ISLAMIC UNIVERSITY OF TECHNOLOGY (IUT)

Efficient Ensemble-Based Approaches to Personal Health Mention Detection

Authors

Nuzhat Nower, 180041116

Fida Kamal, 180041208

Alvi Aveen Khan, 180041229

Supervisor

Tareque Mohmud Chowdhury

Assistant Professor

Dept. of CSE, IUT

Co-Supervisor

Tasnim Ahmed

Lecturer

Dept. of CSE, IUT

A thesis submitted in partial fulfilment of the requirements for the degree of B. Sc. Engineering in Computer Science and Engineering

Academic Year: 2021-2022

Department of Computer Science and Engineering (CSE)

Islamic University of Technology (IUT)

A Subsidiary Organ of the Organization of Islamic Cooperation (OIC)

Dhaka, Bangladesh

May 19, 2023

Declaration of Authorship

This is to certify that the work presented in this thesis is the outcome of the analysis and experiments carried out by under the supervision of Tareque Mohmud Chowdhury, Assistant Professor of the Department of Computer Science and Engineering (CSE), Islamic University of Technology (IUT), Dhaka, Bangladesh. It is also declared that neither of this thesis nor any part of this thesis has been submitted anywhere else for any degree or diploma. Information derived from the published and unpublished work of others has been acknowledged in the text and a list of references is given.

Authors:

Nuzhat
Nuzhat Nower

Student ID - 180041116

Fida Kamal
Fida Kamal

Student ID - 180041208

ALVI AVEEN
Alvi Aveen Khan

Student ID - 180041229

Approved By:

Supervisor:

[illegible] 08/06/2023

Tareque Mohmud Chowdhury

Assistant Professor

Department of Computer Science and Engineering (CSE)

Islamic University of Technology (IUT), OIC

Co-Supervisor:

[illegible] 9/6/23

Tasnim Ahmed

Lecturer

Department of Computer Science and Engineering (CSE)

Islamic University of Technology (IUT), OIC

Acknowledgement

We would like to thank our mentor and advisor, **Tareque Mohmud Chowdhury**, Assistant Professor, Department of Computer Science & Engineering, IUT. The inspiration, advice and knowledge he provided were invaluable to this study, and this research would not have been possible without it. He guided us through each phase of the thesis, from the initial conceptualization, through the implementation details and the final reviews, for which we are grateful.

We are also grateful to **Tasnim Ahmed**, Lecturer, Department of Computer Science & Engineering, IUT for his insights, suggestions and constant support. His enthusiasm for this research was on par with ours and had a huge contribution towards its successful completion.

Abstract

An important component of public health surveillance is the analysis of personal health-related posts on social media platforms, known as Personal Health Mention (PHM) Detection. PHM detection is essential to quickly detecting epidemics, allowing health organisations to prepare themselves and warn the general public to take precautionary steps. One of the key complexities of this task is the informal nature of the language used in social media, which makes it difficult to understand their context. The architectures that are capable of discerning context are also computationally expensive to use and often mistrusted in the medical community due to their decision-making strategies being hidden behind a black box.

In this thesis, we address each of these issues separately. We introduce four transformer-based ensemble architectures that have not been previously explored in the PHM domain and show that these architectures can achieve state-of-the-art results across the domain. Combined with computationally efficient training mechanisms, our architectures also use fewer resources than existing ones. Additionally, we provide methods to explain the outputs produced by the architectures in order to address the concerns related to explainability.

Contents

1 Introduction

1.1 Overview

In many countries, there are dedicated government organisations that exist to monitor public health, such as the Centers for Disease Control and Prevention[1] in the USA and the Care Quality Commission[2] in the UK. One of the main services these organisations provide is the detection of new and emerging threats to public health, such as COVID-19. Traditionally, this has been done through a lengthy process of collecting, validating, and analysing health reports. The amount of time this process takes prevents these organisations from being able to detect rapidly spreading health threats, which can lead to dire consequences [56]. With the increasing use of social media platforms throughout the world, online health monitoring has become one of the key ways in which these organisations are able to vastly reduce the amount of time spent on data collection [25].

1.2 Problem Statement

Data from social media platforms like Twitter has been used for the early detection of various infectious diseases [9, 11]. One of the most important steps in this process is the accurate identification of posts that talk about the author's personal health, known as Personal Health Mention (PHM) detection. This is defined as the detection of any post that mentions a health issue being faced either directly by the author or by someone the author knows [37]. For example, the text 'I have a headache' would fall under this category, but the text '7 cures to headaches' would not. The importance of PHM detection has been acknowledged by academia [52, 66], and a large amount of work has already been done in the domain.

[1]https://www.cdc.gov/

[2]https://www.cqc.org.uk/

1.3 Research Challenges

There are several issues that are faced when trying to address the problem stated above. The challenges are varied and occur both with the data that must be used and with the models being created. Addressing these challenges will make the process of creating and training the models easier and also allow the models to attain improved performance. Some of the challenges are discussed below:

1. Noisy Text: Text on social media sites are uniquely different from general text due to the nature of the noise they contain. Social media presents an informal setting which allows for the relaxation of common rules that are usually followed in other forms of textual communication. It is generally accepted that users will disregard minor spelling mistakes or use short forms of common words. In addition to this, the majority of the data in the PHM domain is collected from the social media site Twitter, which limits posts to 280 characters. As a result, users are forced to invent new and unique ways of shortening their posts so as to fully communicate their thoughts within that limit. Such noise makes it difficult to train natural language processing architectures [14].

2. Context Recognition: Informal communication also has a tendency to encourage the use of sarcastic, figurative or hyperbolic language. This can go so far as to become confusing even for other humans [31]. Understanding the context of a post is essential to being able to correctly identify whether it is a PHM or not, and such language makes the task all the more difficult.

3. Resource Usage: Currently, one of the most reliable methods of understanding context is to use models based on the transformer architecture. However, these models are very large and require significantly powerful resources to run. This presents a problem for health organizations which have limited resources that would be better spent on more direct ways of supporting public health. The computational expense of the architecture thus prevents practical applications from using them.

These issues have been addressed by past work primarily in two ways. The issue of unstandardized text is addressed using a variety of data cleaning and text normalisation techniques. The issue of understanding context has been brought up more recently as researchers have begun to acknowledge the presence of figurative language in social media posts. This has resulted in new datasets being created with the explicit intent of identifying such language. In addition, the advent of transformer-based models has made the task of understanding the context of a post significantly easier. The superior capability of such models in understanding context has resulted in their widespread adoption in the PHM domain.

1.4 Thesis Objectives

The objective of this thesis is to address the issues that have been introduced by existing solutions to the mentioned problems. The use of transformer-based models, however, has presented two additional issues. The first is that these models fall under the category of deep-learning models, which tend to be very large. The size of these models puts a significant computational burden on the systems using them. The second issue is with their decision-making strategies. Prior to the use of deep-learning models, the mechanisms that led to the identification of personal health were the result of specific, hand-crafted features chosen by the researchers. This made it possible to explain the reasoning behind the decisions made by the models. Transformer-based models, however, are 'black boxes', in that the decisions they make are not based on features chosen by the researchers but rather on patterns they learn from the data. This makes their decisions far more difficult to explain.

1.5 Thesis Contribution

In this thesis, our work concentrates primarily on three issues:

1. We introduce four ensemble-based architectures that, to the best of our knowledge, have not been studied previously in the domain of PHM. By

examining their performance in comparison to existing work, we show that these architectures achieve state-of-the-art results across the domain.

2. We address the computational resources required by transformer-based models and introduce mechanisms that can reduce the requirements.

3. We explore the issue of explainability and provide a system to understand the results produced by the models.

1.6 Organization of the Thesis

The rest of this thesis is organized as follows: in Chapter 2, a review of existing literature in the PHM domain is presented, along with discussions about their contributions and limitations. In Chapter 3, the datasets that exist in the domain are discussed. Chapter 4 provides a detailed discussion about the different parts of our proposed pipeline, while Chapter 5 presents the experimental findings of our work. Chapter 6 discusses two optimizations techniques we have explored in detail, Chapter 7 presents our ablation students and Chapter 8 discusses the techniques used to make the outputs of the trained models explainable. Finally, Chapter 9 concludes the thesis and discusses possible directions for future work.

2 Literature Review

A variety of mechanisms have been explored by past authors, achieving varying levels of success in PHM detection. This section examines a few of these mechanisms to provide a better understanding of how the past work leads to ours.

2.1 Traditional Machine Learning Approaches

The traditional machine learning approaches that have been used are comparatively straightforward and do not provide information about possible approaches that would be useful anymore. However, they are still being included for reference, but are being combined under one section.

These approaches almost invariably consist of two sections, a feature extraction section and a classification section. To extract features, various methods have been used including using user mentions and emotion keywords [22], n-gram feature extraction [3], and topic modelling [7, 55]. To actually classify the text as a positive or negative personal health mention, the extracted features were passed to a classifier. Popular classifiers in the domain include Support Vector Machines (SVMs) [3, 53], K-Nearest Neighbour (KNN) networks [22] and Decision Trees.

2.2 Deep Learning Approaches

The reliance on handcrafted features was a huge issue for traditional machine learning models, since the process of identifying which features to extract required medical expertise and was cumbersome and time-consuming. Deep-learning models bypassed this need by using word embeddings, which are vectors representing linguistic information about the words in a piece of text. In the domain of PHM, the importance of word embeddings was proven by Iyer et al. [21]. Their research adds to the justification behind the widespread use of word embeddings in the domain. Jiang et al. [23] for example, used Word2Vec [43] word embeddings to train a Long Short-Term Memory (LSTM) network, while Wang et al. [65] used GloVe-based word embeddings [57] along with a bi-directional LSTM (Bi-LSTM) network.

2.3 WESPAD Method

One of the most prominent works in recent years that made use of deep learning algorithms was by Karisani and Agichtein [28], who proposed the Word Embedding Space Partitioning and Distortion (WESPAD) method. This combines lexical, syntactic, word embedding-based, and context-based features. It partitions the word embedding space to generalise from a small amount of training data and distorts the embedding space to effectively detect true health mentions.

2.4 Contextual Word Representations

Biddle et al. [5] was the first to use contextual word representations in the PHM domain. These became popular in the domain of NLP in general due to the introduction of the BERT architecture [12], which is a transformer-based architecture. Initially proposed by Vaswani et al. [63], transform-based architectures use a concept called self-attention which allows the models to learn the relationship between different parts of a sentence. This essentially means that the models are extremely effective in understanding context, which has consequently led to their use in a variety of language-related tasks [1, 13, 38]. Several architectures have since been developed based on the transformer architecture, with varying degrees of changes.

Transformer-based architectures have shown such a significant improvement over previous architectures in the domain of NLP, that simply fine-tuning these pre-trained models to a dataset is often enough to achieve exceptionally good results. In the domain of PHM detection, Khan et al. [32] was one of the first to use a pre-trained transformer-based model. They used the XLNet architecture [67] and fine-tuned it to the HMC2019 dataset. Their work showed that the permutation-based word representations used by the XLNet architecture were more effective than the bi-directional word representations used by the BERT architecture. The authors later also provided a comprehensive comparison of the performance of various pre-trained transformer-based models on the HMC2019 dataset [34], where they concluded that the RoBERTa architecture [40] achieved the best performance.

A similar approach was taken by Naseem et al. [50], who proposed the PHS-BERT pre-trained language model. This model was initialised with the weights from the BERT architecture and then further pre-trained on social media texts collected from Twitter so as to ensure improved performance on downstream tasks that used data from social media. Their model was evaluated on seven Personal Health Surveillance tasks, one of which was PHM detection.

2.5 Adversarial Training

To improve performance further, Khan et al. [35] proposed a new approach to the fine-tuning process. They used Adversarial Training (AT) to generate perturbed examples, which they used to fine-tune the BERT and RoBERTa architectures. Along with this, they used the Barlow Twins (BT) contrastive loss function, which pushed the clean and perturbed examples closer together, thus allowing the models to learn noise-invariant feature representations. In this manner, they improved the robustness of the models against noise. Another work [33] seems to have used the same approach, with no apparent differences.

2.6 CT-BERT + Bi-LSTM

Naseem et al. [49] proposed another architecture which combined several features, including contextual representations, parts-of-speech tags and sentiment distributions. They initialised the BERT architecture with the weights of CT-BERT [45], which was pre-trained on Twitter posts related to the COVID-19 pandemic for a variety of natural language processing tasks, including classification. Due to the training process, the CT-BERT architecture is able to obtain superior performance compared to similar architectures on health-related social media data specifically. The authors further concatenated the context embeddings from this architecture with parts-of-speech tags generated using an English parts-of-speech tagger [6]. This module was included to capture syntactic dependencies in text, which they showed contributed to improved performance. Next, the authors used a Bi-LSTM classifier, the output of which they concatenated with attention scores generated by a symptom-based attention mechanism [68] as well as Valence, Arousal, and Dominance (VAD) sentiment distributions [44]. The symptom-based attention mechanism helped the model learn to pay special attention to symptom keywords, and the VAD sentiment distributions, previously used by Biddle et al. [5], was already known to contribute towards improved performance.

Clearly, the architectures being used in the PHM domain have gotten increasingly large over time. In this thesis, we attempt to address this issue by exploring techniques to reduce the computational resources required by the models while also achieving state-of-the-art results. Additionally, we explore the use of several ensemble techniques that, to the best of our knowledge, have not been previously studied in the PHM domain. The ensemble techniques we have explored are further described in section 4.5.

3 Datasets

This chapter examines the various datasets that exist in the PHM domain, as well as the shortcomings of each.

3.1 PHM2017

The PHM2017 dataset was introduced by Karisani and Agichtein [28] and has since been widely used in the PHM domain. Every dataset before this was severely limited in some way or another, either because they were based on medicine names instead of diseases [23], concentrated on just one disease [37] or had a limited sample size [29]. The PHM2017 dataset consisted of 7,192 samples across six diseases, which was significant compared to the existing datasets. Having samples from multiple diseases pushed future architectures to learn to identify personal health mentions in general instead of learning to identify a specific disease.

The dataset was created by scraping English tweets from Twitter using keyword searches. Colloquial disease names were used to search for the data, with the keywords covering six diseases and conditions in total. Re-tweets and replies were removed from the data, and the remaining tweets were manually annotated into one of four classes: self-mention, other-mention, awareness and non-health. The self-mention and other-mention classes consisted of tweets related to the health of a specific person, either the author themselves in the case of self-mention or someone else in the case of other-mention. The awareness class was for tweets discussing a disease but not any specific person suffering from the disease, while the non-health class was for tweets that were entirely unrelated to health.

3.2 HMC2019

Despite its widespread use, the PHM2017 dataset had one major issue. Posts collected from social media texts frequently involved sarcasm or figurative language. The dataset did not explicitly take these use cases into account, and as a result, models that trained on the dataset, struggled with figurative language [24]. To address this issue, Biddle et al. [5] expanded the PHM2017 dataset and introduced a new dataset, the HMC2019 dataset. This dataset has a separate class for figurative health mentions, which allows models that are trained using this dataset to learn to identify figurative language.

Twitter policy[3] does not allow tweets to be stored as plain text in public datasets, so the PHM2017 dataset only contained the tweet IDs. The authors of the HMC2019 dataset had to re-scrape the data using the IDs, which resulted in the loss of a large number of samples due to deleted tweets. At the time of download, only 5,497 samples could be collected. The six disease keywords from the PHM2017 dataset, along with four new ones, were then used to collect an additional 14,061 tweets.

The authors of the HMC2019 dataset combined the self-mention and other-mention classes from the PHM2017 dataset into a single class, the positive class. Similarly, they combined the awareness and non-health classes into a single negative class. Additionally, they introduced a new class label, figurative, with the aim of creating a dataset that focused on figurative use cases of disease keywords. All of the samples were manually annotated to divide them into one of these three classes. A lot of work that has made use of the HMC2019 dataset has further combined the figurative and negative classes into a single negative class, thus creating a binary classification dataset. We have also included this variant in our experiments for

[3]https://developer.twitter.com/en/developer-terms/policy

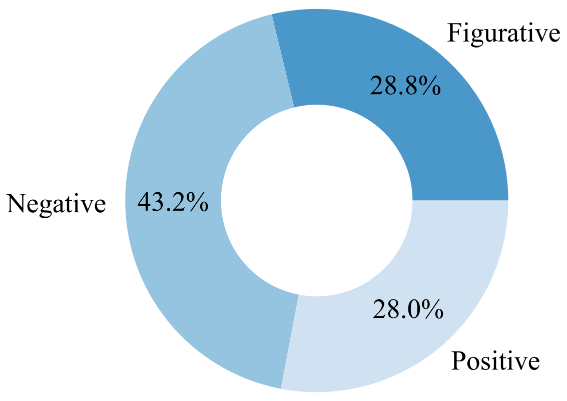


Figure 1: Class Distribution of HMC2019 Dataset

comparison with past work. Throughout the rest of this thesis, the binary variant of the HMC2019 dataset is referred to as the HMC2 dataset, whereas the original variant is referred to as the HMC3 dataset. When referring to the dataset in general, regardless of the variant, it is addressed as the HMC2019 dataset.

Since the HMC2019 dataset also uses data from Twitter, it suffers from the same issue that the PHM2017 dataset did in that the texts from the tweets are not publicly available. As such, the dataset had to be re-scraped. At the time of download, 19,475 of the samples remained. Other authors who have worked with this dataset have reported varying values depending on the time at which they downloaded the data. The class distribution for the dataset as we collected it is provided in Fig. 1.

3.3 RHMD

The RHMD dataset was created by Naseem et al. [48] with the intention of contributing comparatively longer samples of text to the PHM domain. To this end, the dataset was created by collecting posts from Reddit using disease and symptom keyword searches. In total, the dataset consists of 10,015 posts across fifteen diseases. An additional benefit of the posts being collected from Reddit is that the text can be made publicly available, so none of the samples were lost. The RHMD

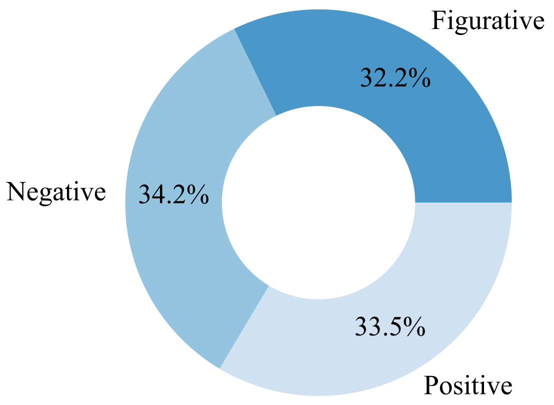


Figure 2: Class Distribution of RHMD Dataset

dataset originally consisted of four class labels: positive, negative, figurative, and hyperbolic. The publicly available version of the dataset released by the authors, however, combines the figurative and hyperbolic classes. Fig. 2 shows the class distributions for the RHMD dataset.

Table 1 shows samples for each of the classes in both datasets in order to provide a better understanding of what constitutes a positive, negative, or figurative health mention.

4 Proposed Methodology

The proposed architecture is primarily divided into two sections: a pre-trained model that serves as a feature extractor, and a classifier network. In addition to this, we have also incorporated four separate ensemble architectures. The proposed pipeline is shown in Fig. 3. Each of the components is discussed in this section.

4.1 Pre-Processing

The data being used in our experiments was obtained from the social media platforms Twitter and Reddit, as described in section 3. Data from these sources tends to be particularly noisy since users frequently use shorthand expressions, emojis, incorrect spellings, etc. Such data will cause the tokenization process to generate

Dataset	**Label**	**Sample Text**
HMC2019	Figurative	My alarm just gave me a heart attack
	Negative	Can poor sleep lead to Alzheimer's? [URL]
	Positive	I've had this headache for more than 6 hours now wow
RHMD	Figurative	Almost killed a squirrel today I was biking back to my room and a squirrel literally fell right in front of my bike, 2 seconds away from being roadkill. Luckily I swerved to the right just in time. Almost had a heart attack.
	Negative	Australia pulls Nurofen products over 'misleading claims' - The court says products marketed to treat specific pains, such as migraine, are identical to one another.
	Positive	Woke up with a terrible migraine while visiting my Mother-in-law. Without opening my eyes, I popped the 2 Tylenol she handed me. Still had a migraine hours later.

Table 1: Sample Documents from Datasets

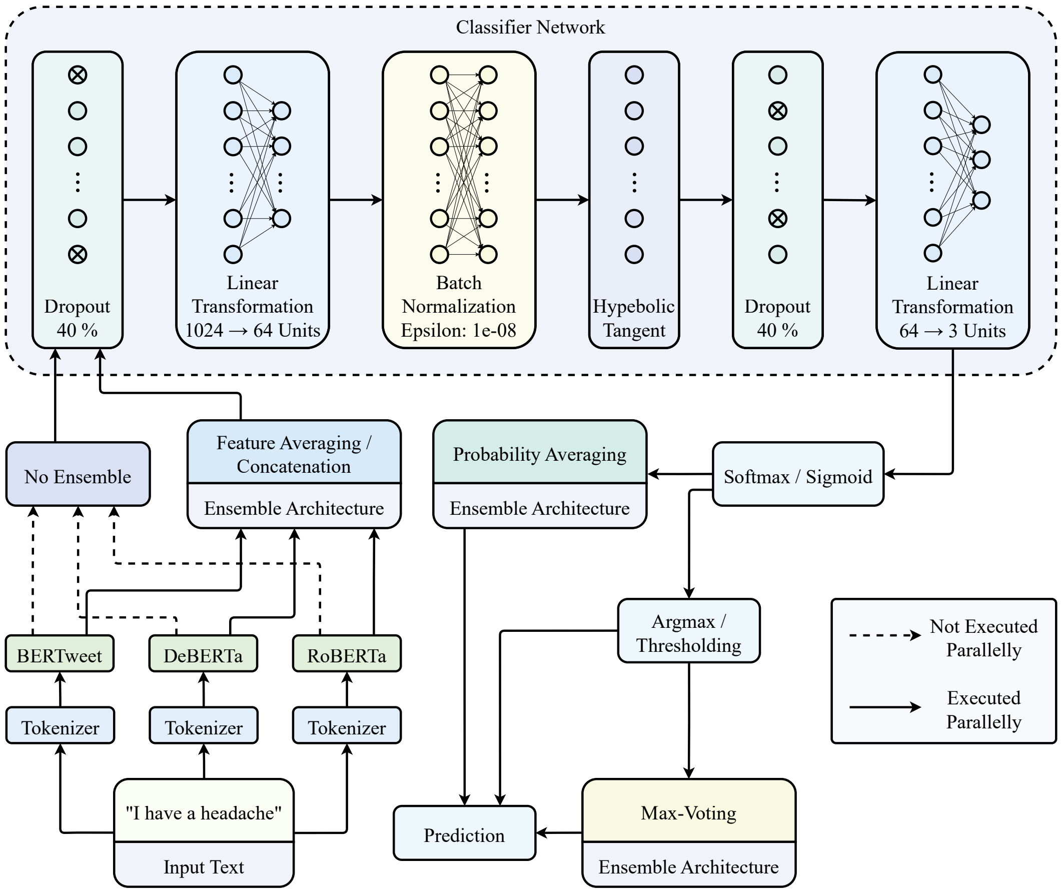


Figure 3: Proposed Pipeline

a large number of [UNK] tokens, which will hamper performance. Further details about the tokenization process are provided in section 4.3, but the key takeaway is that the data should be cleaned and normalised as much as possible before being handed over to the tokenizers. To this end, a variety of data-cleaning techniques were employed. The majority of the work was done in accordance with the work of Baziotis et al. [4]. This provided us with primarily three cleaning mechanisms:

1. URLs, email addresses, phone numbers, etc. appear commonly in social media posts, but they do not provide our models with any useful information for classifying the text. As such, they were replaced with indicative tags, such as [URL] for URLs, [EMAIL] for email addresses, and so on. This removes the noise caused by them while preserving the structural information they provide.

2. Hashtags usually combine multiple words without any spaces, e.g., #heartattack. Such combinations will cause issues in the tokenization process, which relies on words being separated by spaces to be able to identify them. Thus, splitting the hashtags into separate words should help. The splitting was performed using a one-to-one mapping between the hashtags and the split words. This map was based on a corpus of text collected from Twitter.

3. Common misspellings were corrected using a dictionary that was created using Twitter posts as well as with the help of a custom dictionary.

In addition to the above, we also removed punctuation marks and expanded contractions.

4.2 Feature Extractors

As feature extractors, we have opted to use three pre-trained models, namely BERTweet [51], DeBERTa [17] and RoBERTa [40]. All three of these models are based on the transformer architecture [63], which is a sequence-to-sequence architecture. This architecture takes an input sentence, extracts features from it using an encoder block, and uses those features to make output predictions using a decoder block. The pre-trained models we have used have several variants, each with a different size and parameter count. Generally, the performance of the models improves as the number of parameters is increased, with the drawback that the size also increases. Based on the computational resources available to us, the large variants of each of these models were found to provide an acceptable balance between performance and computational requirements.

All transformer-based architectures fall into one of two categories: auto-regressive or auto-encoding. The three feature extractors we have used follow the auto-encoding architecture. Auto-encoding models are trained by masking tokens from the input sentences using a special [MASK] token and training the model to predict the masked words. This is called the Masked Language Model (MLM) objective, and it allows the model to learn the underlying contextual dependencies of the language. By contrast, auto-regressive models are trained with the objective of predicting the next word given an input sequence. Unfortunately, because of the way they are trained, auto-regressive models are unable to take advantage of bi-directional context, i.e., they can only use the context of the words before them, not those that come after. Auto-encoding models are able to do this since the masked words they need to predict are usually in the middle of the sequence, not at the end.

The first auto-encoding model we chose to explore was RoBERTa, due to the simple yet impressive improvements the authors made to the original BERT architecture. They showed that the hyperparameters and design choices made during the pre-training of BERT were limiting the architecture from performing at its full potential. In this respect, they improved performance by making the following changes:

1. Instead of using a fixed mask for the MLM objective like BERT, RoBERTa dynamically generates the masking pattern during the pre-training process.

2. The Next Sequence Prediction (NSP) objective, which involves predicting whether two segments of text appear one after another, was removed.

3. RoBERTa used an alternative implementation of Byte-Pair encoding [61], originally introduced by Radford et al. [59]. This uses bytes instead of Unicode characters, which significantly reduces the vocabulary size while also being able to encode any input text. The implementation degrades performance but provides a universal encoding scheme.

4. RoBERTa was trained on larger batch sizes, for a larger number of pre-training steps, and on significantly more data. BERT used 13GB of data from a single dataset [69], while RoBERTa used three additional datasets [15, 46, 62] which totalled 160 GB of data.

BERTweet also used the BERT architecture and followed the same pre-training procedure as RoBERTa. The change the authors made was related to the data on which the model was pre-trained. The authors used a combination of two datasets, the first being the general Twitter Stream collected by the Archive Team[4] and the second being a dataset of tweets related to COVID-19 that they collected themselves. For the general Twitter Stream dataset, the English tweets were extracted using fasttext [26]. In total, these two datasets represented 80 GB of data consisting of 850 million English tweets. The authors also followed a specific data-cleaning process. They tokenized the tweets using TweetTokenizer [6], converted emojis to their textual representations, replaced user mentions and hyperlinks with indicative tags, and removed retweets and tweets shorter than 10 tokens or longer than 64 tokens.

In contrast to BERTweet and RoBERTa, DeBERTa modifies the BERT architecture. BERT takes the word embeddings and position embeddings of each token and creates a single vector by taking the sum of the two. The authors of DeBERTa argue that the combination of the two vectors harms the performance of BERT since the information about individual words and their positions gets mixed together. Instead, they suggest keeping the two vectors separate. This, however, requires changing the way in which the attention scores are calculated, for which they propose a new mechanism, the disentangled attention mechanism.

[4]https://archive.org/details/twitterstream

Additionally, the position embeddings are relative, unlike BERT, which uses absolute ones. In many cases, however, the absolute position embeddings becomes important. To this end, the absolute position embeddings are also added to the model, just before the model reaches the softmax layer. This modified decoder section is referred to as the enhanced mask decoder by the authors.

He et al. [18] later improved the DeBERTa model further by replacing the MLM objective with a Replaced Token Detection (RTD) objective, which was proposed by ELECTRA [10]. Under this approach, a generator is used to generate ambiguous corruptions in the tokens, and a discriminator is used to identify the corruptions. The original implementation used the same token embeddings for the generator and discriminator. However, the generator exists to bring tokens that are similar closer together, while the discriminator exists to pull such tokens apart. Thus, using the same token embeddings harms performance on downstream tasks. To deal with this, the authors of DeBERTa proposed a new gradient-disentangled embedding sharing (GDES) method. Here, the embeddings are still shared, but the gradients from the discriminator are not backpropagated to the generator, which avoids the issue of the tokens being pulled in opposite directions.

4.3 Tokenization

Transformer models take input in the form of tokens, which are integer representations for each word or word fragment. Each of the models we have worked with has its own tokenizer, which performs the conversion of words to tokens. There are some differences in the tokenization mechanisms of each tokenizer, which are briefly discussed in section 4.2, but the general idea for all of the tokenizers is that words that are closely related to each other tend to have similar token values. These values and their relationships with each other is what allows transformer-based models to understand the context.

In addition to the tokens used to represent words, the tokenization process also adds a few special tokens to the token array. For example, the [SEP] token is added at points where one sentence ends and another one begins, and the [UNK] token is used for any words or word fragments that the tokenizer did not have in its dictionary and could not tokenize. The use of [UNK] tokens is what makes the data cleaning process described in section 4.1 particularly important. Data which contains a lot of misspelt or abbreviated words will cause the tokenizers to generate [UNK] tokens for those words, even though they would recognize the same words had their spellings been correct. This in turn will adversely affect the performance of the model since the misspelt words will essentially not be part of the input. This makes it essential that such misspellings and abbreviations be avoided as much as possible. The BERTweet model, however, should be comparatively more robust to such issues since its training process used the TweetTokenizer to tokenize its training text corpus. The TweetTokenizer was created specifically to tokenize text from social media. Whereas a general tokenizer would fail to recognise things like hashtags and emojis and thus split them into individual punctuations (e.g., splitting ':)' into ':' and ')'), the TweetTokenizer is able to keep these entities intact and produce tokens for them.

Another special token that is usually used is the [PAD] token, which is appended to the end of shorter token sequences since transformer models expect every token sequence to be the same length. In our case, we used dynamic padding with uniform length batches to bypass this requirement and improve performance. This mechanism is further explored in section 6.1. However, a maximum token length was still required, which was set to 512. This value was chosen by analysing the token lengths of the two datasets. Fig. 4 and 5 show a histogram of the token lengths of the HMC2019 and RHMD datasets respectively. From this, it can be seen that the majority of the samples have token lengths that are significantly smaller than 512. In fact, this limit is not crossed by any of the samples from the HMC2019 dataset and is crossed by only one of the samples from the RHMD dataset. As such, it was determined that the use of even larger models capable of supporting larger token lengths would not have been a fruitful undertaking, given the additional computational overhead they would add.

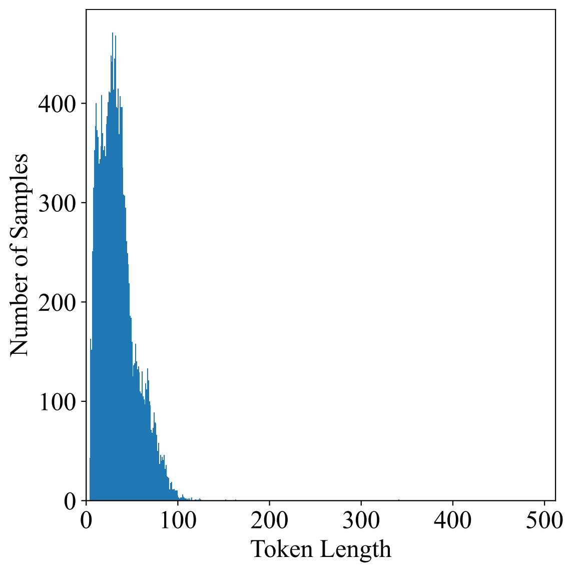


Figure 4: Frequency Distribution of Samples per Token Length for the HMC2019 Dataset

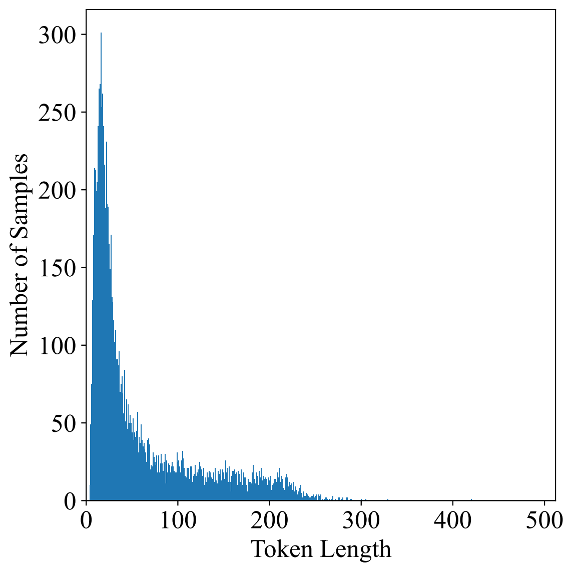


Figure 5: Frequency Distribution of Samples per Token Length for the RHMD Dataset

4.4 Classifier Network

The outputs of the pre-trained models represent features of the text. In order to determine the correct class for the text using those features, we need a classifier network. The classifier network we have designed is shown in Fig. 3. Unlike the pre-trained models, the classifier network had its weights initialised randomly and was trained from scratch. Additionally, the weights of the pre-trained models were also fine-tuned during the training process. These models were pre-trained on large corpora of text, which enables them to grasp linguistic features and makes them good feature extractors. The features they learnt, however, are generic language features since they were trained on general text. Fine-tuning them to the specific downstream task at hand generally improves performance [27].

The rest of this subsection discusses the four types of layers that have been used in our classifier network: dropout layers, linear transformation layers, batch normalisation layers [20] and hyperbolic tangent activation layers. The linear transformation layers exist to map the higher-dimensional vectors to lower-dimensional ones. This occurs in two steps. Firstly, to reduce the 1024-unit output from the pre-trained models to a 64-unit vector, and later, to reduce the 64-unit vector to either a single value or a 3-unit vector depending on the number of classes in the dataset. Between the linear transformation layers, we have a batch normalisation and a dropout layer. Batch normalisation is used to rescale the values across different features, thus reducing the amount of time required for gradient descent to converge. The dropout layer randomly sets a portion of the output units from the previous layer to 0. The percentage of output units set to 0 is called the dropout rate, and this was set to 40%. By doing this, the dropout layer behaves as an L2 regulariser [64], preventing the model from over-fitting to the training data. An equivalent dropout layer was also used before the first linear layer for the same purpose.

We have used the hyperbolic tangent function as an activation function in the classifier network. This function bounds the outputs from the previous layer to a range of -1 to $+1$, which prevents the gradient values from exploding and becoming very large over time. Traditionally, the sigmoid activation function was used, which bounds the values to a range of 0 to +1. The hyperbolic tangent function is in fact the same as the sigmoid function, except that it uses a different range. However, by being symmetric around the zero point, the hyperbolic tangent function is able to provide faster convergence [58].

If the final output from the second linear transformation layer is a 3-unit vector, this vector is put through a logarithmic softmax layer. The softmax function maps the 3 values to the range of 0 to 1 in such a manner that they sum up to 1. Thus, each value represents the probability that the correct output is of the corresponding class. The logarithm function is applied on top of this for numerical stability. The class with the highest output value is chosen as the correct one. On the other hand, if the final output is a single value, a sigmoid layer is used. Afterwards, we map the output value to the negative class if it is below 0.5 and to the positive class if it is above.

4.5 Ensemble Techniques

Four ensemble techniques were explored: max-voting, probability averaging [60], feature averaging, and concatenation.

Max-voting and probability averaging work with the fine-tuned models. For max-voting, we examine the results of all three models on the same test set. In cases where the three models disagree on the correct output, the output chosen by the majority of the models is assumed to be the correct one. Taking into account the opinions of multiple models in this manner should lead to better results. Theoretically, there is a possibility of some conflict here in cases where there are 3 possible outcomes, as in the RHMD and HMC3 datasets, but our results show that the technique improved performance, which indicates that such conflicts did not arise. On the other hand, probability averaging uses the probability values assigned to the different possible outcomes by each model. For binary classification, the probability values are mapped to 0 or 1 using a threshold of 0.5. For multi-class classification, the class with the highest probability value is assigned the value of 1, while the other classes are assigned the value of 0. Probability averaging involves taking the average of the probability values given by each of the three models before performing the final classification. In cases where the majority of the models make an incorrect classification with low confidence but a minority make the correct classification with high confidence, max-voting will fail since it relies purely on the number of models voting for a particular classification, not the confidence of the models. Probability averaging resolves this issue.

Two variants of probability averaging were explored. The first variant used the arithmetic mean of the probability values, which is calculated as:

$$\text{Arithmetic Mean } (\mathcal{A}) = \frac{1}{n}\sum_{i=1}^{n} p_i \tag{1}$$

Here, n is the total number of pre-trained models, and p_i is the probability value obtained from the ith model. The second variant used the harmonic mean of the probability values, given by:

$$\text{Harmonic Mean } (\mathcal{H}) = \frac{n}{\sum_{i=1}^{n} \frac{1}{p_i}} \tag{2}$$

The two variants provide slightly different outcomes. The arithmetic mean gives a result that is centralised between all of the available outcomes, while the harmonic mean gives a result that leans towards the majority of the values. This can be useful in cases where outliers exist. In such cases, the arithmetic mean will become skewed towards the outlier, resulting in the average value misrepresenting the group. The harmonic mean is more resistant to such outliers, making the average value a better representative of the group.

In contrast to the techniques discussed above, feature averaging and concatenation are both applied during the training process of the model. For feature averaging, the initial input is provided to each of the three pre-trained models, which each give us a 1024-unit feature vector as the output. The average of these three vectors is then provided to the classifier network. The theoretical reasoning behind why this should work is similar to that of probability averaging. For a particular feature, an individual model may give an incorrect value as the output. Taking the average value from multiple models should resolve such issues. Concatenation, on the other hand, involves concatenating each of the 1024-unit feature vectors from the three pre-trained models to create a 3072-unit vector. This larger vector is then provided to the classifier network. Concatenation attempts to address a potential shortcoming in the logic behind feature averaging. The different pre-trained models may not give us the same features as their outputs. As such, if we want to maximise the number of features we obtain, we should take all of the outputs into account simultaneously. Both of these techniques are extremely computationally expensive since they involve fine-tuning three separate pre-trained models simultaneously. As such, additional efforts had to be made to reduce the computational expense. The mechanisms used to accomplish this are further discussed in section 6.

4.6 Experimental Setup and Hyper-parameters

The experiments in this study were conducted using hardware provided by Google Cloud[5], PyTorch [54] and CUDA Version 11.6. We used an Intel Xeon Platinum 8273CL CPU with 85 GB of RAM and an Nvidia A100 GPU with 40 GB of video RAM.

Before the experiments were started, the datasets were first split into training, validation, and test sets using the split ratio 60 : 20 : 20. During the training process, the models were given data in batches of size 16 from the training set and trained for 5 epochs. It was found that training for more epochs did not result in improved performance. The models were evaluated on the validation set after every epoch, and the weights that achieved the best performance on the validation set were saved for the test phase. The test set remained unseen throughout the training process.

During the training stage, the gradient values were updated after each batch of data was processed. Each of these updates is called a step. The first 20% of the steps in each epoch during the training stage were used as warm-up steps.

[5]https://cloud.google.com/

The learning rate was gradually increased from 0 to $1e-05$ during these steps. Warm-up steps are used to reduce the possibility of the model becoming biased during the early stages of training, as it is still being introduced to new data [16]. The weights for the model are adjusted using the cross-entropy loss function. To update the model weights, the AdamW optimizer [41] was used with a weight decay of 0.03 and an epsilon value of $1e-08$, the default value. Weight decay acts as a regularisation mechanism, while the epsilon value is used purely for numerical stability and does not affect the performance or results of our experiments. The optimization process took place after each step of batch training.

5 Result Analysis

A thorough analysis of the results obtained using each of the pre-trained models and ensemble techniques described in Chapter 4 is presented in this section.

5.1 Evaluation Metrics

The metrics that should be used to evaluate model performance vary from one dataset to another. Every metric is not necessarily sensible for every dataset, and using the wrong metric can lead to misleading results. Because of this, we have included several metrics, starting with fairly standard ones and using more advanced metrics where the standard ones showed signs of being erroneous.

Accuracy, precision, and recall are all widely used evaluation metrics and have been included in our results. Unfortunately, the datasets we have used, especially the HMC2 and HMC3 datasets, are imbalanced. For imbalanced datasets, these metrics are known to produce misleading results [39]. It is possible to reach a high accuracy while not truly performing well if the model is biased towards a class that has a significantly larger number of samples. For example, if one of the classes contains 90% of the samples, the model will be able to achieve 90% accuracy by simply always predicting that class. This, however, makes the model useless practically. Similar issues occur with precision and recall since neither metric provides any penalty for incorrect results. Precision only accounts for true positives and false positives, while recall only accounts for true positives and false negatives. As such, it is possible to get a high precision by making a single correct positive classification and a high recall by constantly making positive classifications.

The issues outlined above are addressed by the use of two other metrics: F1-Score and Area Under the Receiver Operating Characteristic Curve (AUC-ROC). F1-Score is the harmonic mean of precision and recall. By depending on both precision and recall for its final outcome, F1-Score is able to remove the issues seen with the two metrics individually. It takes into account the total number of true positives, false negatives, and false positives for each class and calculates the weighted average of the results. By taking the weighted average, F1-Score takes class imbalance into account.

The Receiver Operating Characteristic (ROC) curve evaluates the False Positive Rate (FPR) and the True Positive Rate (TPR) for a series of predictions, which allows the curve to create a summary of the model's ability to differentiate between classes. The ROC curve sees classification as a binary problem for each class, i.e., either a sample belongs to this class or it does not. It calculates the FPR and TPR values based on this definition. By default, binary classification uses a threshold of 0.5 to divide the predicted probability values into either the positive or negative class. The ROC curve varies this threshold value, which in turn affects the FPR and TPR values. AUC-ROC provides a single value that represents the results of the ROC curve. Since the ROC curve varies the threshold, a high area under the curve indicates that the model is able to correctly separate one class from the rest for a large number of thresholds.

5.2 Performance Comparison

Table 2 compares the results of our experiments with those of previous research on the RHMD, HMC3, and HMC2 datasets. The best results across each of the evaluation metrics are highlighted in bold. Note that the results of the PHS-BERT architecture for the RHMD dataset are for the 4-class variant of the dataset, which is not publicly available.

It is challenging to provide a definitive comparison between our results and those of previous authors, primarily due to variations in the quantity of data available, as outlined in section 3. Our results would most likely change slightly if we had access to the exact same data as the previous authors. However, the comparison shown here indicates that our models have outperformed existing work on several fronts.

PHS-BERT was initialised with the weights of BERT and further pre-trained on social media texts before being fine-tuned to downstream tasks. Despite the extra pre-training, PHS-BERT was outperformed by our models on both the RHMD and HMC3 datasets. This calls into question the rationality of choosing to initialise PHS-BERT with the weights of the BERT architecture. The authors of RoBERTa have thoroughly proven that BERT was not sufficiently pre-trained, so a different architecture would most likely have provided a better starting point for PHS-BERT. A similar issue can be seen for the CT-BERT + Bi-LSTM architecture. Our results show that the pre-trained models we have used outperform the CT-BERT + Bi-LSTM architecture by a significant margin on the RHMD dataset. This indicates that the architecture could have had improved performance on this dataset had it been initialised differently. The architecture is initialised with the weights of the CT-BERT model, which was trained on data from Twitter. This explains its poor performance on the RHMD dataset, which consists of longer text from Reddit.

The BiLSTM + Senti model, as the name suggests, concatenates sentiment information with the output from a BiLSTM classifier. The use of transformer-based models allows us to outperform this architecture by a significant margin on both the RHMD and HMC2 datasets. The authors did show, however, that the use of sentiment information in the BiLSTM + Senti model improved performance over the Bi-LSTM classifier. This indicates an important avenue for future exploration in the PHM domain, since incorporating the sentiment information

Table 2: Experimental Results

Dataset	Paper	Architecture	Accuracy	Precision	Recall	F1 Score	AUC-ROC
RHMD	Naseem et al. [47]	BiLSTM + Senti	-	71.00	71.00	71.00	-
	Naseem et al. [50]	PHS-BERT	-	-	-	77.16	-
	Naseem et al. [49]	CT-BERT + Bi-LSTM	-	-	-	79.00	-
	Naseem et al. [48]	HMCNET	-	81.00	81.00	81.00	-
	Khan et al. [35]	RoBERTa + AT + Ctr	-	**83.43**	83.27	83.23	-
	Ours	BERTweet + Classifier	82.63	82.65	82.63	82.64	91.73
		DeBERTa + Classifier	82.53	82.61	82.53	82.56	91.94
		RoBERTa + Classifier	82.78	82.74	82.78	82.71	92.06
		Feature Averaging	82.63	82.64	82.63	82.58	90.89
		Concatenation	82.58	82.52	82.58	82.53	**92.57**
		Max Voting	83.37	83.35	83.37	83.36	-
		Probability Averaging	**83.42**	83.41	**83.42**	**83.42**	91.63
HMC3	Naseem et al. [50]	PHS-BERT	-	-	-	91.71	-

Continued on next page

Table 2: Experimental Results (Continued)

Dataset	**Paper**	**Architecture**	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**AUC-ROC**
	Naseem et al. [49]	CT-BERT + Bi-LSTM	-	92.40	92.10	92.50	-
	Ours	BERTweet + Classifier	91.96	92.02	91.96	91.97	96.26
		DeBERTa + Classifier	91.94	92.04	91.94	91.95	97.05
		RoBERTa + Classifier	91.81	91.86	91.81	91.82	96.23
		Feature Averaging	91.81	91.86	91.81	91.82	97.15
		Concatenation	91.89	91.97	91.89	91.90	96.91
		Max Voting	**92.61**	**92.68**	**92.61**	**92.62**	-
		Probability Averaging	**92.61**	92.67	**92.61**	**92.62**	**97.40**
HMC2	Biddle et al. [5]	BiLSTM + Senti	-	75.60	92.00	82.90	-
	Khan et al. [32]	XLNet	-	89.10	88.20	88.40	-
	Khan et al. [34]	RoBERTa	-	93.00	93.00	93.00	-
	Khan et al. [35]	RoBERTa + AT + Ctr	-	94.25	94.35	94.30	-
	Khan et al. [33]	RoBERTa + AT + BT	-	94.35	94.40	94.45	-

Continued on next page

Table 2: Experimental Results (Continued)

Dataset	**Paper**	**Architecture**	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**AUC-ROC**
	Ours	BERTweet + Classifier	95.34	95.38	95.34	95.35	97.93
		DeBERTa + Classifier	95.34	95.40	95.34	95.36	97.41
		RoBERTa + Classifier	94.82	94.80	94.82	94.81	97.56
		Feature Averaging	95.34	95.42	95.34	95.36	**98.06**
		Concatenation	**95.67**	**95.70**	**95.67**	**95.68**	97.95
		Max Voting	**95.67**	95.69	**95.67**	**95.68**	-
		Probability Averaging	95.65	95.66	95.65	95.65	97.49

into our proposed models could lead to further improvement in performance.

Another impressive achievement of our work is that the results obtained by the pre-trained models we used were better than those obtained by the XLNet and RoBERTa-based models used by Khan et al. [32] and Khan et al. [34] respectively. Specifically, the improved performance of the RoBERTa-based model we have used compared to the RoBERTa-based model used by Khan et al. [34] is of significant importance, since the only difference between the two is the classifier section. This is both an indication that our classifier is better and proof of the importance of concentrating on not only choosing the correct feature extractor but also designing a strong classifier network.

No differences could be found between the RoBERTa + AT + Ctr and RoBERTa + AT + BT models, so they are discussed here together. The use of adversarial training and contrastive loss causes the performance of the base RoBERTa model to improve significantly. These models had far better performance on both the HMC2 and RHMD datasets compared to other existing work. Although these models were outperformed by the pre-trained models we used on the HMC2 dataset, the same cannot be said for the RHMD dataset, where only the max voting and probability averaging ensemble techniques were able to achieve better performance. This strongly indicates that the use of adversarial training and contrastive loss is an important direction for further research, especially when working with longer texts such as those in the RHMD dataset. Another possibility for further research lies with the use of user behavioural features, as was done in the HMCNET model. However, the results of this model are comparatively worse, based on which we can conclude that further exploration related to these features should be given a lower priority in the PHM domain.

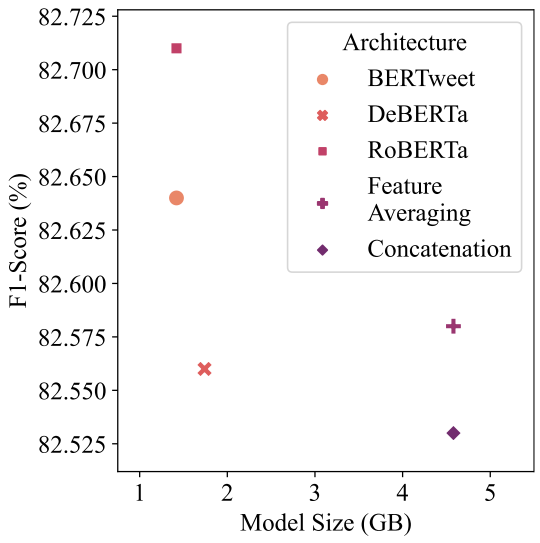
5.3 Performance Analysis

The main takeaway from the results focuses on the ensemble techniques. Both feature averaging and concatenation achieved results that are comparable to those of the pre-trained models. There is only one case, with the HMC2 dataset, in which concatenation led to better results than any of the pre-trained models. These results become even more unfavourable for these two ensemble techniques when we consider the fact that they have significantly larger model sizes and higher computational expenses compared to the pre-trained models. Fig. 6 illustrates this by comparing the model size, number of floating-point operations (FLOPs), and number of parameters for each of the architectures trained on the RHMD dataset.

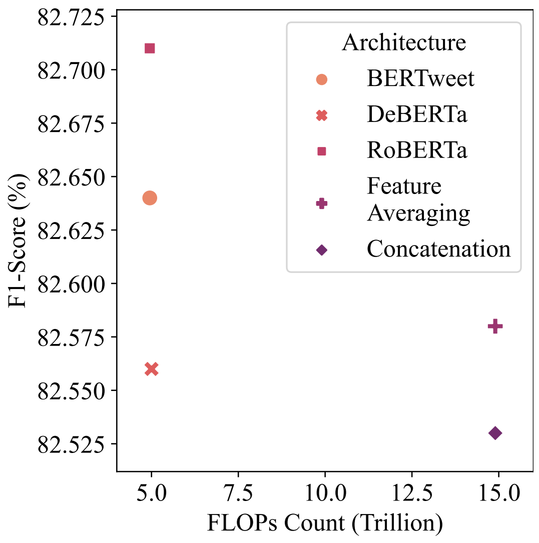
Max voting and probability averaging, by comparison, have slightly better results when considering the overall performance on all three of the datasets. This is understandable since both techniques improve upon the results already obtained by the pre-trained models. Based on this, it seems that max-voting and probability averaging are superior ensemble techniques, especially considering that they have significantly lower computational requirements. Comparing the max-voting and probability averaging ensemble techniques to the pre-trained models, our results agree with the work of Ahmed et al. [2], in that both techniques result in improved performance. Between the two techniques, it is difficult to determine conclusively which is better due to the proximity of their results.

For probability averaging, the results for the arithmetic mean variant are shown. It was found that the arithmetic mean gave slightly better results than the harmonic mean, with a difference of 0.2 in their F1-Scores on average. These results suggest that there are cases where a single model is giving the correct results while the other two are not. This discrepancy is the most likely cause behind the arithmetic mean, which would become skewed towards the single model in such cases, being able to perform slightly better than the harmonic mean, which would not.

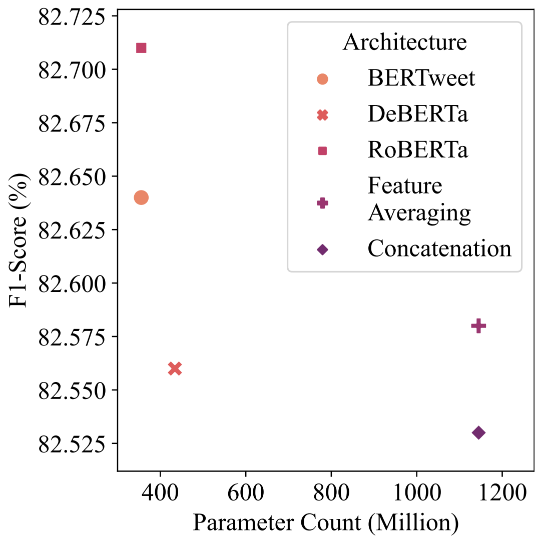
Fig. 7 shows the ROC curves of the probability averaging ensemble for the HMC2, HMC3, and RHMD datasets. The performance for the RHMD dataset is the worst, which aligns with the results shown in Table 2. These results can be attributed to the larger lengths of the samples, which can be seen in Fig. 5, combined with the smaller dataset size. Longer sentences present a bigger challenge



(a) Model Size

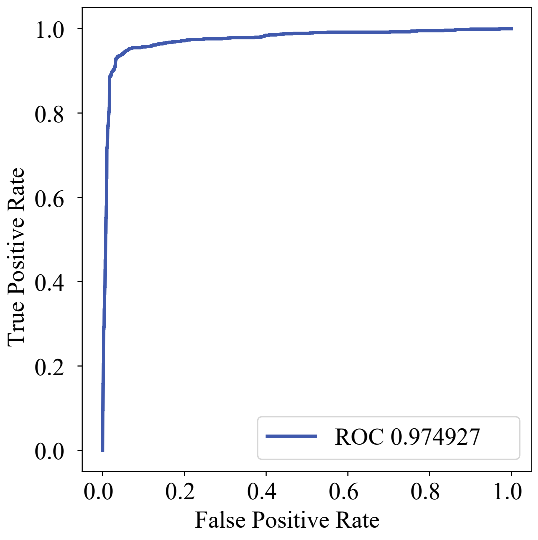


(b) FLOPs Count

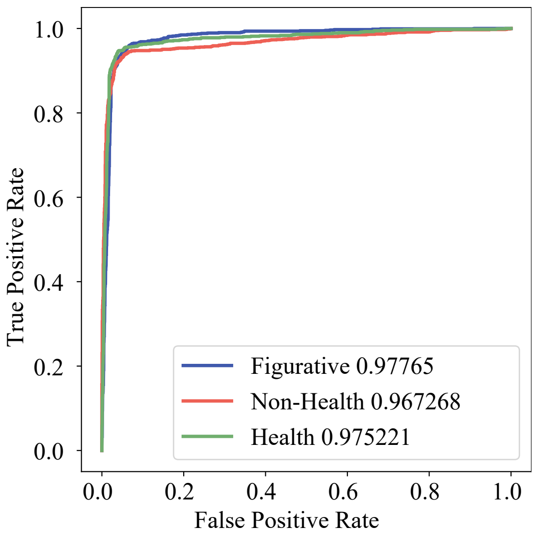


(c) Parameters Count

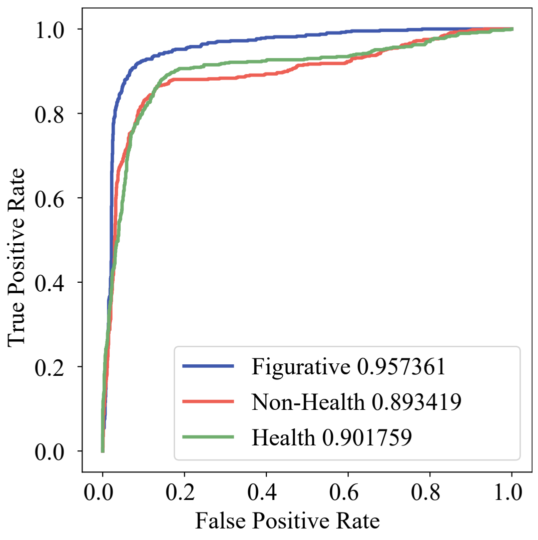
Figure 6: Comparison of Architecture Metrics



(a) HMC2



(b) HMC3



(c) RHMD

Figure 7: ROC Curves for Probability Averaging Ensemble

to the models since it becomes more difficult to retain the context information from sections of text that are very far apart. On top of this, the smaller dataset size gives the model less opportunity to learn the hidden patterns in the samples required to classify them correctly. Additionally, it seems the model is able to identify figurative mentions better than negative or positive ones. Section 5.4 presents a more detailed analysis of specific mistakes the model made to understand this behaviour better.

5.4 Error Analysis

In order to examine the possible reasons behind the errors our models made during the testing stage, we analysed the results for the probability averaging ensemble technique. A few of the erroneous predictions are shown in Table 3.

Label	**Target**	**Sample Text**
Negative	Positive	I got a 23 and Me test to see if I'm going to develop Alzheimer's. I forgot the results.
Figurative	Positive	Found out I'm allergic to ceiling mounted dart boards... They always make me throw up
Positive	Negative	Doctor: You have Alzheimer's and cancer. Patient: Thank God it's not Alzheimer's.
Positive	Negative	Did you hear about the hairdresser that had cancer? She dyed
Negative	Figurative	Even the Chinese know that Bing is cancer...
Positive	Figurative	I got baby fever
Negative	Positive	recurring fever
Positive	Negative	depression is winning

Table 3: Samples of Erroneous Predictions

One of the major issues that was found involved the use of humour, with the problem being comparatively more prominent in the RHMD dataset. A large number of the errors were on posts that were jokes, but the use of the first-person point of view led to the model marking such posts as positive health mentions. For example, this issue can be seen in the text ‘I got a 23 and Me test to see if I’m going to develop Alzheimer’s. I forgot the results.’. The problem was further exacerbated by the fact that the authors of the RHMD and HMC2019 datasets did not provide explicit instructions to their annotators regarding how to handle humorous posts. This led to some of these posts being marked as figurative mentions while others were marked as negative mentions, even when there were no discernible differences. Such issues highlight an important shortcoming of the entire PHM domain as it stands. So far, the use of humour in social media posts has not been taken into account in any of the existing works, despite its clear and widespread prevalence. The incorporation of mechanisms to detect such humour and the addition of a separate class specifically for humorous posts in the datasets could be important avenues for future work.

Another major source of errors was incorrect labels in the datasets themselves. This applied to both the humorous posts mentioned above and general ones. For example, ‘Did you hear about the hairdresser that had cancer? She dyed’ is labelled as a positive health mention even though it is not, while ‘Even the Chinese know that Bing is cancer...’ is labelled as a negative mention even though it is a figurative one. Correcting these labels would most likely improve performance, but the need for manual annotation makes the process of finding and correcting all of the incorrect labels quite difficult. As such, it is not being pursued at this time.

Posts that were extremely short also caused incorrect classifications. For example, the text ‘depression is winning’ was given a negative classification by our model whereas the dataset labelled it as positive. It is difficult to judge whether the prediction or the label is wrong in such cases since the length of the text makes the context ambiguous.

6 Resource Optimization

The pre-trained models used in our experiments were all quite large. The BERTweet model has a size of 1.42 GB, the DeBERTa model has a size of 874 MB, and the RoBERTa model has a size of 1.43 GB. On top of this, we have two ensemble techniques, feature averaging and concatenation, that combine the three models together, meaning they have to be trained at the same time. Training such large models is extremely resource-intensive in terms of both memory and time. To reduce the time and memory required to train the models, we have implemented two techniques: dynamic padding and gradient checkpointing. These techniques, along with their effects on the resource requirements of the models, are examined in this section.

6.1 Dynamic Padding

One requirement of transformer models is that every array of tokens provided to it has to be the same length. This is due to the fact that transformer models perform matrix multiplications internally, and matrices need to have rows of the same length. However, sentences encountered in social media comments or posts vary in length, which in turn means that the array of tokens generated for the sentences will also vary in length. To be able to work with transformers, we are then forced to add padding to the ends of shorter sentences in order to make the array of tokens match the length of the largest one. This is as simple as adding zeros to the end of the original array of tokens.

The most naive method of adding padding is to simply take the longest sentence in a dataset and pad every other sentence to match that length. This method adds a lot of overhead. Even though all the extra padding does not provide the model with any extra information, the model is still forced to process it. Using the tokenizer for the BERTweet model, it was found that the RHMD dataset has an average token length of 59.81, while the maximum token length is 512. For the HMC2019 dataset, the average token length is just 34.30, but the maximum token length is 341. In both cases, the maximum length is roughly 10 times the average length, which means that sentences have to be padded to 10 times their original length on average. Note that the maximum token length for the RHMD dataset is 512 because this is the largest token length supported by the pre-trained model. Sentences that are too long for the model are truncated to the maximum supported length. The largest sentence in the dataset is larger than this, but since only one of the samples crosses this limit, the use of larger pre-trained models that can support a larger limit was deemed unnecessary.

The sentences we feed to the transformer-based models are more often than not fed in batches. The ideal scenario would be to give the model the entire dataset at once, but that is not practically possible due to memory limitations. Because of this, fixed-sized groups of data are provided to the model one by one. This fixed size is known as the batch size. For our experiments, the batch size used was 16. Dynamic Padding involves reducing the amount of padding required by utilising the fact that the data is being provided in batches. The transformer models do not actually require that every sentence in the entire dataset be the same length. They only require that the sentences they are being given simultaneously be of the same length. This means that we do not need to pad sentences to the length of the longest sentence in the entire dataset, just to the length of the longest sentence in the batch being processed. This reduces the amount of padding the model must process, which in turn improves its training and inference times.

Along with dynamic padding, we have also used uniform-length batches. This involves sorting the tokens by size before dividing them into batches and adding padding. This results in sentences of similar lengths being grouped together, ensuring that we do not end up in a scenario where a large amount of padding must be added to every sentence in a batch just because there is a single long sentence in that batch. The amount of padding applied is the least it could possibly be in this scenario. When using uniform-length batches, the samples should be sorted based on token length. This presents a unique problem when dealing with two of the ensemble techniques we have used, feature averaging and

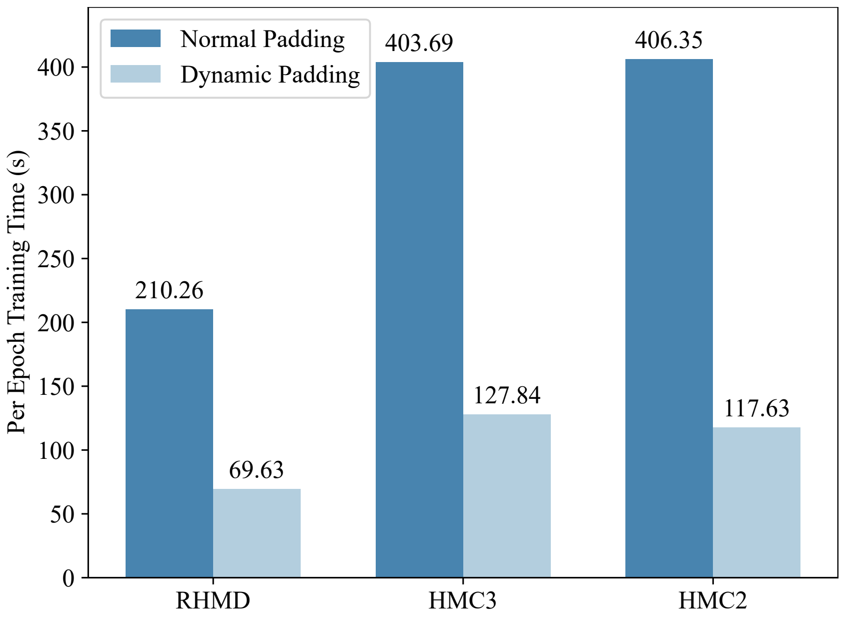


Figure 8: Effect of Dynamic Padding on per Epoch Training Time

concatenation, which both involve combining multiple models. Each of the models uses its own tokenizer and thus has a different set of tokens for every sample. Since it is not guaranteed that the tokens generated by each of the tokenizers for the same sample will be of the same length, the manner in which the samples should be sorted becomes ambiguous. As a compromise, the samples were sorted based on text length. This is sub-optimal since the token lengths are not the same as the text lengths. However, a large improvement in computation time is still achieved. Fig. 8 and Fig. 9 show the training and inference times, respectively, for the BERTweet model on the RHMD, HMC3, and HMC2 datasets with and without dynamic padding with uniform length batches implemented.

In addition to the reduced training and inference times, using dynamic padding also allows us to use the largest size supported by the pre-trained models as the maximum token length. The high computational cost and memory requirements of processing the padded token arrays frequently force researchers to limit the maximum token length to a much smaller value. Previous work on the HMC2019 dataset, for example, used a maximum token size of 128 [33, 35]. Limiting the maximum token length in this manner results in sentences being truncated, which in turn results in the models being unable to utilise the full context provided by

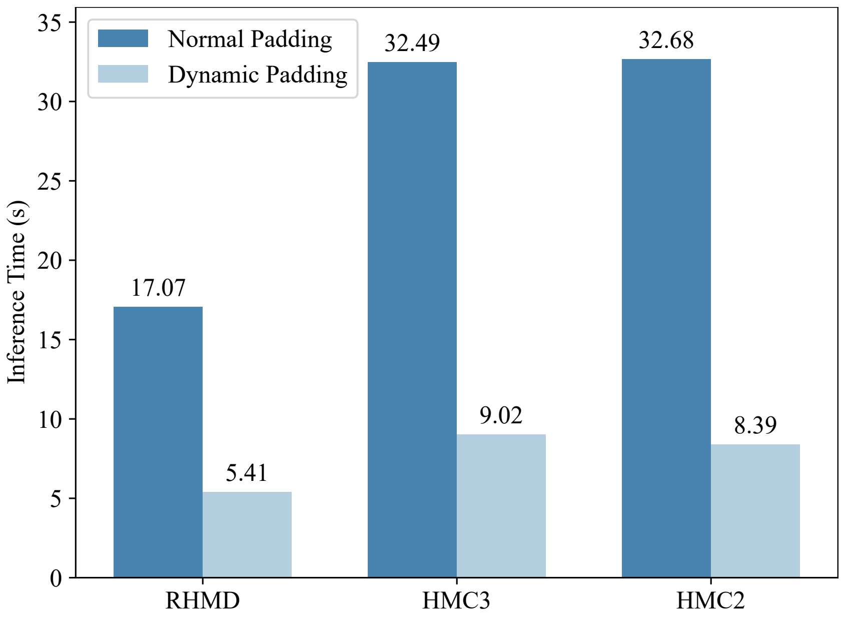


Figure 9: Effect of Dynamic Padding on Inference Time

the sentence. This shortcoming affects the RHMD dataset in particular since the dataset contains a significant number of longer sentences. Since dynamic padding reduces the computational cost, the maximum token length can be set to a larger value, thus preventing most of the sentences from being truncated.

6.2 Gradient Checkpointing

Very large models with a lot of layers require a huge amount of memory to train. Processors powerful enough to work with such models are not usually available to researchers, making it impossible for them to even evaluate the models, let alone train or fine-tune them. Gradient checkpointing [8] is a mechanism that attempts to reduce the amount of memory required by such large models, allowing us to use them on processors with limited memory. For a network with n layers, the memory consumption is reduced from $O(n)$ to $O(\sqrt{n})$ using this mechanism.

During the training process of a neural network, on every epoch, the model goes through a forward pass and then a backward pass. During the forward pass, the model calculates the output for each neuron based on the input to that neuron. The output from the final layer of neurons is compared to the target

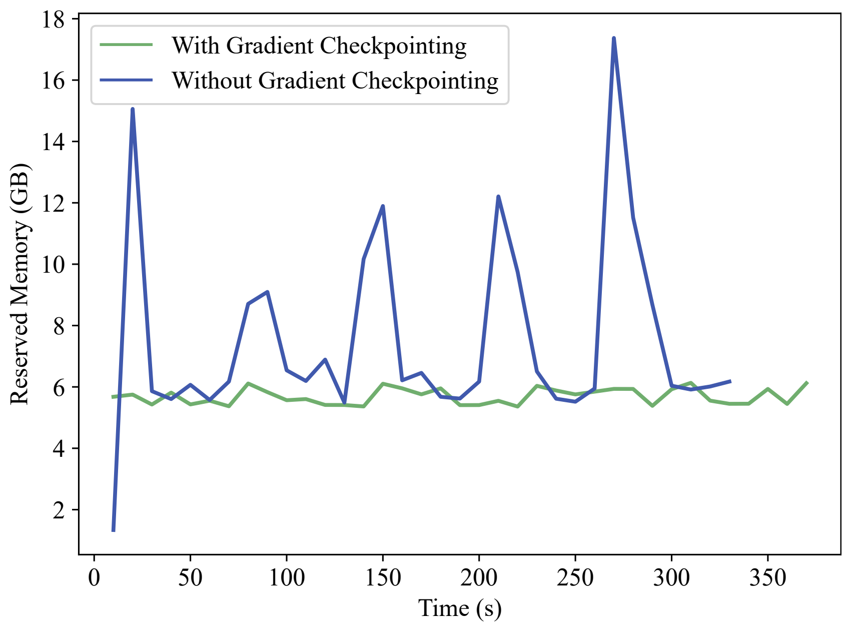


Figure 10: Effect of Gradient Checkpointing on Memory Usage

values using a loss function. The output from the loss function is then used to perform a backward pass through the model, where the model goes over each layer of neurons in the reverse direction, calculating the gradient values. These values are then used to update the weights of the neurons.

The most memory-intensive part of the whole process is the gradient calculation. The model is storing the output values of every neuron during the forward pass and then using those values again during the backward pass to calculate the gradient. This uses up a huge amount of memory for models with a large number of layers. Instead of storing the outputs during the forward pass, gradient checkpointing suggests that we discard them and re-calculate the values during the backward pass. This reduces the memory required to store the outputs during the forward pass. Fig. 10 shows the memory usage during the training of the BERTweet model for the RHMD dataset. Measurements were taken at 10-second intervals.

As can be seen in the figure, there are large spikes in memory when gradient checkpointing is disabled. These spikes would have caused memory issues on hardware with more limited resources. The use of gradient checkpointing removes these spikes, thus reducing the maximum memory usage. There are five significant

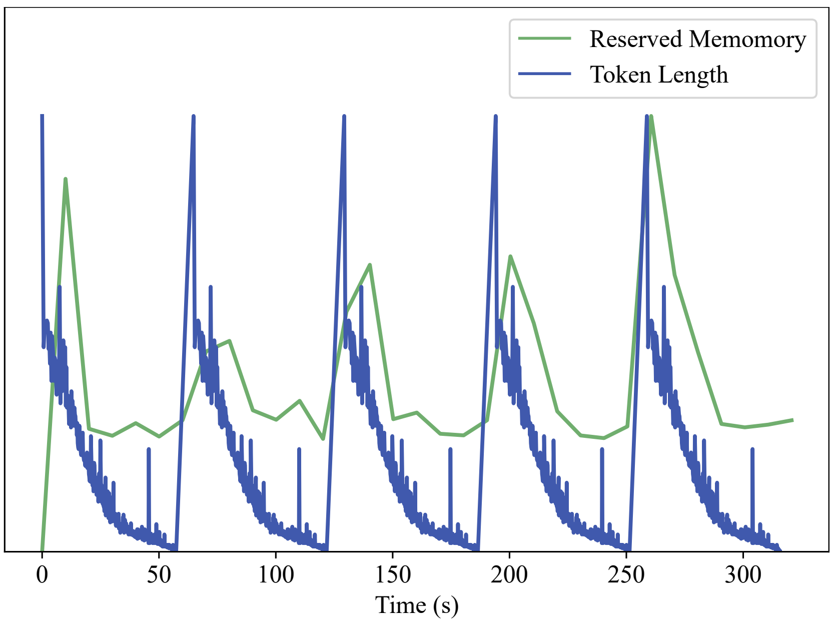


Figure 11: Comparison of Memory Spikes with Token Lengths

spikes, each of which occurs at the start of an epoch of training. The RHMD dataset has a single sample that has a token length of 512. This sample is processed at the start of each epoch, which causes the memory spike. Once the sample has been processed, the memory usage goes back down, since the use of dynamic padding results in the smaller samples that come afterwards having shorter token lengths. Fig. 11 provides proof of this. The graph compares memory usage with the token length of the batch being processed. As can be seen, just after the model starts processing the largest batch, there is a spike in memory usage.

There are various other considerations that must be made when using gradient checkpointing. It is not feasible to discard the output of every single layer, since doing so would result in an unacceptable increase in the time required to recalculate all the values. Instead, specific layers, called 'checkpoints', have their outputs stored while the others have their outputs discarded and recalculated. Additionally, some layers, such as dropout layers, behave differently each time a forward pass is performed on them. If we were to discard the output of a dropout layer, the value that was calculated during the forward pass, which is used to calculate the loss, would be different from the value that is calculated when we run the forward pass again during gradient calculation. This is because dropout layers randomly

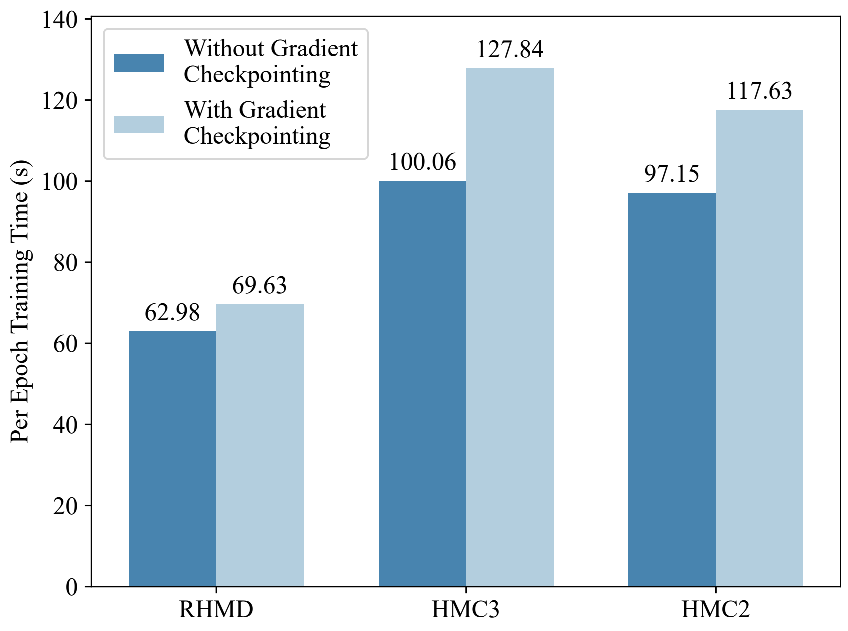


Figure 12: Effect of Gradient Checkpointing on per Epoch Training Time

drop a certain percentage of the previous layer's weights. The specific weights that are dropped will most likely be different on each of the forward passes. Because of issues like this, care must be taken to checkpoint any layers or modules that contain layers that behave in a non-idempotent manner.

The one unavoidable drawback to using gradient checkpointing is that there is some additional computation required in order to recalculate the forward pass values during gradient calculation. The use of dynamic padding in combination with gradient checkpointing, however, reduces this computational overhead to a negligible amount. Fig. 12 compares the results of having gradient checkpointing enabled and disabled with regard to the training time of the BERTweet model on the RHMD and HMC2019 datasets.

7 Ablation Studies

To understand the design choices made in our architectures and the hyperparameters used, we present an ablation study that dissects the contributions of each component towards the final results. We discuss the effect of the use of each ensemble technique and the changes in performance caused by tuning the batch size and weight decay.

From the results presented in table 2, it is clear that the concatenation and feature averaging techniques generally perform worse than the max–voting and probability averaging techniques. A possible reason behind this is the manner in which the concatenation and feature averaging ensembles work. In concatenation, we joined each of the 1024-unit outputs from the three pre-trained models we have used and created a 3072-unit vector. Each unit represents a separate feature. We hypothesised that using all the features together would help improve performance. However, we do not know what each feature represents, so we cannot be sure that using the features from all three pre-trained models together gives the classifier any extra information to work with. A similar issue can be found with feature averaging. Taking the average value for the i-th feature would only work if we knew that the i-th feature for each of the pre-trained models represents the same information. Since this cannot be ascertained, there could be discrepancies caused by the averaging process that cause poor performance. By contrast, max-voting and probability averaging work with the outputs of the classifier, and we know that each output from the classifier represents a class label. As such, taking the average or the maximum vote from the three models at this stage are both sensible approaches that lead to improved results.

Amongst the hyper-parameters, batch size was found to have the most significant effect on performance. Increasing the batch size makes both training and inference faster at the cost of a greater computational cost. The resource optimizations we made, as described in section 6, allowed us to experiment with batch sizes up to 64. The change in F1-Score for the BERTweet model on the RHMD dataset as the batch size is tuned is shown in Fig. 13. The batch size should not have an effect on the generalisability of the model, but our results clearly contradict this. A possible reason could be that larger batch sizes tend to get caught at local minima [30]. However, arguments have been made against this reasoning, suggesting that the generalisation gap is caused by the fact that using a larger batch size results in fewer updates [19]. The gap can be closed by making ad-

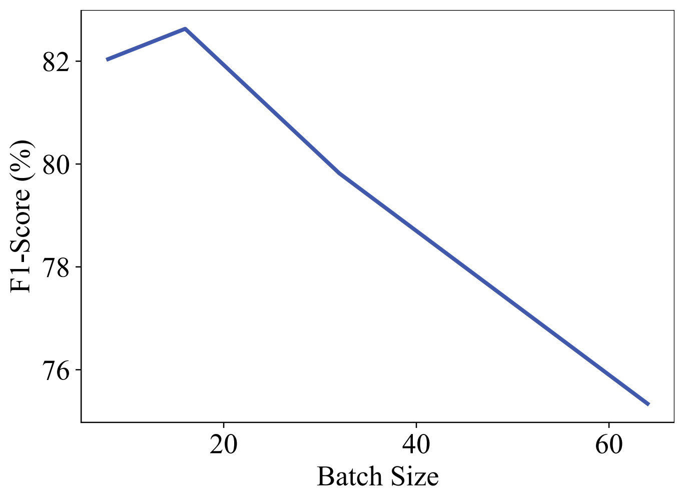


Figure 13: Effect of Batch Size on F1-Score

justments to the training process, such as increasing the number of iterations, but since this would increase the overall training time without improving the model's performance, such adjustments are not being explored here.

Weight decay was also found to have a significant effect on the performance of the models. Weight decay acts as a regularisation mechanism, preventing the model from over-fitting to the training data by limiting the growth of the weights [36].

$$E(w) = E_0(w) + \frac{1}{2}\lambda \sum_i w_i^2 \tag{3}$$

In the equation above, λ is the weight decay parameter. As can be seen, increasing the value of λ increases the error, which in turn forces the weights to become smaller to reduce the error. Because of the way in which this works, using a decay value that is too large should cause under-fitting since the weights will be updated by such a small value that the model will be unable to learn. On the flip side, using a decay value that is too small will result in the weights not being decayed enough, causing over-fitting. In over-fitting, the model adjusts too well to the training data and fails to generalise to the test data. This theoretical reasoning suggests that there is an optimal point somewhere in between that causes neither under-fitting nor over-fitting. Our experimental findings support this theoretical reasoning. The effect of weight decay on the F1-Score for the BERTweet model

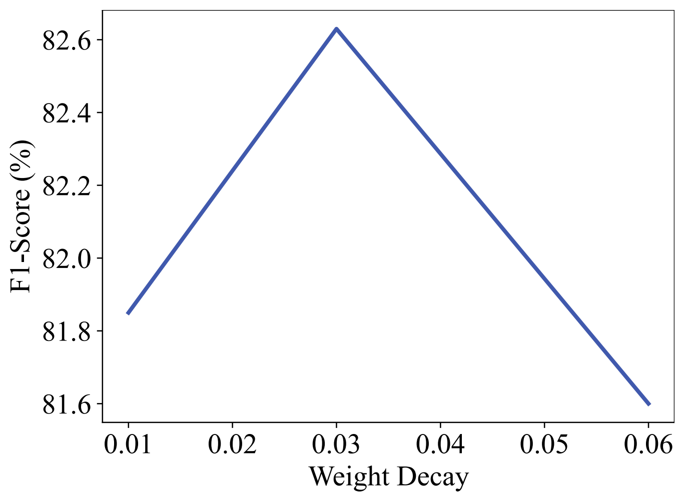


Figure 14: Effect of Weight Decay on F1-Score

on the RHMD dataset is shown in Fig. 14.

8 Explainable Artificial Intelligence

By nature, deep-learning algorithms such as those used in this thesis are difficult to understand. The mechanisms via which the models adjust themselves to the provided input and learn to accurately make predictions are hidden inside a theoretical black box. This makes it difficult to integrate systems built on top of these algorithms into real-life applications. Sensitive use cases, such as those involving healthcare, require the systems to meet strict guidelines. Regulations such as those by the European Union[6] also ensure the right of users to demand a clear explanation behind the decisions made by such systems.

Explainable Artificial Intelligence (XAI) aims to provide these explanations by analysing the models and the factors that lead to the decisions they take. In this thesis, we have utilised Shapley Additive Explanations (SHAP) [42] to perform this analysis. The Shapley value is a concept from game theory that assigns a positive or negative value to the contributors involved in the final outcome of a solution. In terms of machine learning, this means assigning a Shapely value to each of the features from a sample that either contributed towards or against the

[6]https://gdpr.eu/

final outcome. To calculate the Shapley value for a specific feature, we take every possible combination of features that involves the feature we are interested in and compare the difference between the output with the feature present and absent for each combination. The mean of the results is the Shapley value for the feature.

In natural language processing, the features of the input data are the words. Each word has a Shapley value associated with it depending on whether it positively or negatively influences the output predicted by the model. We analyse this influence in two ways. Firstly, we show the influence of different words in specific samples to provide an understanding of how a word affects the output. Secondly, we list the most influential words across the entire test set to provide an overview of the pattern that the model has identified for each classification. To calculate the Shapely value of a word in a specific sample, we provide the sample to the model twice: once with the word masked and once with the word unmasked. The difference between the outputs produced by the model in each case is then compared to calculate the Shapely value.

8.1 Word Influence

Table 4 shows a few samples from the test sets of the RHMD and HMC2019 datasets, along with the influence of each word on the predicted output. The words that contributed towards the prediction are highlighted in blue, while those that contributed against it are highlighted in red. The deeper the colour, the larger the contribution, either for or against the prediction.

Figurative sentences seem to be identified by the use of a disease keyword along with some terms that are completely unrelated to personal health. For example, the sentence ‘I already have puppy fever again.’ is identified as figurative due to the presence of the word ‘puppy’ right before the word ‘fever’. Because of this, the word ‘puppy’ has a strong contribution towards the prediction. On the other hand, the word ‘fever’, being a disease keyword, has the strongest contribution against the prediction.

Negative classifications are identified by the presence of some source of medical

Dataset	Classification	Sample
	Figurative	Nickelodeon posted this to their Twitter feed It's like they had a stroke
RHMD	Negative	Scientists Explore Ties Between Alzheimer's And Brain's Ancient Immune System
	Positive	These pills where prescribed for my OCD
	Figurative	I already have puppy fever again . .
HMC3	Negative	HIV May Double Odds of Heart Attack - WebMD <url> HIV News
	Positive	my anxiety and my depression get so bad sometimes
	Negative	Cartoon Network shows are cancer
HMC2	Negative	What is Parkinson ' s Disease ? <url
	Positive	I Have been having this headache all night . I cannot even sleep

Table 4: Samples Showing Influence of Words Towards Predictions

knowledge along with the disease keyword. For example, the sentence ‘HIV May Double Odds of Heart Attack - WebMD [URL] HIV News’ most likely comes from a post which is sharing a link to an article about the mentioned topic. The most important word here was ‘WebMD’, which is a popular source of medical information. The fact that the words ‘Heart Attack’ are also of significant importance to the negative classification may seem counter-intuitive at first glance. However, it is important to note that a negative personal health mention is still related to health. More often than not, these health mentions are about raising awareness of the disease, which is why it makes sense for the disease keywords to also be important. This point becomes more obvious if we take a look at how positive classifications are identified. In the sentence ‘my anxiety and my depression get so bad sometimes’, the disease keywords are identified as being of high importance. However, the word ‘my’ being present immediately before the disease keywords is also identified as being important to the classification. The presence of such words, which indicate that the content of the post is in relation to the author, is what allows the model to differentiate between a health mention that is personal and one that is not.

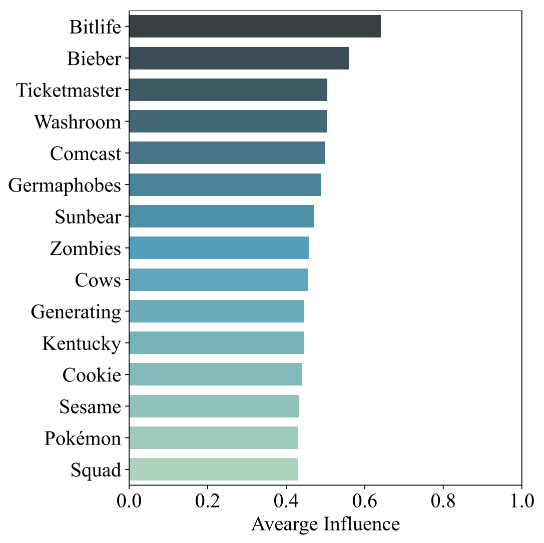
The behaviour of the model when trained on the HMC2 dataset is particularly interesting. In this case, both non-personal health mentions and figurative health mentions had to be classified as negative. The logic behind the classifications remains the same, but it is impressive that the model does not become confused by the combination of the two classes, even though it might intuitively seem like it will. The results shown in section 5 make it clear that the reduced number of classes actually allows the model to perform better than it did for the HMC3 dataset. The analysis of the behaviour of the model on the HMC2 and HMC3 datasets is proof of the model’s ability to understand the context of a health mention. Even though the importance of the actual disease word does not change, the model is still able to identify that both figurative and negative classifications should be labelled as negative classifications.

The analysis above makes it clear that the presence of certain words can have a major influence on the model's decision in support of a particular classification. Fig. 15 lists the words from the RHMD dataset that had the largest influence on average towards each of the three classes in the dataset, figurative, negative, and positive, for the BERTweet model. These charts provide more evidence in support of the explanations laid out earlier in this section. The most important words for the figurative classification are entirely unrelated to health, the ones for the positive classification are nearly all health related, while the negative classification contains a mixture of terms of both types.

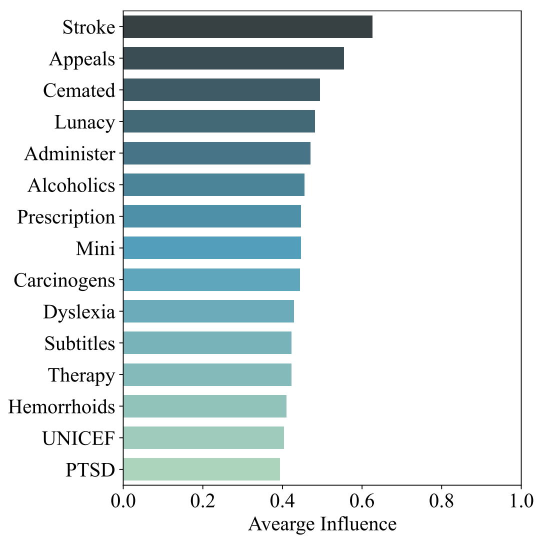
9 Conclusion and Future Works

In this thesis, we made several key contributions to the PHM domain. We introduced computationally efficient mechanisms which reduced the amount of resources required to use existing architectures, and also introduced four ensemble techniques which made use of those architectures. By doing so, we demonstrated that it is possible to achieve state-of-the-art results across the domain using the ensemble techniques without having to face a huge computational overhead. Additionally, we provided methods to explain the outputs produced by our models in support of the push towards XAI.

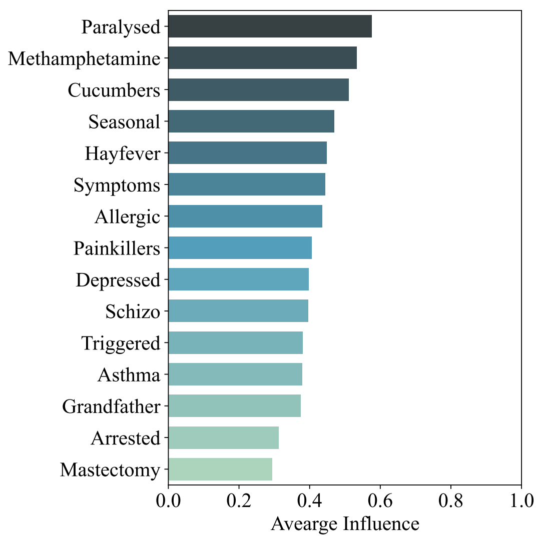
It is important to note that there are important contributions in past work that have not been incorporated into ours, such as the use of sentiment information or adversarial training. This leaves significant scope for future research that examines the effect of those contributions in relation to the ensemble techniques we have introduced. With the use of the resource optimization mechanisms discussed in this thesis, such work should be far easier to experiment with in comparison to the past.



(a) Figurative



(b) Negative



(c) Positive

Figure 15: Words with the Highest Average Influence per Class

References

[1] Tasnim Ahmed, Mohsinul Kabir, Shahriar Ivan, Hasan Mahmud, and Kamrul Hasan. Am i being bullied on social media? an ensemble approach to categorize cyberbullying. In *2021 IEEE International Conference on Big Data (Big Data)*, pages 2442–2453. IEEE, 2021. doi: 10.1109/BigData52589.2021.9671594.

[2] Tasnim Ahmed, Shahriar Ivan, Mohsinul Kabir, Hasan Mahmud, and Kamrul Hasan. Performance analysis of transformer-based architectures and their ensembles to detect trait-based cyberbullying. *Social Network Analysis and Mining*, 12(1):1–17, 2022. doi: 10.1007/s13278-022-00934-4.

[3] Eiji Aramaki, Sachiko Maskawa, and Mizuki Morita. Twitter catches the flu: detecting influenza epidemics using twitter. In *Proceedings of the 2011 Conference on empirical methods in natural language processing*, pages 1568–1576, 2011.

[4] Christos Baziotis, Nikos Pelekis, and Christos Doulkeridis. Datastories at semeval-2017 task 4: Deep lstm with attention for message-level and topic-based sentiment analysis. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 747–754, Vancouver, Canada, August 2017. Association for Computational Linguistics. doi: 10.18653/v1/S17-2126.

[5] Rhys Biddle, Aditya Joshi, Shaowu Liu, Cecile Paris, and Guandong Xu. Leveraging sentiment distributions to distinguish figurative from literal health reports on twitter. In *Proceedings of The Web Conference 2020*, pages 1217–1227, 2020. doi: 10.1145/3366423.3380198.

[6] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.", 2009.

[7] Liangzhe Chen, KSM Tozammel Hossain, Patrick Butler, Naren Ramakrishnan, and B Aditya Prakash. Syndromic surveillance of flu on twitter using weakly supervised temporal topic models. *Data mining and knowledge discovery*, 30(3):681–710, 2016. doi: 10.1007/s10618-015-0434-x.

[8] Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost. *arXiv preprint arXiv:1604.06174*, 2016. doi: 10.48550/arXiv.1604.06174.

[9] Rumi Chunara, Jason R Andrews, and John S Brownstein. Social and news media enable estimation of epidemiological patterns early in the 2010 haitian cholera outbreak. *The American journal of tropical medicine and hygiene*, 86 (1):39, 2012. doi: 10.4269/ajtmh.2012.11-0597.

[10] Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*, 2020. doi: 10.48550/arXiv.2003.10555.

[11] Omar B Da'ar, Faisel Yunus, Nassif Md Hossain, and Mowafa Househ. Impact of twitter intensity, time, and location on message lapse of bluebird's pursuit of fleas in madagascar. *Journal of Infection and Public Health*, 10(4):396–402, 2017. doi: 10.1016/j.jiph.2016.06.011.

[12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, 2019. doi: 10.18653/v1/N19-1423.

[13] Syed Mohammed Sartaj Ekram, Adham Arik Rahman, Md Sajid Altaf, Mohammed Saidul Islam, Mehrab Mustafy Rahman, Md Mezbaur Rahman, Md Azam Hossain, and Abu Raihan Mostofa Kamal. Banglarqa: A benchmark dataset for under-resourced bangla language reading comprehension-

based question answering with diverse question-answer types. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2518–2532, 2022.

[14] Samah Jamal Fodeh, Mohammed Al-Garadi, Osama Elsankary, Jeanmarie Perrone, William Becker, and Abeed Sarker. Utilizing a multi-class classification approach to detect therapeutic and recreational misuse of opioids on twitter. *Computers in biology and medicine*, 129:104132, 2021. doi: 10.1016/j.compbiomed.2020.104132.

[15] Aaron Gokaslan and Vanya Cohen. Openwebtext corpus. http://Skylion007.github.io/OpenWebTextCorpus, 2019. Accessed: 2022-11-22.

[16] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017. doi: 10.48550/arXiv.1706.02677.

[17] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: Decoding-enhanced bert with disentangled attention. In *International Conference on Learning Representations*, 2020.

[18] Pengcheng He, Jianfeng Gao, and Weizhu Chen. Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing. *arXiv preprint arXiv:2111.09543*, 2021. doi: 10.48550/arXiv.2111.09543.

[19] Elad Hoffer, Itay Hubara, and Daniel Soudry. Train longer, generalize better: closing the generalization gap in large batch training of neural networks. *Advances in neural information processing systems*, 30:1731–1741, 2017.

[20] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[21] Adith Iyer, Aditya Joshi, Sarvnaz Karimi, Ross Sparks, and Cecile Paris. Figurative usage detection of symptom words to improve personal health mention detection. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1142–1147, 2019. doi: 10.18653/v1/P19-1108.

[22] Keyuan Jiang, Ricardo Calix, and Matrika Gupta. Construction of a personal experience tweet corpus for health surveillance. In *Proceedings of the 15th workshop on biomedical natural language processing*, pages 128–135, 2016. doi: 10.18653/v1/W16-2917.

[23] Keyuan Jiang, Shichao Feng, Qunhao Song, Ricardo A Calix, Matrika Gupta, and Gordon R Bernard. Identifying tweets of personal health experience through word embedding and lstm neural network. *BMC bioinformatics*, 19 (8):67–74, 2018. doi: 10.1186/s12859-018-2198-y.

[24] Aditya Joshi, Sarvnaz Karimi, Ross Sparks, Cécile Paris, and C Raina Macintyre. Survey of text-based epidemic intelligence: A computational linguistics perspective. *ACM Computing Surveys (CSUR)*, 52(6):1–19, 2019. doi: 10.1145/3361141.

[25] Aditya Joshi, Ross Sparks, Sarvnaz Karimi, Sheng-Lun Jason Yan, Abrar Ahmad Chughtai, Cecile Paris, and C Raina MacIntyre. Automated monitoring of tweets for early detection of the 2014 ebola epidemic. *PloS one*, 15(3): e0230322, 2020. doi: 10.1371/journal.pone.0230322.

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

[27] Mohsinul Kabir, Tasnim Ahmed, Md Bakhtiar Hasan, Md Tahmid Rahman Laskar, Tarun Kumar Joarder, Hasan Mahmud, and Kamrul Hasan. Deptweet: A typology for social media texts to detect depression severities.

Computers in Human Behavior, 139:107503, 2023. doi: 10.1016/j.chb.2022.107503.

[28] Payam Karisani and Eugene Agichtein. Did you really just have a heart attack? towards robust detection of personal health mentions in social media. In *Proceedings of the 2018 World Wide Web Conference*, pages 137–146, 2018. doi: 10.1145/3178876.3186055.

[29] Makoto P Kato, Kazuaki Kishida, Noriko Kando, Tetsuya Saka, and Mark Sanderson. Report on ntcir-12: The twelfth round of nii testbeds and community for information access research. In *ACM SIGIR Forum*, volume 50, pages 18–27. ACM New York, NY, USA, 2017. doi: 10.1145/3053408.3053413.

[30] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016. doi: 10.48550/arXiv.1609.04836.

[31] Alvi Aveen Khan, Fida Kamal, Nuzhat Nower, Tasnim Ahmed, and Tareque Mohmud Chowdhury. An evaluation of transformer-based models in personal health mention detection. In *2022 25th International Conference on Computer and Information Technology (ICCIT)*, pages 1–6. IEEE, 2022. doi: 10.1109/ICCIT57492.2022.10054937.

[32] Pervaiz Iqbal Khan, Imran Razzak, Andreas Dengel, and Sheraz Ahmed. Improving personal health mention detection on twitter using permutation based word representation learning. In *International Conference on Neural Information Processing*, pages 776–785. Springer, 2020. doi: 10.1007/978-3-030-63830-6_65.

[33] Pervaiz Iqbal Khan, Imran Razzak, Andreas Dengel, and Sheraz Ahmed. A novel approach to train diverse types of language models for health mention classification of tweets. In *Artificial Neural Networks and Machine Learning – ICANN 2022*, pages 136–147, 2022. doi: 10.1007/978-3-031-15931-2_12.

[34] Pervaiz Iqbal Khan, Imran Razzak, Andreas Dengel, and Sheraz Ahmed. Performance comparison of transformer-based models on twitter health mention classification. *IEEE Transactions on Computational Social Systems*, pages 1–10, 2022. doi: 10.1109/TCSS.2022.3143768.

[35] Pervaiz Iqbal Khan, Shoaib Ahmed Siddiqui, Imran Razzak, Andreas Dengel, and Sheraz Ahmed. Improving health mention classification of social media content using contrastive adversarial training. *IEEE Access*, 10:87900–87910, 2022. doi: 10.1109/ACCESS.2022.3200159.

[36] Anders Krogh and John Hertz. A simple weight decay can improve generalization. *Advances in neural information processing systems*, 4:950–957, 1991.

[37] Alex Lamb, Michael Paul, and Mark Dredze. Separating fact from fear: Tracking flu infections on twitter. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 789–795, 2013.

[38] Md Tahmid Rahman Laskar, Xiangji Huang, and Enamul Hoque. Contextualized embeddings based transformer encoder for sentence similarity modeling in answer selection task. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 5505–5514, 2020.

[39] Joffrey L Leevy, Taghi M Khoshgoftaar, Richard A Bauder, and Naeem Seliya. A survey on addressing high-class imbalance in big data. *Journal of Big Data*, 5(1):1–30, 2018. doi: 10.1186/s40537-018-0151-6.

[40] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019. doi: 10.48550/arXiv.1907.11692.

[41] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2018.

[42] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30:4765–4774, 2017.

[43] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013. doi: 10.48550/arXiv.1301.3781.

[44] Saif Mohammad. Obtaining reliable human ratings of valence, arousal, and dominance for 20,000 english words. In *Proceedings of the 56th annual meeting of the association for computational linguistics (volume 1: Long papers)*, pages 174–184, 2018. doi: 10.18653/v1/P18-1017.

[45] Martin Müller, Marcel Salathé, and Per E Kummervold. Covid-twitter-bert: A natural language processing model to analyse covid-19 content on twitter. *arXiv preprint arXiv:2005.07503*, 2020. doi: 10.48550/arXiv.2005.07503.

[46] Sebastian Nagel. Cc-news. https://commoncrawl.org/2016/10/news-dataset-available/, 2016. Accessed: 2022-11-22.

[47] Usman Naseem, Matloob Khushi, Jinman Kim, and Adam G Dunn. Rhmd: A real-world dataset for health mention classification on reddit. *IEEE Transactions on Computational Social Systems*, pages 1–10, 2022. doi: 10.1109/TCSS.2022.3186883.

[48] Usman Naseem, Jinman Kim, Matloob Khushi, and Adam G Dunn. Identification of disease or symptom terms in reddit to improve health mention classification. In *Proceedings of the ACM Web Conference 2022*, pages 2573–2581, 2022. doi: 10.1145/3485447.3512129.

[49] Usman Naseem, Jinman Kim, Matloob Khushi, and Adam G Dunn. Robust identification of figurative language in personal health mentions on twitter. *IEEE Transactions on Artificial Intelligence*, pages 1–1, 2022. doi: 10.1109/TAI.2022.3175469.

[50] Usman Naseem, Byoung Chan Lee, Matloob Khushi, Jinman Kim, and Adam Dunn. Benchmarking for public health surveillance tasks on social media with a domain-specific pretrained language model. In *Proceedings of NLP Power! The First Workshop on Efficient Benchmarking in NLP*, pages 22–31, 2022. doi: 10.18653/v1/2022.nlppower-1.3.

[51] Dat Quoc Nguyen, Thanh Vu, and Anh-Tuan Nguyen. Bertweet: A pre-trained language model for english tweets. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 9–14, 2020. doi: 10.18653/v1/2020.emnlp-demos.2.

[52] Dat Quoc Nguyen, Thanh Vu, Afshin Rahimi, Mai Hoang Dao, Long Doan, et al. Wnut-2020 task 2: Identification of informative covid-19 english tweets. In *Proceedings of the Sixth Workshop on Noisy User-generated Text (W-NUT 2020)*, pages 314–318, 2020. doi: 10.18653/v1/2020.wnut-1.41.

[53] Robert T Olszewski. Bayesian classification of triage diagnoses for the early detection of epidemics. pages 412–416, 2003.

[54] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32:8026–8037, 2019.

[55] Michael Paul and Mark Dredze. You are what you tweet: Analyzing twitter for public health. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 5, pages 265–272, 2011. doi: 10.1609/icwsm.v5i1.14137.

[56] Michael J Paul and Mark Dredze. Social monitoring for public health. *Synthesis Lectures on Information Concepts, Retrieval, and Services*, 9(5):1–183, 2017. doi: 10.1007/978-3-031-02311-8.

[57] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014. doi: 10.3115/v1/D14-1162.

[58] Yao Qian, Yuchen Fan, Wenping Hu, and Frank K Soong. On the training aspects of deep neural network (dnn) for parametric tts synthesis. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3829–3833. IEEE, 2014. doi: 10.1109/ICASSP.2014.6854318.

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

[60] Lior Rokach. Ensemble methods for classifiers. In *Data mining and knowledge discovery handbook*, pages 957–980. Springer, 2005. doi: 10.1007/0-387-25465-X_45.

[61] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *54th Annual Meeting of the Association for Computational Linguistics*, pages 1715–1725. Association for Computational Linguistics (ACL), 2016. doi: 10.18653/v1/P16-1162.

[62] Trieu H Trinh and Quoc V Le. A simple method for commonsense reasoning. *arXiv preprint arXiv:1806.02847*, 2018. doi: 10.48550/arXiv.1806.02847.

[63] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30:5998–6008, 2017.

[64] Stefan Wager, Sida Wang, and Percy S Liang. Dropout training as adaptive regularization. *Advances in neural information processing systems*, 26:351–359, 2013.

[65] Chen-Kai Wang, Onkar Singh, Zhao-Li Tang, and Hong-Jie Dai. Using a recurrent neural network model for classification of tweets conveyed influenza-related information. In *Proceedings of the International Workshop on Digital Disease Detection Using Social Media 2017 (DDDSM-2017)*, pages 33–38, 2017.

[66] Davy Weissenbacher, Abeed Sarker, Arjun Magge, Ashlynn Daughton, Karen O'Connor, Michael Paul, and Graciela Gonzalez. Overview of the fourth social media mining for health (smm4h) shared tasks at acl 2019. In *Proceedings of the fourth social media mining for health applications (# SMM4H) workshop & shared task*, pages 21–30, 2019. doi: 10.18653/v1/W19-3203.

[67] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32:5753–5763, 2019.

[68] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical attention networks for document classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 1480–1489, 2016. doi: 10.18653/v1/N16-1174.

[69] Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 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*, pages 19–27, 2015. doi: 10.1109/ICCV.2015.11.