. would need to be adapted to create more than 7 labels. As the original approach leveraged 131 unique hashtags into 7 classes, coordination with other experts would be needed to update the lexicon for more fine-grained emotion classes.

11.5 Extending Ensembles to Other NLP Tasks

Finally, ensembles need not be limited to the field of emotion detection in short text from social media. Transformers exist for numerous NLP tasks, including named entity recognition, question answering, document summarization, text generation, translation, and zero-shot classification, and new models and applications are being rapidly developed and released [69]. In fact, the original HuggingFace Transformer paper references "thousands" of published models, and more are added daily. With so many potential applications, one can easily see that ensembles could be studied to improve on existing applications of transformers for other NLP tasks. For example,

as of the date of this document, there are currently more than 10 English document summarization models on the HuggingFace Transformers site. Researchers could build a news curation website to summarize news articles with multiple engines and prompt users to complete surveys about the summarization quality, content, and readability.

Regardless of which of these opportunities I choose to pursue, I plan to continue my research in the field, with the goal of turning predictions into meaningful ways for AI to help improve the usability, safety, and satisfaction of users for automated systems.

REFERENCES

[1] J. Ranganathan, *Emotion Mining from Text and Actionable Pattern Discovery.* PhD thesis, The University of North Carolina at Charlotte, 2020.

[2] L. Yue *et al.*, "A survey of sentiment analysis in social media," *Knowledge and Information Systems*, vol. 60, no. 2, pp. 617–663, 2019.

[3] J. Khan, "Sentiment analysis : Key to empathetic customer service," *Ameyo.*

[4] N. Gupta, M. Gilbert, and G. D. Fabbrizio, "Emotion detection in email customer care," *Computational Intelligence*, vol. 29, no. 3, pp. 489–505, 2013.

[5] J. Wolfe, "Want faster airline customer service? try tweeting," *The New York Times.*

[6] C. Walther, "Sentiment analysis in marketing: What are you waiting for?," *CMS Wire.*

[7] S. Gupta, "Applications of sentiment analysis in business," *Towards Data Science.*

[8] M. De Choudhury, M. Gamon, S. Counts, and E. Horvitz, "Predicting depression via social media," in *Seventh international AAAI conference on weblogs and social media*, 2013.

[9] R. McIlroy-Young and A. Anderson, "From âwelcome new gabbersâ to the pittsburgh synagogue shooting: The evolution of gab," in *Proceedings of the international aaai conference on web and social media*, vol. 13, pp. 651–654, 2019.

[10] J. Mei and R. Frank, "Sentiment crawling: Extremist content collection through a sentiment analysis guided web-crawler," in *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015*, pp. 1024–1027, ACM, 2015.

[11] R. Lara-Cabrera, A. G. Pardo, K. Benouaret, N. Faci, D. Benslimane, and D. Camacho, "Measuring the radicalisation risk in social networks," *IEEE Access*, vol. 5, pp. 10892–10900, 2017.

[12] M. Asif, A. Ishtiaq, H. Ahmad, H. Aljuaid, and J. Shah, "Sentiment analysis of extremism in social media from textual information," *Telematics and Informatics*, vol. 48, p. 101345, 2020.

[13] E. Ferrara, W.-Q. Wang, O. Varol, A. Flammini, and A. Galstyan, "Predicting online extremism, content adopters, and interaction reciprocity," in *International conference on social informatics*, pp. 22–39, Springer, 2016.

[14] F. A. Pujol, H. Mora, and M. L. Pertegal, "A soft computing approach to violence detection in social media for smart cities," *Soft Computing*, vol. 24, no. 15, pp. 11007–11017, 2020.

[15] T. Arango, N. Bogel-Burroughs, and K. Benner, "Minutes before el paso killing, hate-filled manifesto appears online," *The New York Times*, 2019.

[16] A. Sage, "Gunman scorned california garlic festival on social media before mass shooting," *Reuters*.

[17] L. Devillers, L. Vidrascu, and L. Lamel, "Challenges in real-life emotion annotation and machine learning based detection," *Neural Networks*, vol. 18, no. 4, pp. 407–422, 2005.

[18] P. Shaver, J. Schwartz, D. Kirson, and C. O'connor, "Emotion knowledge: further exploration of a prototype approach.," *Journal of personality and social psychology*, vol. 52, no. 6, p. 1061, 1987.

[19] P. Ekman, "Basic emotions," *Handbook of cognition and emotion*, vol. 98, no. 45-60, p. 16, 1999.

[20] T. D. Kemper, "How many emotions are there? wedding the social and the autonomic components," *American journal of Sociology*, vol. 93, no. 2, pp. 263–289, 1987.

[21] A. Hassan, A. Abbasi, and D. Zeng, "Twitter sentiment analysis: A bootstrap ensemble framework," in *2013 international conference on social computing*, IEEE, 2013.

[22] R. H. Frye and D. C. Wilson, "Comparative analysis of transformers to support fine-grained emotion detection in short-text data," in *The Thirty-Fifth International Flairs Conference*, 2022.

[23] D. Maynard, K. Bontcheva, and D. Rout, "Challenges in developing opinion mining tools for social media," *Proceedings of the@ NLP can u tag# usergeneratedcontent*, pp. 15–22, 2012.

[24] G. Vinodhini and R. Chandrasekaran, "Sentiment analysis and opinion mining: a survey," *International Journal*, vol. 2, no. 6, pp. 282–292, 2012.

[25] L. K. Hansen and P. Salamon, "Neural network ensembles," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 10, pp. 993–1001, 1990.

[26] R. Burke, "Hybrid recommender systems: Survey and experiments," *User modeling and user-adapted interaction*, vol. 12, no. 4, pp. 331–370, 2002.

[27] W. M. Lim and H. T. Madabushi, "Uob at semeval-2020 task 12: Boosting BERT with corpus level information," *CoRR*, vol. abs/2008.08547, 2020.

[28] Y. P. Babu and R. Eswari, "Cia_nitt at WNUT-2020 task 2: Classification of COVID-19 tweets using pre-trained language models," *CoRR*, vol. abs/2009.05782, 2020.

[29] P. H. Lai, J. Y. Chan, and K. O. Chin, "Ensembles for text-based sarcasm detection," in *2021 IEEE 19th Student Conference on Research and Development (SCOReD)*, pp. 284–289, IEEE, 2021.

[30] Z. Yun-tao, G. Ling, and W. Yong-cheng, "An improved tf-idf approach for text classification," *Journal of Zhejiang University-Science A*, vol. 6, no. 1, pp. 49–55, 2005.

[31] M. Taboada, J. Brooke, M. Tofiloski, K. Voll, and M. Stede, "Lexicon-based methods for sentiment analysis," *Computational linguistics*, vol. 37, no. 2, pp. 267–307, 2011.

[32] S. Ghosh, O. Vinyals, B. Strope, S. Roy, T. Dean, and L. Heck, "Contextual lstm (clstm) models for large scale nlp tasks," *arXiv.org*, 2016.

[33] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv.org*, 2014.

[34] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv.org*, 2018.

[35] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv.org*, 2019.

[36] A. Rocha, W. J. Scheirer, C. W. Forsall, T. Cavalcante, A. Theophilo, B. Shen, A. R. Carvalho, and E. Stamatatos, "Authorship attribution for social media forensics," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 1, pp. 5–33, 2017.

[37] R. H. Frye and D. C. Wilson, "Defining forensic authorship attribution for limited samples from social media," in *The Thirty-First International Flairs Conference*, 2018.

[38] K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning, "Electra: Pre-training text encoders as discriminators rather than generators," *arXiv.org*, 2020.

[39] A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. GuzmÃ¡n, E. Grave, M. Ott, L. Zettlemoyer, and V. Stoyanov, "Unsupervised cross-lingual representation learning at scale," *arXiv.org*, 2019.

[40] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, "Xlnet: Generalized autoregressive pretraining for language understanding," *Advances in neural information processing systems*, vol. 32, 2019.

[41] S. Symeonidis, D. Effrosynidis, and A. Arampatzis, "A comparative evaluation of pre-processing techniques and their interactions for twitter sentiment analysis," *Expert Systems with Applications*, vol. 110, pp. 298–310, 2018.

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

[43] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, (Doha, Qatar), pp. 1532–1543, Association for Computational Linguistics, Oct. 2014.

[44] J. Ranganathan, N. Hedge, A. Irudayaraj, and A. Tzacheva, "Automatic detection of emotions in twitter data-a scalable decision tree classification method," in *Proceedings of the RevOpID 2018 Workshop on Opinion Mining, Summarization and Diversification in 29th ACM Conference on Hypertext and Social Media*, 2018.

[45] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

[46] R. E. Schapire, "The strength of weak learnability," *Machine learning*, vol. 5, no. 2, pp. 197–227, 1990.

[47] O. Araque, I. Corcuera-Platas, J. F. Sanchez-Rada, and C. A. Iglesias, "Enhancing deep learning sentiment analysis with ensemble techniques in social applications," *Expert Systems with Applications*, vol. 77, pp. 236–246, 2017.

[48] A. Oussous, A. A. Lahcen, and S. Belfkih, "Impact of text pre-processing and ensemble learning on arabic sentiment analysis," in *Proceedings of the 2nd International Conference on Networking, Information Systems & Security*, p. 65, ACM, 2019.

[49] I. Perikos and I. Hatzilygeroudis, "Recognizing emotions in text using ensemble of classifiers," *Engineering Applications of Artificial Intelligence*, vol. 51, pp. 191–201, 2016.

[50] V. Duppada, R. Jain, and S. Hiray, "Seernet at semeval-2018 task 1: Domain adaptation for affect in tweets," *arXiv.org*, 2018.

[51] P. Burnap, G. Colombo, R. Amery, A. Hodorog, and J. Scourfield, "Multi-class machine classification of suicide-related communication on twitter," *Online social networks and media*, vol. 2, pp. 32–44, 2017.

[52] K. Kowsari, K. Jafari Meimandi, M. Heidarysafa, S. Mendu, L. Barnes, and D. Brown, "Text classification algorithms: A survey," *Information*, vol. 10, no. 4, p. 150, 2019.

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

[54] S. Smetanin, "Emosense at semeval-2019 task 3: Bidirectional lstm network for contextual emotion detection in textual conversations," in *Proceedings of the 13th International Workshop on Semantic Evaluation*, pp. 210–214, 2019.

[55] J. R. Quinlan, *C4.5: programs for machine learning*. Morgan Kaufmann, San Mateo, CA, 1993.

[56] P. Domingos and M. Pazzani, "On the optimality of the simple bayesian classifier under zero-one loss," *Machine Learning*, vol. 29, pp. 103–130, Nov 1997.

[57] Y. Liu, J.-W. Bi, and Z.-P. Fan, "Multi-class sentiment classification: The experimental comparisons of feature selection and machine learning algorithms," *Expert Systems with Applications*, vol. 80, pp. 323–339, 2017.

[58] V. Vapnik, *The nature of statistical learning theory*. Springer, New York, 2000.

[59] Y. Liu and Y. F. Zheng, "One-against-all multi-class svm classification using reliability measures," in *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, vol. 2, pp. 849–854, IEEE, 2005.

[60] Y. Yang, X. Liu, *et al.*, "A re-examination of text categorization methods," in *Sigir*, vol. 99, p. 99, 1999.

[61] W. Lam and Y. Han, "Automatic textual document categorization based on generalized instance sets and a metamodel," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 5, pp. 628–633, 2003.

[62] M. R. Huq, A. Ali, and A. Rahman, "Sentiment analysis on twitter data using knn and svm," *Int J Adv Comput Sci Appl*, vol. 8, no. 6, pp. 19–25, 2017.

[63] K. Nigam, J. Lafferty, and A. McCallum, "Using maximum entropy for text classification," in *IJCAI-99 workshop on machine learning for information filtering*, vol. 1, pp. 61–67, 1999.

[64] Y. Rao, H. Xie, J. Li, F. Jin, F. L. Wang, and Q. Li, "Social emotion classification of short text via topic-level maximum entropy model," *Information & Management*, vol. 53, no. 8, pp. 978–986, 2016.

[65] X. Xie, S. Ge, F. Hu, M. Xie, and N. Jiang, "An improved algorithm for sentiment analysis based on maximum entropy," *Soft Computing*, vol. 23, no. 2, pp. 599–611, 2019.

[66] L. Zhang, S. Wang, and B. Liu, "Deep learning for sentiment analysis: A survey," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 8, no. 4, p. e1253, 2018.

[67] "Keras documentation," 2019.

[68] Y. Wu *et al.*, "Google's neural machine translation system: Bridging the gap between human and machine translation," 2016.

[69] T. Wolf *et al.*, "Huggingface's transformers: State-of-the-art natural language processing," 2020.

[70] Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio, "Object recognition with gradient-based learning," in *Shape, contour and grouping in computer vision*, pp. 319–345, Springer, 1999.

[71] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[72] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[73] K. S. Tai, R. Socher, and C. D. Manning, "Improved semantic representations from tree-structured long short-term memory networks," *arXiv.org*, 2015.

[74] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[75] Y. Zhu *et al.*, "Aligning books and movies: Towards story-like visual explanations by watching movies and reading books," in *Proceedings of the IEEE international conference on computer vision*, pp. 19–27, 2015.

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

[77] S. Nagel, "Common crawl," 2016. Data retrieved from http://https://commoncrawl.org/2016/10/news-dataset-available/.

[78] J. Callan, M. Hoy, C. Yoo, and L. Zhao, "Clueweb09 data set," 2009.

[79] R. Parker, D. Graff, J. Kong, K. Chen, and K. Maeda, "English gigaword fifth edition, linguistic data consortium," *Google Scholar*, 2011.

[80] G. Lample and A. Conneau, "Cross-lingual language model pretraining," *arXiv.org*, 2019.

[81] A. Conneau, G. Lample, R. Rinott, A. Williams, S. R. Bowman, H. Schwenk, and V. Stoyanov, "Xnli: Evaluating cross-lingual sentence representations," *arXiv.org*, 2018.

[82] R. Elshawi, M. Maher, and S. Sakr, "Automated machine learning: State-of-the-art and open challenges," 2019.

[83] K. Murray, J. Kinnison, T. Q. Nguyen, W. J. Scheirer, and D. Chiang, "Auto-sizing the transformer network: Improving speed, efficiency, and performance for low-resource machine translation," *CoRR*, vol. abs/1910.06717, 2019.

[84] L. Yang and A. Shami, "On hyperparameter optimization of machine learning algorithms: Theory and practice," *Neurocomputing*, vol. 415, pp. 295–316, 2020.

[85] S. Abreu, "Automated architecture design for deep neural networks," *CoRR*, vol. abs/1908.10714, 2019.

[86] P. Koch, B. Wujek, O. Golovidov, and S. Gardner, "Automated hyperparameter tuning for effective machine learning," in *proceedings of the SAS Global Forum 2017 Conference*, pp. 1–23, SAS Institute Inc. Cary, NC, 2017.

[87] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization.," *Journal of machine learning research*, vol. 13, no. 2, 2012.

[88] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of computer and system sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[89] M. D. Cao and I. Zukerman, "Experimental evaluation of a lexicon-and corpus-based ensemble for multi-way sentiment analysis," in *Proceedings of the Australasian Language Technology Association Workshop 2012*, pp. 52–60, 2012.

[90] A. Bickerstaffe and I. Zukerman, "A hierarchical classifier applied to multi-way sentiment detection," in *Proceedings of the 23rd international conference on computational linguistics*, pp. 62–70, Association for Computational Linguistics, 2010.

[91] A. Liaw, M. Wiener, *et al.*, "Classification and regression by randomforest," *R news*, vol. 2, no. 3, pp. 18–22, 2002.

[92] X. Fang and J. Zhan, "Sentiment analysis using product review data," *Journal of Big Data*, vol. 2, no. 1, p. 5, 2015.

[93] C. Baziotis, N. Athanasiou, P. Papalampidi, A. Kolovou, G. Paraskevopoulos, N. Ellinas, and A. Potamianos, "Ntua-slp at semeval-2018 task 3: Tracking ironic tweets using ensembles of word and character level attentive rnns," *arXiv.org*, 2018.

[94] N. F. Da Silva, E. R. Hruschka, and E. R. Hruschka Jr, "Tweet sentiment analysis with classifier ensembles," *Decision Support Systems*, vol. 66, pp. 170–179, 2014.

[95] R. Xia, C. Zong, and S. Li, "Ensemble of feature sets and classification algorithms for sentiment classification," *Information Sciences*, vol. 181, no. 6, pp. 1138–1152, 2011.

[96] G. Wang, J. Sun, J. Ma, K. Xu, and J. Gu, "Sentiment classification: The contribution of ensemble learning," *Decision support systems*, vol. 57, pp. 77–93, 2014.

[97] H. Al-Omari, M. Abdullah, and N. Bassam, "EmoDet at SemEval-2019 task 3: Emotion detection in text using deep learning," in *Proceedings of the 13th International Workshop on Semantic Evaluation*, (Minneapolis, Minnesota, USA), pp. 200–204, Association for Computational Linguistics, June 2019.

[98] A. Chatterjee, K. N. Narahari, M. Joshi, and P. Agrawal, "SemEval-2019 task 3: EmoContext contextual emotion detection in text," in *Proceedings of the 13th International Workshop on Semantic Evaluation*, (Minneapolis, Minnesota, USA), pp. 39–48, Association for Computational Linguistics, June 2019.

[99] T. Yue, C. Chen, S. Zhang, H. Lin, and L. Yang, "Ensemble of neural networks with sentiment words translation for code-switching emotion detection," in *CCF International Conference on Natural Language Processing and Chinese Computing*, pp. 411–419, Springer, 2018.

[100] M. Kang, J. Ahn, and K. Lee, "Opinion mining using ensemble text hidden markov models for text classification," *Expert Systems with Applications*, vol. 94, pp. 218–227, 2018.

[101] S. Thavareesan and S. Mahesan, "Sentiment lexicon expansion using word2vec and fasttext for sentiment prediction in tamil texts," in *2020 Moratuwa Engineering Research Conference (MERCon)*, pp. 272–276, IEEE, 2020.

[102] "Semeval-2018: Internation workshop on semantic evaluation," 2018.

[103] I. Barandiaran, "The random subspace method for constructing decision forests," *IEEE Trans. Pattern Anal. Mach. Intell*, vol. 20, no. 8, pp. 1–22, 1998.

[104] Y. Kim, "Convolutional neural networks for sentence classification," *CoRR*, vol. abs/1408.5882, 2014.

[105] A. Joulin, E. Grave, P. Bojanowski, M. Douze, H. JÃ©gou, and T. Mikolov, "Fasttext.zip: Compressing text classification models," *arXiv.org*, 2016.

[106] M. Müller, M. Salathé, and P. E. Kummervold, "Covid-twitter-bert: A natural language processing model to analyse COVID-19 content on twitter," *CoRR*, vol. abs/2005.07503, 2020.

[107] C. Perrio and H. T. Madabushi, "CXP949 at WNUT-2020 task 2: Extracting informative COVID-19 tweets - roberta ensembles and the continued relevance of handcrafted features," *CoRR*, vol. abs/2010.07988, 2020.

[108] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Information processing & management*, vol. 45, no. 4, pp. 427–437, 2009.

[109] S. Devgan, "100 social media statistics you must know in 2022 [+infographic]," *STATUSBREW*.

[110] C. Beveridge, "33 twitter stats that matter to marketers in 2022," *Hootsuite*.

[111] L. Rossi, C. Neumayer, J. Henrichsen, and L. Beck, "Measuring violence: a computational analysis of violence and propagation of image tweets from political protest," *Social Science Computer Review*, 2021.

[112] M. Suárez-Gutiérrez, J. L. Sánchez-Cervantes, M. A. Paredes-Valverde, E. A. Marín-Lozano, H. Guzmán-Coutiño, and L. R. Guarneros-Nolasco, "Measuring violence levels in mexico through tweets," in *New Perspectives on Enterprise Decision-Making Applying Artificial Intelligence Techniques*, pp. 169–196, Springer, 2021.

[113] D. U. Patton, S. Patel, J. S. Hong, M. L. Ranney, M. Crandall, and L. Dungy, "Tweets, gangs, and guns: A snapshot of gang communications in detroit," *Violence and victims*, vol. 32, no. 5, pp. 919–934, 2017.

[114] CrowdFlower, "Sentiment analysis: Emotion in text," 2017.

[115] T. Josephy, M. Lease, P. Paritosh, M. Krause, M. Georgescu, M. Tjalve, and D. Braga, "Workshops held at the first aaai conference on human computation and crowdsourcing: A report," *AI Magazine*, vol. 35, no. 2, pp. 75–78, 2014.

[116] T. Liu, T. H. Kang, and C. Y. Ken, "Multi-class emotion classification for short texts," 2019. https://github.com/tlkh/text-emotion-classification.

[117] M. Bouazizi and T. Ohtsuki, "Sentiment analysis: from binary to multi-class classification:a pattern-based approach for multi-class sentiment analysis in twitter," *IEEE Access*, vol. 5, pp. 20617–20639, 2017.

[118] W. Wang, L. Chen, K. Thirunarayan, and A. P. Sheth, "Harnessing twitter "big data" for automatic emotion identification," in *2012 International Conference on Privacy, Security, Risk and Trust and 2012 International Confernece on Social Computing*, pp. 587–592, IEEE, 2012.

[119] S. A. Seyeditabari, *Detecting Discrete Emotions in Text Using Neural Networks*. PhD thesis, The University of North Carolina at Charlotte, 2020.

[120] E. Summers, "Hydrator: Documenting the now." https://github.com/DocNow/hydrator, 2020.

[121] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[122] T. C. Rajapakse, "Simple transformers." https://github.com/ThilinaRajapakse/simpletransformers, 2019.

[123] F. Chollet *et al.*, "Keras." https://github.com/fchollet/keras, 2015.

[124] T. pandas development team, "pandas-dev/pandas: Pandas," Feb. 2020.

[125] E. Loper and S. Bird, "Nltk: The natural language toolkit," in *In Proceedings of the ACL Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics.*, 2002.

[126] C. R. Harris *et al.*, "Array programming with NumPy," *Nature*, vol. 585, pp. 357–362, Sept. 2020.

[127] S. Salman and X. Liu, "Overfitting mechanism and avoidance in deep neural networks," *arXiv.org*, 2019.

[128] A. Sedik *et al.*, "Deploying machine and deep learning models for efficient data-augmented detection of covid-19 infections," *Viruses*, vol. 12, no. 7, p. 769, 2020.

[129] I. Safder, S.-U. Hassan, A. Visvizi, T. Noraset, R. Nawaz, and S. Tuarob, "Deep learning-based extraction of algorithmic metadata in full-text scholarly documents," *Information Processing Management*, vol. 57, no. 6, p. 102269, 2020.

[130] J. Hestness, S. Narang, N. Ardalani, G. Diamos, H. Jun, H. Kianinejad, M. M. A. Patwary, Y. Yang, and Y. Zhou, "Deep learning scaling is predictable, empirically," *arXiv.org*, 2017.

[131] E. Loper and S. Bird, "Nltk: The natural language toolkit," in *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1*, ETMTNLP '02, (Stroudsburg, PA, USA), pp. 63–70, Association for Computational Linguistics, 2002.

[132] F. L. dos Santos and M. Ladeira, "The role of text pre-processing in opinion mining on a social media language dataset," in *2014 Brazilian Conference on Intelligent Systems*, pp. 50–54, IEEE, 2014.

[133] R. P. Duin, "Classifiers in almost empty spaces," in *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*, vol. 2, pp. 1–7, IEEE, 2000.

[134] X. Wang, Y. Yang, Y. Zeng, C. Tang, J. Shi, and K. Xu, "A novel hybrid mobile malware detection system integrating anomaly detection with misuse detection," in *Proceedings of the 6th International Workshop on Mobile Cloud Computing and Services*, pp. 15–22, ACM, 2015.

[135] B. Efron, "The efficiency of logistic regression compared to normal discriminant analysis," *Journal of the American Statistical Association*, vol. 70, no. 352, pp. 892–898, 1975.

[136] N. Zainuddin and A. Selamat, "Sentiment analysis using support vector machine," in *2014 international conference on computer, communications, and control technology (I4CT)*, pp. 333–337, IEEE, 2014.

[137] W. Ramadhan, S. A. Novianty, and S. C. Setianingsih, "Sentiment analysis using multinomial logistic regression," in *2017 International Conference on Control, Electronics, Renewable Energy and Communications (ICCREC)*, pp. 46–49, IEEE, 2017.

[138] K. Dashtipour, M. Gogate, A. Adeel, C. Ieracitano, H. Larijani, and A. Hussain, "Exploiting deep learning for persian sentiment analysis," in *International Conference on Brain Inspired Cognitive Systems*, pp. 597–604, Springer, 2018.

[139] Z. Jianqiang, G. Xiaolin, and Z. Xuejun, "Deep convolution neural networks for twitter sentiment analysis," *IEEE Access*, vol. 6, pp. 23253–23260, 2018.

[140] Y. K. Chia, S. Witteveen, and M. Andrews, "Transformer to cnn: Label-scarce distillation for efficient text classification," *arXiv.org*, 2019.

[141] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv.org*, 2014.

[142] F. Barbieri, J. Camacho-Collados, L. Neves, and L. Espinosa-Anke, "Tweeteval: Unified benchmark and comparative evaluation for tweet classification," *arXiv.org*, 2020.

[143] W. Yin, J. Hay, and D. Roth, "Benchmarking zero-shot text classification: Datasets, evaluation and entailment approach," *CoRR*, vol. abs/1909.00161, 2019.

[144] D. Masters and C. Luschi, "Revisiting small batch training for deep neural networks," *CoRR*, vol. abs/1804.07612, 2018.

[145] G. Press, "54 predictions about the state of data in 2021," *Forbes*.

[146] A. Chowdhery *et al.*, "Palm: Scaling language modeling with pathways," *arXiv.org*, 2022.

[147] A. Wang, Y. Pruksachatkun, N. Nangia, A. Singh, J. Michael, F. Hill, O. Levy, and S. Bowman, "Superglue: A stickier benchmark for general-purpose language understanding systems," *Advances in neural information processing systems*, vol. 32, 2019.

[148] E. Sheng, K.-W. Chang, P. Natarajan, and N. Peng, "Societal biases in language generation: Progress and challenges," *arXiv.org*, 2021.

[149] N. Carlini, D. Ippolito, M. Jagielski, K. Lee, F. Tramer, and C. Zhang, "Quantifying memorization across neural language models," *arXiv.org*, 2022.

[150] L. Weidinger, J. Mellor, M. Rauh, C. Griffin, J. Uesato, P.-S. Huang, M. Cheng, M. Glaese, B. Balle, A. Kasirzadeh, Z. Kenton, S. Brown, W. Hawkins, T. Stepleton, C. Biles, A. Birhane, J. Haas, L. Rimell, L. A. Hendricks, W. Isaac, S. Legassick, G. Irving, and I. Gabriel, "Ethical and social risks of harm from language models," *arXiv.org*, 2021.

[151] H. Fei, Y. Zhang, Y. Ren, and D. Ji, "Latent emotion memory for multi-label emotion classification," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, pp. 7692–7699, 2020.

[152] X. Ju, D. Zhang, J. Li, and G. Zhou, "Transformer-based label set generation for multi-modal multi-label emotion detection," in *Proceedings of the 28th ACM International Conference on Multimedia*, pp. 512–520, 2020.

[153] N. Azzouza, K. Akli-Astouati, and R. Ibrahim, "Twitterbert: Framework for twitter sentiment analysis based on pre-trained language model representations," in *International Conference of Reliable Information and Communication Technology*, pp. 428–437, Springer, 2019.

[154] K. Kawintiranon and L. Singh, "Knowledge enhanced masked language model for stance detection," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 4725–4735, 2021.

APPENDIX A: SUPPLEMENTAL EXPERIMENTAL DATA

A.1 TL Accuracy Tables

The following tables contain the accuracy results across all epochs assessed for the TL models.

Table A.1: BERT accuracy scores for all epochs assessed.

Model	**Seed**	**Epoch**	**Acc.**	**Epoch**	**Acc.**	**Epoch**	**Acc.**
BERT	1	2	74.2261%	3	78.2462%	4	81.4623%
BERT	2	2	74.0433%	3	78.0882%	4	81.3312%
BERT	3	2	74.0560%	3	78.0536%	4	81.3510%
BERT	4	2	74.1227%	3	78.0970%	4	81.5016%
BERT	5	2	74.1079%	3	78.0434%	4	81.4576%
BERT	6	2	74.1359%	3	78.1764%	4	81.4060%
BERT	7	2	74.1060%	3	78.1352%	4	81.4505%
BERT	8	2	74.1587%	3	78.1533%	4	81.4054%
BERT	9	2	74.1947%	3	78.1695%	4	81.4678%
BERT	10	2	74.2252%	3	78.1181%	4	81.4183%
BERT	1	5	83.9006%	6	85.5264%	7	86.7672%
BERT	2	5	83.7399%	6	85.2992%	7	86.5471%
BERT	3	5	83.7547%	6	85.3942%	7	86.6329%
BERT	4	5	83.8756%	6	85.4975%	7	86.7496%
BERT	5	5	83.7885%	6	85.4102%	7	86.6238%
BERT	6	5	83.8149%	6	85.4695%	7	86.6925%
BERT	7	5	83.7992%	6	85.4684%	7	86.6612%
BERT	8	5	83.8275%	6	85.4275%	7	86.6818%
BERT	9	5	83.8528%	6	85.4459%	7	86.7021%
BERT	10	5	83.7893%	6	85.3733%	7	86.6463%
BERT	1	8	87.4794%	9	87.9295%	10	88.1397%
BERT	2	8	87.2970%	9	87.7561%	10	87.9724%
BERT	3	8	87.3272%	9	87.7701%	10	87.9869%
BERT	4	8	87.4706%	9	87.9067%	10	88.1240%
BERT	5	8	87.3910%	9	87.8855%	10	88.0878%
BERT	6	8	87.4063%	9	87.8861%	10	88.0911%
BERT	7	8	87.3563%	9	87.8443%	10	88.0548%
BERT	8	8	87.4555%	9	87.9045%	10	88.1026%
BERT	9	8	87.3926%	9	87.8473%	10	88.0683%
BERT	10	8	87.3173%	9	87.7828%	10	88.0012%

Table A.2: ELECTRA accuracy scores for all epochs assessed.

Model	**Seed**	**Epoch**	**Acc.**	**Epoch**	**Acc.**	**Epoch**	**Acc.**
ELECTRA	1	2	72.8481%	3	76.5423%	4	79.4829%
ELECTRA	2	2	72.6217%	3	76.3176%	4	79.2293%
ELECTRA	3	2	72.7096%	3	76.4382%	4	79.3035%
ELECTRA	4	2	72.7571%	3	76.4135%	4	79.3540%
ELECTRA	5	2	72.7288%	3	76.4544%	4	79.4114%
ELECTRA	6	2	72.7547%	3	76.4272%	4	79.4200%
ELECTRA	7	2	72.7192%	3	76.4080%	4	79.4112%
ELECTRA	8	2	72.7657%	3	76.4533%	4	79.3746%
ELECTRA	9	2	72.8646%	3	76.5022%	4	79.3947%
ELECTRA	10	2	72.8297%	3	76.4657%	4	79.3914%
ELECTRA	1	5	81.9690%	6	83.8624%	7	85.2436%
ELECTRA	2	5	81.8126%	6	83.6615%	7	85.0917%
ELECTRA	3	5	81.8407%	6	83.6940%	7	85.0648%
ELECTRA	4	5	81.9613%	6	83.8423%	7	85.2233%
ELECTRA	5	5	81.8959%	6	83.8063%	7	85.1711%
ELECTRA	6	5	81.9327%	6	83.7731%	7	85.1818%
ELECTRA	7	5	81.9028%	6	83.8300%	7	85.1978%
ELECTRA	8	5	81.9091%	6	83.8017%	7	85.1203%
ELECTRA	9	5	81.9195%	6	83.7750%	7	85.1807%
ELECTRA	10	5	81.9061%	6	83.7808%	7	85.1851%
ELECTRA	1	8	86.1817%	9	86.8181%	10	87.0456%
ELECTRA	2	8	86.0432%	9	86.6428%	10	86.8920%
ELECTRA	3	8	85.9784%	9	86.6200%	10	86.8692%
ELECTRA	4	8	86.1979%	9	86.8227%	10	87.0368%
ELECTRA	5	8	86.1089%	9	86.7112%	10	86.9571%
ELECTRA	6	8	86.1309%	9	86.7145%	10	86.9675%
ELECTRA	7	8	86.1504%	9	86.7395%	10	86.9750%
ELECTRA	8	8	86.1171%	9	86.7112%	10	86.9412%
ELECTRA	9	8	86.1034%	9	86.7241%	10	86.9618%
ELECTRA	10	8	86.1108%	9	86.7307%	10	86.9662%

Table A.3: RoBERTa accuracy scores for all epochs assessed.

Model	**Seed**	**Epoch**	**Acc.**	**Epoch**	**Acc.**	**Epoch**	**Acc.**
RoBERTa	1	2	72.6423%	3	75.1701%	4	77.8205%
RoBERTa	2	2	72.4107%	3	75.0042%	4	77.6323%
RoBERTa	3	2	72.4755%	3	74.9822%	4	77.7442%
RoBERTa	4	2	72.4365%	3	75.0089%	4	77.7296%
RoBERTa	5	2	72.4892%	3	75.0754%	4	77.7895%
RoBERTa	6	2	72.4961%	3	74.9641%	4	77.7354%
RoBERTa	7	2	72.4472%	3	75.1122%	4	77.7302%
RoBERTa	8	2	72.4582%	3	75.0927%	4	77.6925%
RoBERTa	9	2	72.5733%	3	75.1600%	4	77.7513%
RoBERTa	10	2	72.5096%	3	75.1089%	4	77.6810%
RoBERTa	1	5	79.8132%	6	82.1308%	7	83.4247%
RoBERTa	2	5	79.5944%	6	81.9492%	7	83.2230%
RoBERTa	3	5	79.7236%	6	82.0470%	7	83.3538%
RoBERTa	4	5	79.6722%	6	82.0426%	7	83.3280%
RoBERTa	5	5	79.7409%	6	82.0418%	7	83.3241%
RoBERTa	6	5	79.7230%	6	82.0712%	7	83.3417%
RoBERTa	7	5	79.6851%	6	82.0745%	7	83.3807%
RoBERTa	8	5	79.7689%	6	82.0484%	7	83.3972%
RoBERTa	9	5	79.6785%	6	82.0734%	7	83.3566%
RoBERTa	10	5	79.7478%	6	81.9885%	7	83.2771%
RoBERTa	1	8	84.8458%	9	85.6934%	10	86.0556%
RoBERTa	2	8	84.5916%	9	85.5236%	10	85.8605%
RoBERTa	3	8	84.7092%	9	85.5668%	10	85.9600%
RoBERTa	4	8	84.7438%	9	85.6454%	10	85.9778%
RoBERTa	5	8	84.7309%	9	85.6028%	10	85.9476%
RoBERTa	6	8	84.7090%	9	85.6080%	10	85.9407%
RoBERTa	7	8	84.7249%	9	85.6088%	10	85.9693%
RoBERTa	8	8	84.7917%	9	85.6704%	10	86.0188%
RoBERTa	9	8	84.7364%	9	85.6047%	10	85.9624%
RoBERTa	10	8	84.6686%	9	85.5701%	10	85.8946%

Table A.4: XLM-R accuracy scores for all epochs assessed.

Model	**Seed**	**Epoch**	**Acc.**	**Epoch**	**Acc.**	**Epoch**	**Acc.**
XLM-R	1	2	70.5172%	3	72.4571%	4	75.2493%
XLM-R	2	2	70.3914%	3	72.3466%	4	75.1119%
XLM-R	3	2	70.3406%	3	72.3950%	4	75.0685%
XLM-R	4	2	70.4752%	3	72.4184%	4	75.1828%
XLM-R	5	2	70.3964%	3	72.4511%	4	75.1578%
XLM-R	6	2	70.4142%	3	72.4236%	4	75.1234%
XLM-R	7	2	70.3681%	3	72.4112%	4	75.1614%
XLM-R	8	2	70.4667%	3	72.4607%	4	75.2361%
XLM-R	9	2	70.5131%	3	72.5217%	4	75.2064%
XLM-R	10	2	70.4563%	3	72.3942%	4	75.1432%
XLM-R	1	5	77.5815%	6	79.3403%	7	81.3299%
XLM-R	2	5	77.5845%	6	79.2458%	7	81.1988%
XLM-R	3	5	77.4362%	6	79.2004%	7	81.1439%
XLM-R	4	5	77.5908%	6	79.3384%	7	81.2683%
XLM-R	5	5	77.5694%	6	79.3309%	7	81.2400%
XLM-R	6	5	77.5276%	6	79.2859%	7	81.2266%
XLM-R	7	5	77.5084%	6	79.3227%	7	81.2321%
XLM-R	8	5	77.6593%	6	79.3779%	7	81.3483%
XLM-R	9	5	77.6117%	6	79.3870%	7	81.3216%
XLM-R	10	5	77.5749%	6	79.2716%	7	81.2092%
XLM-R	1	8	82.7224%	9	83.7006%	10	84.0833%
XLM-R	2	8	82.6334%	9	83.5948%	10	84.0097%
XLM-R	3	8	82.5487%	9	83.5368%	10	83.9572%
XLM-R	4	8	82.7059%	9	83.6887%	10	84.0644%
XLM-R	5	8	82.6570%	9	83.6431%	10	84.0328%
XLM-R	6	8	82.6556%	9	83.6159%	10	83.9954%
XLM-R	7	8	82.6713%	9	83.6385%	10	84.0366%
XLM-R	8	8	82.7405%	9	83.7385%	10	84.1240%
XLM-R	9	8	82.7174%	9	83.6415%	10	84.0679%
XLM-R	10	8	82.6460%	9	83.6008%	10	83.9863%

Table A.5: XLNet accuracy scores for all epochs assessed.

Model	**Seed**	**Epoch**	**Acc.**	**Epoch**	**Acc.**	**Epoch**	**Acc.**
XLNet	1	2	71.7933%	3	75.1990%	4	77.8865%
XLNet	2	2	71.5583%	3	74.9990%	4	77.6766%
XLNet	3	2	71.6141%	3	75.0330%	4	77.7233%
XLNet	4	2	71.6309%	3	75.1086%	4	77.7922%
XLNet	5	2	71.6471%	3	75.1237%	4	77.7659%
XLNet	6	2	71.6410%	3	75.1811%	4	77.7447%
XLNet	7	2	71.6518%	3	75.0732%	4	77.8486%
XLNet	8	2	71.7408%	3	75.1509%	4	77.8362%
XLNet	9	2	71.7205%	3	75.1949%	4	77.8024%
XLNet	10	2	71.7254%	3	75.2064%	4	77.8216%
XLNet	1	5	80.3165%	6	82.2517%	7	83.8753%
XLNet	2	5	80.1110%	6	82.0434%	7	83.6635%
XLNet	3	5	80.1209%	6	82.1528%	7	83.6970%
XLNet	4	5	80.2553%	6	82.2157%	7	83.7767%
XLNet	5	5	80.1967%	6	82.2377%	7	83.7399%
XLNet	6	5	80.1651%	6	82.1775%	7	83.8121%
XLNet	7	5	80.2215%	6	82.1984%	7	83.7780%
XLNet	8	5	80.2662%	6	82.2193%	7	83.7572%
XLNet	9	5	80.2234%	6	82.1696%	7	83.7846%
XLNet	10	5	80.1643%	6	82.1610%	7	83.7159%
XLNet	1	8	85.0769%	9	85.8682%	10	86.1699%
XLNet	2	8	84.9167%	9	85.7283%	10	86.0042%
XLNet	3	8	84.9848%	9	85.7893%	10	86.0787%
XLNet	4	8	85.0233%	9	85.8130%	10	86.1012%
XLNet	5	8	84.9837%	9	85.7772%	10	86.0784%
XLNet	6	8	85.0447%	9	85.8338%	10	86.1298%
XLNet	7	8	85.0307%	9	85.8437%	10	86.1240%
XLNet	8	8	85.0829%	9	85.8413%	10	86.1278%
XLNet	9	8	84.9815%	9	85.7803%	10	86.0781%
XLNet	10	8	84.9472%	9	85.7509%	10	86.0550%

A.2 Top 5 Ensemble Accuracy Tables

The following table contains the average accuracy, weighted precision, weighted recall, and weighted f-measure for the 5 most accurate ensemble algorithms for all folds.

Table A.6: Average accuracy, weighted precision, weighted recall, and weighted f-measure scores for 5 most accurate ensembles.

Model	**Seed**	**Avg.Acc.**	**Wt.Prec.**	**Wt.Rec.**	**Wt.F-meas.**
All_TLs	1	88.5268%	0.884877	0.885268	0.884450
All_TLs	2	88.3930%	0.883545	0.883930	0.883103
All_TLs	3	88.3743%	0.883297	0.883743	0.882922
All_TLs	4	88.5238%	0.884829	0.885238	0.884468
All_TLs	5	88.4499%	0.884110	0.884499	0.883666
All_TLs	6	88.4367%	0.883987	0.884367	0.883557
All_TLs	7	88.4771%	0.884407	0.884771	0.883960
All_TLs	8	88.5178%	0.884793	0.885178	0.884393
All_TLs	9	88.4505%	0.884118	0.884505	0.883684
All_TLs	10	88.4054%	0.883631	0.884054	0.883261
BER_DFS	1	89.5251%	0.896318	0.895251	0.895378
BER_DFS	2	89.3520%	0.894591	0.893520	0.893671
BER_DFS	3	89.3597%	0.894691	0.893597	0.893779
BER_DFS	4	89.4712%	0.895943	0.894712	0.894954
BER_DFS	5	89.4369%	0.895495	0.894369	0.894534
BER_DFS	6	89.4261%	0.895504	0.894261	0.894492
BER_DFS	7	89.4110%	0.895266	0.894110	0.894305
BER_DFS	8	89.4380%	0.895459	0.894380	0.894571
BER_DFS	9	89.4047%	0.895108	0.894047	0.894195

Table A.6 continued from previous page

Model	Seed	Avg.Acc.	Wt.Prec.	Wt.Rec.	Wt.F-meas.
BER_DFS	10	89.4050%	0.895161	0.894050	0.894262
BER_DS	1	89.5251%	0.896318	0.895251	0.895378
BER_DS	2	89.3520%	0.894591	0.893520	0.893671
BER_DS	3	89.3597%	0.894691	0.893597	0.893779
BER_DS	4	89.4712%	0.895943	0.894712	0.894954
BER_DS	5	89.4369%	0.895495	0.894369	0.894534
BER_DS	6	89.4261%	0.895504	0.894261	0.894492
BER_DS	7	89.4110%	0.895266	0.894110	0.894305
BER_DS	8	89.4380%	0.895459	0.894380	0.894571
BER_DS	9	89.4047%	0.895108	0.894047	0.894195
BER_DS	10	89.4050%	0.895161	0.894050	0.894262
Dectree_all_TLs	1	89.4468%	0.894109	0.894468	0.893859
Dectree_all_TLs	2	89.2998%	0.892641	0.892998	0.892394
Dectree_all_TLs	3	89.2819%	0.892403	0.892819	0.892223
Dectree_all_TLs	4	89.4396%	0.894025	0.894396	0.893837
Dectree_all_TLs	5	89.3616%	0.893250	0.893616	0.893012
Dectree_all_TLs	6	89.3822%	0.893451	0.893822	0.893231
Dectree_all_TLs	7	89.3715%	0.893345	0.893715	0.893129
Dectree_all_TLs	8	89.4371%	0.894003	0.894371	0.893796
Dectree_all_TLs	9	89.3459%	0.893089	0.893459	0.892851
Dectree_all_TLs	10	89.3311%	0.892906	0.893311	0.892734
BERT 4,3	1	88.2386%	0.881986	0.882386	0.881890
BERT 4,3	2	88.1597%	0.881220	0.881597	0.881100
BERT 4,3	3	88.2372%	0.881970	0.882372	0.881874

Table A.6 continued from previous page

Model	Seed	Avg.Acc.	Wt.Prec.	Wt.Rec.	Wt.F-meas.
BERT 4,3	4	88.2177%	0.881793	0.882177	0.881698
BERT 4,3	5	88.2109%	0.881673	0.882109	0.881570
BERT 4,3	6	88.2727%	0.882292	0.882727	0.882235
BERT 4,3	7	88.2131%	0.881710	0.882131	0.881654
BERT 4,3	8	88.2526%	0.882129	0.882526	0.882021
BERT 4,3	9	88.2383%	0.881977	0.882383	0.881882
BERT 4,3	10	88.2095%	0.881652	0.882095	0.881590