Singapore Institute of Technology - University of Glasgow
Joint Degree in Computing Science Degree Programme

CSC3101 Capstone Project

Please complete the following form and attach it to the Capstone Report submitted.

Capstone Period: 02 SEP 2024 to 23 MAR 2025
Assessment Trimester: Final Trimester
Project Type: Academic

Academic Supervisor Details:

Name: David Miguel Sanan Baena
Designation: Assistant Professor
Email Address: david.miguel@singaporetech.edu.sg
Contact Number: +65 6592-3043

Student Particulars & Declaration:

Name of Student: Ashley Tay Yong Jun
Student ID: 2200795

I hereby acknowledge that I have engaged and discussed with my Academic Supervisor on the contents of this Capstone Report (Problem Definition) and have sought approval to release the report to the Singapore Institute of Technology and the University of Glasgow.

Signature 

Date: 22 MAR 2025

END OF FORM

Singapore Institute of Technology - University of Glasgow
Joint Degree in Computing Science Degree Programme

Final Capstone Report
Enhancing NLP Models for Social Media Sentiment Analysis

For **Final Trimester** from 02 SEP 2024 to 23 MAR 2025

Ashley Tay Yong Jun
Student ID: 2200795

Academic Supervisor: David Miguel Sanan Baena

Submitted as part of the requirement for CSC3101 Capstone Project

Table of Contents

1. Abstract 1
2. Introduction 1
    2.1 Motivation and purpose 1
    2.2 Problem definition 3
    2.3 Project Objectives 7
3. Literature Review 8
    3.1 Overview 8
    3.2 Transformer and BERT 8
    3.3 Related works 10
    3.4 Summary of key findings 15
    3.5 Conclusion of Review 16
4. Methodology 17
    4.1 Overview 17
    4.2 Data sourcing and collection 17
    4.3 Model selection 18
    4.4 Data preprocessing 19
    4.5 Model modifications and enhancements 20
    4.6 Model training and hyper tuning 25
    4.7 Model evaluation 26
    4.8 Web Application 26
5. Results and Analysis 26
    5.1 Overview 26
    5.2 Data Preparation 26
    5.3 Improvement Implementation 36
    5.4 Web Application Implementation 40
    5.5 Training Process 42
    5.6 Results 43
    5.7 Practical Testing 53
    5.8 Discussion 56
6. Project Management 64
7. Conclusion 67
8. Knowledge and Training Requirements 68
9. References 69

Appendices 75

    Appendix A: Code Repository Link 75

Appendix B: Pseudo Codes 75
Appendix C: Web Application Screenshots 78

List of Figures

Figure 1: Timeline of transformer models [35] 10
Figure 2: Machine Learning Process Pipeline 17
Figure 3: Flowchart of data preprocessing flow 19
Figure 4: Flowchart of vocabulary insertion process 22
Figure 5: Weighted Sum Aggregation Across BERT's Final Layers 24
Figure 6: Bar Chart of Sourced Data Distribution 29
Figure 7: Bar Chart of Twitter Emotions (6 Labels) Distribution 35
Figure 8: Bar Chart of SemEval 2017 Distribution 36
Figure 9: Binary sentiment prediction on user interface 41
Figure 10: Compute Resource for Binary Sentiment Analysis Task 53
Figure 11: Word importance value with Base model 54
Figure 12: Word importance value with Proposed model 55
Figure 13: Gantt chart of project timeline 65
Figure 14: Pseudo code for algorithm to perform data split 75
Figure 15: Pseudo code for WHLA wrapper class 76
Figure 16: Pseudo code for vocabulary expansion of model and tokenizer 77
Figure 17: Layout of web application UI 78
Figure 18: Ternary Sentiment Analysis with web application UI 78
Figure 19: Layout of web application UI predicting multiple sentiments 79
Figure 20: Layout of web application UI predicting multiple sentiments (Evaluation mode) 79
Figure 21: Web application API documentation page 80

List of Tables

Table 1: Distribution of number of tokens per text record with default tokenizer 30
Table 2: Sentiment Distribution Across Data Sources 30
Table 3: Sentiment Distribution for Pre-Training Dataset 32
Table 4: Sentiment Distribution for Fine-Tuning Dataset 32
Table 5: Distribution of TFIDF value counts for Slang 38
Table 6: Distribution of TFIDF value counts for Emojis 39
Table 7: Distribution of number of tokens per text record with expanded tokenizer 40
Table 8: Evaluation results for MLM pre-training 44
Table 9: Evaluation results for binary sentiment analysis improvement check 45
Table 10: Weight values for WHLA model in table 9 46
Table 11: Evaluation results for binary sentiment analysis generalization check 47
Table 12: Test results for binary sentiment analysis generalization check 47
Table 13: Evaluation results for binary sentiment analysis (Proposed model) 48
Table 14: Weight values for Proposed model with WHLA in table 13 48
Table 15: Evaluation results for ternary sentiment analysis 49
Table 16: Weight values for Proposed model in table 15 49
Table 17: M-Rec benchmark for SemEval 2017 ternary sentiment analysis 50
Table 18: Evaluation results for emotions sentiment analysis (6 labels) 51
Table 19: Weight values for Proposed model in table 18 51
Table 20: Evaluation results for emotions sentiment analysis (28 labels) 52
Table 21: Weight values for Proposed model in table 20 52
Table 22: Test results for emotions sentiment analysis (28 labels) 52

Acknowledgments

This project would not have been possible without the support of my supervisors. I would like to thank my academic supervisor Professor David for his guidance, encouragement and feedback throughout the duration of the capstone project. I would also like to thank Professor Peter for his constructive criticism and advice towards the capstone during the early phases of the project.

1. Abstract

This capstone project explores the enhancement of sentiment analysis models in the domain of social media, addressing issues in existing models such as poor generalization capabilities across different domains and contextual ambiguity. Existing models struggle to the diverse linguistic expressions used in social media including emojis, slang and sarcasm. To enhance the models, techniques such as vocabulary expansions, additional domain pre-training with mask language modelling and hidden layer aggregation will be implemented on the BERT model. A binary sentiment analysis dataset was sourced that contained data from Twitter, Reddit, Instagram, IMDB and YELP to help address data generalization issues. Experimental results show that the techniques implemented helped to improve sentiment classification for specific cases but still struggled against unseen data and the handling of nuanced language. The findings highlight the importance of data quality and diversity in the domain of social media, where the use of language evolves quickly. Future works on this project will explore more complex techniques of enhancements such as multi-task learning and hierarchical models.

2. Introduction

2.1 Motivation and purpose

Over the past two decades or so, the internet has grown considerably in popularity, size and usage, specifically the area surrounding social media platforms such as Twitter, Reddit, Facebook and even niche forums. From the period of 2005 to 2015, the number of American adults using social media has increased by almost tenfold from a mere 7% [1]. This has garnered high amounts of human interaction through online exchanges via those platforms that are stored and mostly publicly accessible. This large amount of information can prove extremely useful for data mining and analysis for all sorts of industries and purposes. Multitudes of studies, from businesses to researchers have turned their eyes towards these large amounts of information through social media analytics (SMA) [2].

One of those industries involves the growing concern and importance of individuals that suffer from mental health disorders. These individuals are on the rise and there is a large emphasis on early intervention due to links between physical-mental health states [3]. Prolonged exposure to poor mental health would lead an individual to suffer from lack of self-confidence and isolation from social groups, with some cases even leading to suicide attempts [4]. Social

media thus provides a platform where these individuals can perform mental health discussions with others through the expression of their thoughts and experiences through mental health communities. This shared information can give mental health researchers insight into the early signs of mental health disorders [5] through some form of natural language processing (NLP) technique or model [5, 6].

Businesses also use SMA as a tool to gather additional insights into their industry to boost their profits and market share. This is done through using various SMA techniques such as sentiment analysis that can help the business analyze and predict the preference of their stakeholders and customers and thus improving customer engagement which will lead them to perform better in the market [7]. One case study was done on companies such as Costco, Amazon and several others where SMA was performed on the commentary and comments on social media regarding those companies which helped them identify the general sentiment of the customers to the company during events such as offers and cash cards [8].

The political scene can also be observed within social media domains like Twitter, SMA such as sentiment analysis can then be used to analyze comments made by public users. The analysis can be used to identify public opinions towards certain political parties and predict polling results, such information could prove extremely useful towards these parties in terms of planning and strategizing their campaign efforts [9].

As seen from the above industry example, within the domain of SMA, one of the more popular methods used to analyze social media data is the form of sentiment analysis through machine learning and natural language processing. In many cases, sentiment analysis involves the analysis of text information and subsequently sentiment classification using polarities such as positive, negative or neutral, or even emotion invoked from the text [10]. This method presents the opportunity to aid the different industries by better understanding their users.

With the abundance of internet data and areas of application of sentiment analysis models, it will prove worthwhile to invest time and research into improving their performance and accuracy that can in turn benefit the livelihood of many different industries and individuals.

The aim of the capstone is to create a general sentiment analysis natural processing language model that can successfully identify the general mood of any given text. The process of the development of the models will look towards improving existing research in sentiment analysis by exploring the current limitations of natural processing language models and identifying

potential enhancements from current technologies and methods. The improvement process involves training and exploring existing models used in many sentiment analysis solutions by using data extracted from social media and other online platforms.

A web application will be used to house the developed models that users can interact with by providing text information where they will get a response about the general mood of the full text provided. In addition, attempts will be made towards providing contextual reasoning on why the individual that wrote the text feels that way. API endpoints will also be implemented to allow for long entries of data to be fed into the model through multiple API calls.

The idea behind the website is to allow users throughout the internet to be able to identify the general sentiment behind an individual or group of people with ease. An individual can feed the text of comments inside a trending twitter topic to find out how people feel towards that topic. Businesses can use the API to iterate through a series of comments on their product and allow them to find out how their customers feel towards said product. The freedom of access to a sentiment analysis model can allow individuals, organizations and researchers to quickly analyze large amounts of data.

2.2 Problem definition

There has been no shortage of studies and research poured into sentiment analysis NLP models within various domains. This includes developing existing pre-trained models such as BERT, domain-specific trained models, models developed using traditional machine learning techniques as well as models attempting to be used in industry related scenarios such as businesses and healthcare.

BERT [11], also known as Bidirectional Encoder Representations from Transformers is a pretrained deep learning model developed in late 2018 by Google that provided revolutionary changes to the preexisting NLP models like word2vec that year. BERT was trained on large amounts of text data allowing it linguistic patterns and structure of the English language. It was also trained on “mask language modeling (MLM)”, a task where the model is required to predict a randomly masked word of a text [11], and “next sentence prediction (NSP)”, identifying whether the second sentence logically follows the 1st sentence [11]. These training methods allowed it to significantly increase its performance and understanding and thus, BERT became a basis of many different sentiment analysis models used in various studies today [12, 13, 14, 15].

TwitterBERT [16] represents one of the domain-specific trained models trained on top of BERT. This model was trained towards performing sentiment analysis on Twitter data using a 4-stage framework. The process involves further pretraining BERT using Twitter datasets like sentiment140, an annotated Twitter dataset along with other specialized Twitter datasets. The training also involved similar learning tasks for BERT, making use of MLM and NSP. Various classification models such as convolutional neural networks [17] and linear classifiers [18] are used to further optimize the sentiment analysis result. Lastly, blending is used to further improve the performance by combining the representations from pre-trained word embeddings with BERT's sentence-level representations, thus allowing the model to better understand both individual words and sentences and make a more accurate sentiment analysis [16].

Besides BERT, there are other studies that explored the field of sentiment analysis using traditional machine learning methods or other proposed methods. These studies used machine learning algorithms such as Support Vector Machines [19] and Naive Bayes [20] to train the model and perform comparisons against neural networks. Their training process for the model is similar, where they gather a dataset from an online social media platform such Twitter, perform the necessary preprocessing steps for NLP training such as bag of words and word embeddings. Afterwards, a model is selected and trained on the processed dataset with a test train split defined by the study [21, 22, 23].

Despite the recent advancements in technology surrounding NLP models, there still exist several problems that have not been solved entirely. These problems include the lack of ability for an NLP model to generalize across several domains, this issue stems from the lack of diverse and quality datasets that the model is trained on. Contextual ambiguity is another area that has been a major focus NLP research as basic models are unable to perform well when presented with complex sentences involving polysemy, idiomatic expressions, or nuanced contextual meanings. Lastly, the presence of domain specific words can largely impact the model's understanding of the context of the given text as a model only has a limited set of vocabulary to work with.

In sentiment analysis and training of the NLP models, the scope and quality of the data is extremely important and directly affects the performance of a model in different domains and industries. A study made in the context of discourse parsing investigated why models fail to generalize and found that providing a diverse training data set has shown to significantly improve the model performance across multiple genres and unseen text [24]. The finding can

translate into the importance of data for training and testing the sentiment analysis model. For example, a lack of scope and quality can cause a model to fail to generalize different context and languages across different social media platforms. A domain specific model like TwitterBERT may not perform as well on data coming from Reddit and Facebook. Many NLP models like those mentioned above today only look towards major social media platforms like Twitter to get data, however, the internet is full of diverse information and additional sources that can help improve model generalization.

Contextual ambiguity is another problem that many NLP models suffer from and has yet to be fully addressed despite significant efforts of research poured into it. This ambiguity comes from complex and hidden meanings behind human written text such as sarcasm, metaphors or cultural/social forms of emotional expression Models like BERT has been able to work towards addressing this issue by applying bi-directional context with the use of MLM and NSP tasks during training, allowing the model to better understand the meaning and context of the words in a text. However, in cases where there are long sentences that result in long range dependencies of words that could impact the meaning of the text, or if there is a severe lack of context from the text itself, BERT might still struggle to make a correct analysis due to its limited input size of 512 tokens, thus this context information could be lost across multiple chunks of text [11, 25, 26].

There is also the subjectivity and ambiguity in language use within online social media platforms. These language usages include emojis, slang, typos, abbreviations and cryptic words, which can also affect the text's sentiment. The use of an emoji at the back of a sentence may change the meaning of the sentence to convey sarcasm instead of genuine feelings. NLP models can thus often misinterpret the sentiment of a user generated text due to these elements. Even BERT, the model that has significantly improved the contextual understanding of NLP models compared to traditional models has shown to have struggled in some form when trying to classify comments with these elements [15].

Based on these sub problems, the capstone project will aim to identify relevant literature in the field of NLP that has worked towards solving these issues present. By identifying current techniques used to mitigate the problems presented, these techniques can be combined or improved upon to help create a robust sentiment analysis model for use in social media and online platforms. An initial review of literature was conducted below on some proposed ideas on how to solve the present sub problems.

Training Data

For the issue surrounding the training data for the model, the data preprocessing pipeline can be refined to ensure sufficient levels of data scope and quality for the training of the NLP model. This includes collating datasets from beyond Twitter, using information from Reddit, Facebook and even other niche forums. Data augmentation techniques such as Syntax-Tree Transformation to covert a sentence from active voice to passive voice without changing semantic meanings, or Easy Data Augmentation where words are replaced with synonyms can also be applied to artificially increase and diversify the dataset size [27]. By identifying a training dataset from diverse sources of good quality, this issue can be mitigated and allow the sentiment analysis model to capture even more unique nuances in language and context, improving its performance.

Contextual Ambiguity

One possible approach to solve this is by taking pretrained models like BERT and modifying it by adding and altering its architecture. This can be done by implementing hierarchical models [28] to model sentence level and document level sequences separately, multitask learning [28, 29], training the model to predict masked words, sentiments and context, or even a combination of both methods. Understanding the text context and making an accurate prediction is significant especially when applied in real world applications, misinterpretation could result in consequences across different industries such as a wrongly perceived customer feedback or an inaccurate analysis of a patient suffering from mental health disorder. By addressing this issue, the model can hope to handle complex and nuance text and improve its applicability in many different areas.

Domain language

The approach to this issue involves incorporating the same solutions as our data scope and quality problem, by refining the data preprocessing pipeline and ensuring that the dataset used has presence of social media type language which can be used to train the NLP model. In addition, the training process should extend the word embeddings [30] towards abbreviations and emojis to ensure it also captures relationships of these elements beyond just normal text. As mentioned earlier in the introduction, the increase in use of social media worldwide means that the presence of social media slang will be more prominent in all sorts of industries. To ensure that sentiment analysis methods used by these industries on social media remain up to date and relevant, it is important to address this issue to allow NLP models to tackle more complex and nuanced user generated content and be applied in industries with more freedom without the worry of data cleaning.

To conclude, the advancement in competency of NLP models in relation to sentiment analysis has seen significant improvements within past years, however, sub problems such as data scope and quality, contextual ambiguity and language subjectivity remains to be fully addressed. The lack of use of diverse datasets from social media, the requirement of high numbers of inference for NLP models across long ranges of text and the informal nature of social media language online represents area where future NLP models can work on for improvements, by addressing these issues, the applicability of these models can be enhanced throughout different industries.

2.3 Project Objectives

- Develop sentiment analysis model
    - Create a general sentiment analysis model by using data from social media platforms and other areas of the internet to identify the general mood of any given text. This mood classification can range from general sentiment classifications like polarity from positive, neutral and negative or more advanced classifications such as emotions like happy, sad, nostalgic.
- Improve upon existing NLP techniques and models: BERT [11]
    - Aim to address the sub problems identified in current existing solutions such as data scope and quality, contextual ambiguity and language subjectivity in social media.
    - To identify and review current literature as well as their techniques implemented to solve those issues and combine/improve upon those techniques.
    - To meet or exceed current industrial standards for sentiment analysis models used in industries.
- Develop web application and API end points
    - Create a webpage to house the sentiment analysis models developed in previous objectives. Design a user-friendly interface that can allow users to choose a model, feed text into the NLP model and receive a response. Provide an API endpoint that can reach the NLP models to allow users to perform multiple API calls across a series of text which can be applicable for large amounts of data.

The overall target for the capstone project is to develop and experiment with different sentiment analysis models using different methods such as building upon pretrained models, combining

different NLP techniques and refining the machine learning process. The result will be a model that performs similarly or better than current existing solutions in the sentiment analysis of social media data. These targets also include:

- Accuracy and performance scores that rivals current existing solutions or better.
- Applicability throughout different social media platforms and industries by showing generalization capabilities on unseen data

3. Literature Review

3.1 Overview

Sentiment analysis represents a branch of Natural Language Processing where it can identify the underlying sentiment and emotion of a given text. The following literature review will mainly cover related works where sentiment analysis is applied in the field of social media and the internet along with their techniques used to solve the existing problems faced when performing sentiment analysis. In addition, studies that have implemented or proposed techniques to improve the performance of Natural Processing Language models will also be reviewed.

A brief overview of the review includes a general analysis on the performance of transformer models and the Bidirectional Encoder Representation from Transformers (BERT). Studies that use BERT in sentiment analysis will be reviewed and analyzed for their methods and techniques. Studies that proposed techniques or methods to improve and fine tune BERT outside of sentiment analysis will also be relevant for analysis. Some of the related studies mentioned in the introductions will also be expanded upon in this section to provide a more comprehensive analysis of those works. The aim is to identify key techniques that are used by these studies that can be leveraged, combined or improved upon that can be applied to the methodology for the creation of the sentiment analysis model for the project.

3.2 Transformer and BERT

Sentiment analysis can be implemented through various means, whether it be traditional machine learning methods, the use of a transformer architecture model or large language model. Transformer models strike a balance between the spectrum of complexity from simple machine learning model methods and large language models. A performance analysis on BERT was able to showcase the significant improvements of the BERT model over previous

NLP models that relied on traditional machine learning. On the other hand, a study on the application of transformers and large language models in the field of genome, highlighted the tradeoff between performance and compute cost, where even though transformers were able to obtain superb results, they were still outperformed by large language models due to the nature of model complexity [31], and thus reflects the importance of selecting a proper model based on the specific requirements of the task and resources available to ensure a balance in performance and computational cost.

The transformer architecture was a predecessor to BERT, which was released 2017 and provided significant advancements in the machine learning industry by representing an alternative to the need for convolution and recurrence within a model. It introduced the concept of self-attention mechanisms which in turn enabled the use of parallelization during training that allowed it to be trained efficiently on modern GPUs and thus led to superior performance and reduced computational cost [32]. BERT, which was briefly talked about in a related work in the introduction of this report, made use of the encoder portion of the transformer architecture allowed it to perform pre-training with bidirectional representation via task like Masked Language Modeling (MLM) and Next Sentence Prediction (NSP). [Bert paper]

The performance analysis mentioned earlier also noted that BERT's ability to fine tune into specific domains to perform various tasks. Some examples of these domain specific adaptations include BioBERT and SciBERT, where they are pre-trained with a domain specific corpus to improve their performance in their specific domain. These areas highlight BERTs ability to perform transfer learning as well as adapt to various tasks across different domains. [11]

In addition to BERT, the transformer architecture led to major developments in different types of models that incorporated the architecture, all of which included various features that aimed to innovate and improve upon the architecture. In figure 1, it can be observed that as newer models release over time, their number of parameters far exceed the number of parameters that BERT has. Some of these models include RoBERTA [33] and DeBERTA [34], both of which aimed to improve upon BERT. RoBERTa for example, aimed to optimize BERT by removing the Next Sentence Prediction (NSP) as it was found that NSP may not contribute as significantly to downstream performance and instead introduced unnecessary complexity to the model, their findings also shown how models trained without NSP outperformed other models that relied on identifying sentence relationships [33].

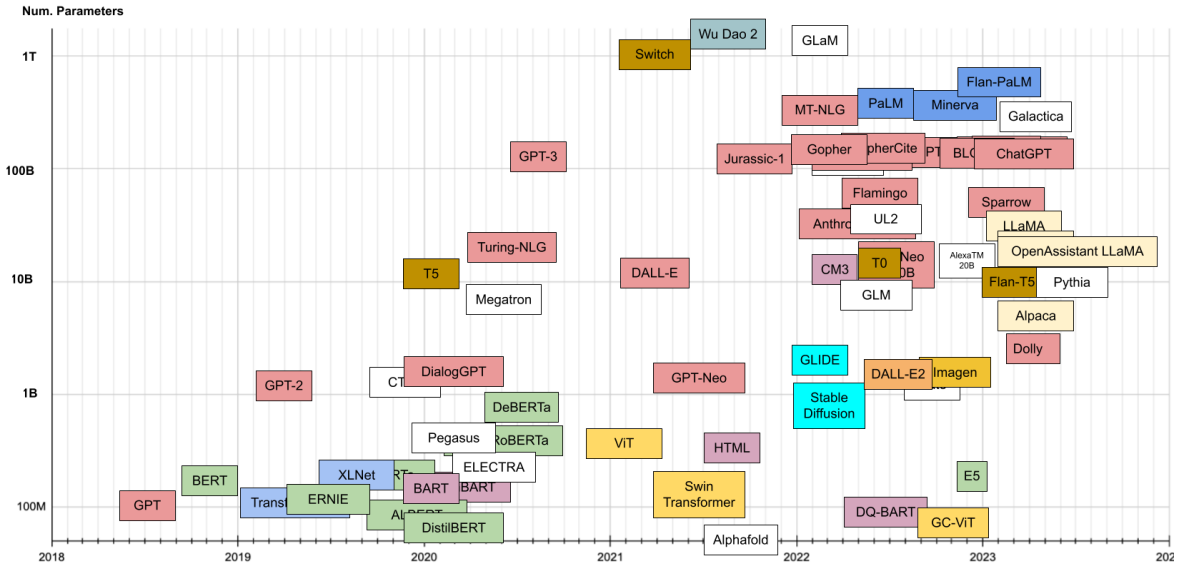


Figure 1: Timeline of transformer models [35]

A comparison study was done between the performance of BERT, RoBERTA and DeBERTA and found that both RoBERTA and DeBERTA was consistently outperforming BERT at the cost of higher computational cost. It was also noted that while RoBERTA contained more parameters than BERT, the tradeoff between performance and computational cost favored the performance aspect and that it is preferable to use RoBERTA to achieve better results unless computational resources is limited [36].

3.3 Related works

TwitterBERT

TwitterBERT is a BERT variant that was designed to conduct sentiment analysis in a social media setting like Twitter [16]. The study proposed a 4-stage framework that involves fine-tuning BERT by providing further pre-training as well as combining pre-trained words embeddings together with BERT embeddings that can ultimately improve the performance of sentiment analysis on Twitter.

The 4-stage framework involves taking the pre-trained BERT model and performing an initial pre-training with a large Twitter dataset like sentiment140 and TSA, making use of unsupervised learning functions originally used by BERT, the next sentence predictions and mask language modelling. Next, intermediate pre-training is performed by using smaller unlabeled datasets from multiple test datasets using the same unsupervised learning methods. The third phase represents the fine-tuning portion of the framework, where the study proposed

4 different types of models that will be added on top of the BERT encoder and trained simultaneously to help perform classification, these models include a linear layer, a convolutional neural network, a bidirectional LSTM as well as BGRU. The last phase involves combining the pre-trained word embeddings from static embeddings like word2vec and GloVe along with the BERT representations of the word before being fed to the classifiers. An ensemble method was used for the classification, whereby an aggregation is performed between the predictions of the 4 models for a given text, in this case, a voting mechanism is used to perform prediction.

The model was then tested on various SemEval datasets to analyze its performance where it was able to obtain the highest score of an F1-score of 71.82% on 1 of the datasets. Additionally, it was noted that the models that were under further pre-training as well as the embedding concatenation were consistently outperforming the baseline models. Next, it was also noted that the linear classifier was also outperforming the other the other 3 models used. Lastly, the ensemble method that was proposed did not show any significant improvements to the results compared to using an individual model.

There was a clear acknowledgment and focus on the current challenges faced in sentiment analysis conducted in a social media setting like Twitter where there is heavy use of informal language and slang. However, there was a lack of hypothesis and explanation on how the models that were implemented on top of BERT were able to improve the results or why certain models would outperform one another.

The methodology proposed was clear and direct, incorporating the use of transfer learning and pre-training along with their proposed innovation of blending the embeddings of BERT and static embeddings together to provide more semantic richness. Their experimentation was justified, using a baseline model to perform classification alongside their own model for comparison. There was a lack of explanation on the possible noise and nature of each dataset and the paper did not fully justify the reason for the selection of their classifiers that were used on top of BERT.

Their methodology that involved an initial pre-training of BERT on domain specific data was also observed in other studies using BERT such as BioBERT and SciBERT, solidifying the fact that transfer learning is a strong suite of BERT. Their proposed methods to perform concatenation of their BERT embeddings along with static embeddings can also be found in other studies in the domain of NLP in social media, such as a study on AraBERT where they

used static embeddings processed through Convolutional Neural Network to combine with BERT embeddings leading to performance increase of the model [37]. Another study on the application of BERT on English tweets also performed the word embedding concatenation after being fed through a Gated Recurrent Unit layer and found that it resulted in better F1-scores for sentiment analysis than using either the BERT embedding or static embeddings alone [38].

Overall, the proposed innovation from TwitterBERT, specifically their 4-stage framework proved to be able to improve performance of sentiment analysis for social media. Their techniques used provides areas that can be leveraged and used in the model to be created for this project such as pre-training and embedding concatenation. However, there are also areas that were not addressed in the paper such as the lack of direct handling of the social media vocabulary and emojis and the computational efficiency needed to train the model on such a large dataset.

Rethinking of BERT sentence embedding

The following study "Rethinking of BERT sentence embedding for text classification" proposes multiple different methods to help improve the performance of BERT when it comes to text classification problems [39]. The study mentions that the common [CLS] token used by BERT might not be the most optimal method to perform classification. It also makes note that methods such as aggregation architectures as well as freezing the BERT layers may result in an improvement in performance in tasks such as sentiment analysis and sarcasm detection. Lastly, multi-task learning is proposed to help further improve the performance in classification areas that could be related.

The implementation of the model involves several different aggregation architectures, making use of the BERT hidden layers and applying different techniques. Pre-trained BERT models like MARTBERT and Qarib, BERT variants that were trained on large arabic datasets were used to implement these techniques. These techniques include performing an average or hierarchical aggregation of the last 4 hidden layers of BERT to generate the final output. Other techniques aim to reduce the computational load required by BERT by only aggregating the final layer of BERT and ignoring the use of embeddings from all the other hidden layers inside BERT. An arabic sarcasm/sentiment dataset, "ArSarcasm-v2" was used as the labeled dataset for both sentiment analysis and sarcasm detection. Multitask learning is also implemented here, comparing the use of shared and separate BERT models across multiple tasks. Lastly, BERT layers were frozen and used to be compared to traditional machine learning fine-tuning methods.

The models were then tested on the test set of the same dataset, and it was found that freezing BERT layers provided similar performance to fine-tuning procedures and can even outperform it. The multitask implementation also provided good performance and results, achieving an F1-score for sarcasm detection of 64.41. Lastly, performing the hierarchical aggregation with the last 4 hidden layers of BERT provided a significant improvement to model performance.

The paper showcased a clear focus on investigating different innovative techniques to improve text classification tasks and showcased experimental results that support their hypothesis. However, there are some areas of concern that were not addressed such as the lack of validation that these techniques would be able to generalize well, beyond the arabic language, the “ArSarcasm-v2” dataset as well as the pre-trained models used. There was also a lack of explanation on how specific aggregation architectures may outperform one another and the limitations that may come about from freezing BERT layers.

Despite the lack of explanation for certain areas, their findings strongly coincide with other studies in the field of experimenting with BERT hidden layers. One study performed hidden layer aggregation in the context of enhancing multi-class text classification models on the English language using BERT, they performed different sorts of aggregation such as obtaining the [CLS] token from the 12 hidden layers of BERT and passing them sequentially through an LSTM before performing classification, leading to increase in model performance [40]. SK BERT is a BERT variant that was proposed and incorporated the use hierarchical weighted aggregation of the last 4 hidden layers of BERT, similar to one of the proposed methodologies proposed in the main study mentioned above that also resulted in an increase in model performance for different task such as sentiment analysis, news topic classification and even subjective vs objective classification. [41] These helped to solidify the fact that the hidden layer aggregation technique can be generalized to other languages and different tasks. Research also indicates how the earlier layers focus more on the structural features of language while the deeper layer of BERT captures the semantic representations of the text [42].

To conclude, the techniques proposed in the study provides insight to the capstone project, these techniques can be leveraged upon in the methodology to help tackle the challenges faced in the sentiment analysis task in social media, particularly the aggregation architectures as well as the freezing of BERT layers, allowing for state-of-the-art accuracy even with limited data and resources.

exBERT

exBERT is a BERT variant that focuses on extending the vocabulary of BERT with domain-specific vocabulary [43]. The hypothesis states that to adapt general pre-trained models to different niche domains, vocabulary from said domains can be inserted into the vocabulary of BERT to help improve performance. Thus, a lightweight extension module was developed for exBERT to help accommodate the domain vocabulary.

The study made use of BERT-base-uncased as their baseline model for training as well as a limited biomedical corpus. In the proposed method, each transformer layer in BERT has an extension module, the original layers of BERT will be frozen while the weights of the extension module will be open for training. The tokenizer embedding layer was also extended with the extension module to handle domain-specific tokens that are not recognizable by the original embedding layer. The final output is calculated using a weighted sum mechanic where they combine the output from the original BERT layer and the extension module, each with their own specified weights.

The model was then evaluated on the MTL-Bioinformatics-2016 dataset, against other BERT variants that were trained to perform in the same domain such as BioBERT and SciBERT on task such as Named Entity Recognition and Relation Extraction and looked at areas such as macro F1-scores for performance comparisons. The experiments showed that in a low resource setting, exBERT was able to outperform the other models that it was benchmarked against, with results indicating that the model consistently obtained higher F1-scores than those models. Thus, it was concluded that this technique of vocabulary expansion can likely be generalized to other domains that have unique and niche language formats and vocabulary.

The paper highlighted an innovative approach to improving models with their proposed module and was able to achieve results that aligned with their initial hypothesis. However, there were areas of their proposed methodology that did not have sufficient justification such as the reasoning behind using a weighted sum mechanism. In addition, their large focus within the biomedical field for this study may limit the generalizability of their proposed solution. There was also a lack of exploration between the tradeoffs of performance increase and computational cost as increasing the vocabulary size will lead to an increase in complexity of the model.

Despite these omissions, their conclusions also largely align with other studies that explore the field of vocabulary expansion of BERT. Vocabulary expansion has been proposed in studies

as a method of fine-tuning BERT to achieve better performance [36]. One study where this technique was implemented is the creation of TVD-BERT. Part of its methodology is the direct insertion of 500 domain specific vocabulary words into the BERT vocabulary embedding layer which contributed significantly to correctly predicting domain specific words during additional pre-training as well as the downstream task for TVD-BERT [44]. Another study also explored the inclusion of compound words in BERT as part of its process for domain fine-tuning where synonym word embeddings were used as part of the vocabulary expansion of BERT in this study and it was found that it obtained similar results as other vocabulary expansion techniques such as the other two studies mentioned above [45].

3.4 Summary of key findings

The literature review highlights three primary strategies to incorporate improvements in BERT based models. These strategies include vocabulary expansion, transfer leaning through pre-training as well as hidden layer aggregation. Transfer learning, as seen from the implementation of TwitterBERT, BioBERT and SciBERT showed how leveraging pre-training on domain specific datasets can help to fine tune BERT to that domain. Vocabulary expansion, as demonstrated in models like exBERT and other similar studies showed how BERT can still perform in a low resource setting by expanding the vocabulary library of BERT via incorporation of domain specific tokens. Lastly, hidden layer aggregation as explored in studies experimenting with hidden layers of BERT has shown the ability for BERT to better capture semantic representations by aggregating the outputs from multiple hidden layers.

There were also other techniques mentioned in the literature above proposed for fine-tuning BERT, including embedding concatenation of BERT embeddings and static embeddings, and multitask learning. While these techniques also contributed to the increased performance of the models, they were not explored fully in the studies reviewed. In addition, applying too many techniques to improve the model may lead to an over complex model which would affect the computational cost of the model. Techniques like performing embedding concatenation may also conflict with the vocabulary expansion due to the nature of word embeddings, leading to gaps or conflicts when performing training and classification such as semantic mismatch, where the values of the static embeddings do not align with the trained embeddings from the vocabulary expansions.

An additional thing to note is that none of the reviewed studies explored the issue of datasets and models' ability to generalize beyond a specific domain and instead only explored a specific

domain. The initial literature exploration in the introduction has explored this area of improvement from a study on discourse parsing thus, it can likely be a potential area where the model can leverage upon to generalize better.

Within the family of transformers, there have been other variants of the BERT model like DeBERTa and RoBERTA that have been shown to outperform BERT by performing further fine-tuning at the cost of higher computational power. However as seen by the relevant studies performed above, many of them chose to use BERT and implement their improvements on top of the model. Their analysis has still shown that despite being of the simplest architectures, BERT presents an excellent opportunity and foundation for testing and developing new techniques as well as operating in a resource limited environment while still achieving good results. It can be noted that some of the optimizations from RoBERTA and DeBERTa can be brought over to be used by the BERT model such as the exclusion of the usage of NSP, which mentioned earlier, removed the additional complexity of the model and improved performance.

3.5 Conclusion of Review

To conclude, the literature review offers insight into the versatility and adaptability of the BERT models and various NLP tasks including sentiment analysis. Techniques such as transfer learning via additional pre-training, vocabulary expansions and hidden layer aggregation have proven to be able to improve performances of BERT models and addresses existing problems faced such as contextual ambiguity and the handling of domain specific vocabulary. The studies show how a combination of techniques can together improve a model, such as TwitterBERTs embedding concatenation and domain specific pre-training, boosting the models semantic understanding and classification accuracy. However, these improvements must be regulated to avoid over-complexity of the model. Despite the advancements in transformer base models in recent years such as RoBERTa, DeBERTa and many other models, with parameter counts reaching over 100B, BERT remains as a foundational and resource efficient model for research studies, presenting itself as an excellent candidate for the model to be selected for the capstone project.

4. Methodology

4.1 Overview

The methodology for this project involves designing a sentiment analysis model based on the conclusions drawn from the literature review. The design will cover the machine learning process pipeline from the start to the end.

Figure 2: Machine Learning Process Pipeline

The above diagram provides the general flow of a machine learning process pipeline. The machine learning process pipeline mainly follows the pipeline architecture of traditional machine learning models as well as other studies mentioned in the literature review.

Based on the above reference, a specialized machine learning process pipeline is created for this project with some extra elements such as selection of the model and tokenizer. Each part of the proposed machine learning process pipeline will be considered a separate sub system and split into the following sections where it will be expanded upon:

- Data sourcing and collection
- Model and tokenizer selection
- Data preparation and preprocessing
- Model fine-tuning and enhancements
- Model training
- Model evaluation and analysis

4.2 Data sourcing and collection

Data represents one of the most important aspects when it comes to training any machine learning model, the quality and diversity of the data will affect what the model learns and how the model will perform the classification after training. Providing the model limited data of poor quality will lead to the model performing well only on the training dataset. However, if the model were to be tested against unseen data, it would not be able to perform well due to its poor ability to generalize.

Based on previous literature mentioned above, providing a diverse dataset can lead to an increase in model performance and its ability to generalize well [27]. To ensure the model will be able to perform well across multiple social media and internet platforms, data to be used for model training and validation should encompass data from multiple sources instead of a single source to train the model.

4.3 Model selection

Based on the key insights and conclusions derived from the literature review, BERT will be chosen as the base model of this project to be trained. While RoBERTa and DeBERTa as well as other complex transformer models prove to be able to outperform BERT is many aspects, the computational cost must be considered especially due to the limiting factors as mentioned in the project timeline. The simpler architecture and robust nature of BERT allows the project to test the implementations of the techniques without worrying about additionalcomplexity added in previously and still obtain good performance. Many of the studies reviewed in the literature review also chose BERT as their base model to perform the implementations, by using BERT as the base model, there can be a fair comparison between the pre-trained models proposed in the studies reviewed against the models created in this project. The version of BERT to be used will be the “bert-based-uncased" model from hugging face repository.

The BERT tokenizer, a tokenizer created specifically for BERT will be used alonside the BERT model to ensure the outputs from the tokenizer align with the requirements for BERT. This tokenizer uses the WordPiece tokenization algorithm which breaks words into smaller sub words or characters if the word is not in the vocabulary, enabling efficient handling of out of vocabulary cases. The tokenizer also includes the use of special tokens such as the popular [CLS] token that has been mentioned in reviewed literature, [SEP] tokens that is used to separate two different sentences as well as [PAD] tokens to ensure the fixed length of 512 tokens.

4.4 Data preprocessing

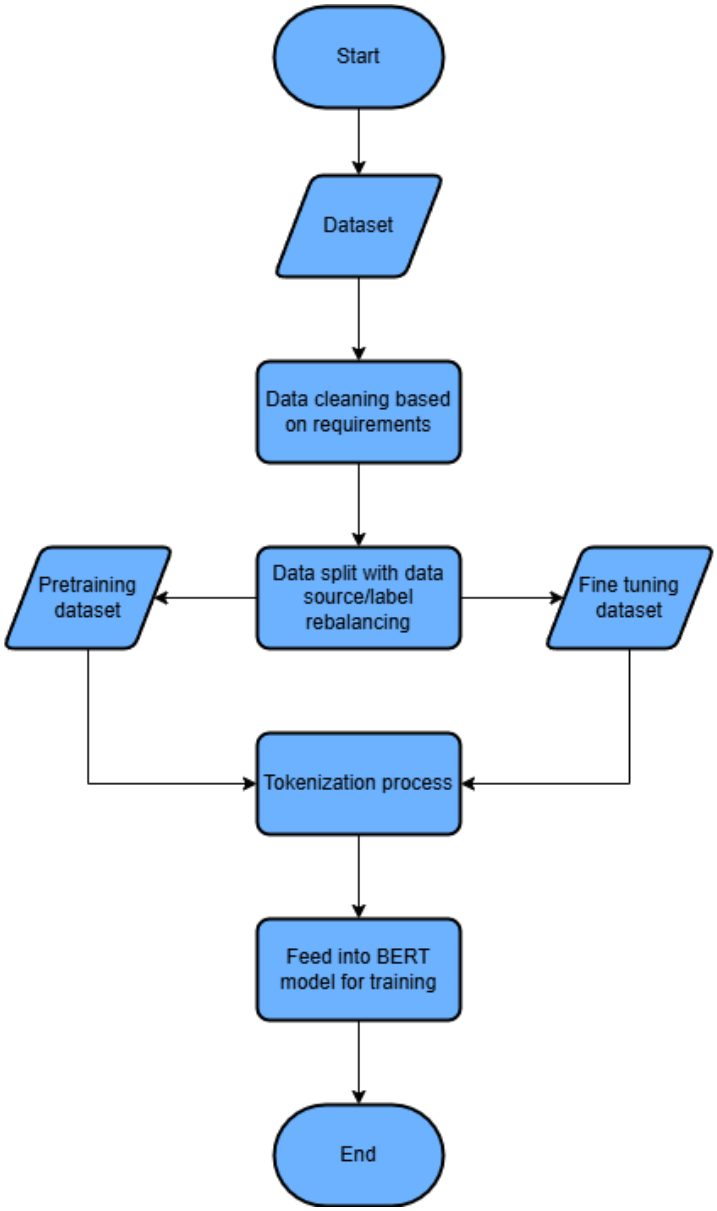


Figure 3: Flowchart of data preprocessing flow

The data preprocessing process will mainly follow standard machine learning preprocessing procedures to ensure that the data being fed into the model for training is valid and will not interfere with the training process and model performance. The above flowchart showcases the general data preprocessing flow of the data input, the dataset, until it reaches the BERT model for training. The datasets obtained from section 4.2, will be analyzed and collated into a single dataset which will be used in the capstone project. Additional datasets may also be sourced out to help perform evaluations, comparisons and benchmarks for the models. For all the

datasets, data preprocessing will be conducted in the following ways:

- Removal of URLS present in the text
- Removal of any @ or r/ mentions towards users used in various social media platforms, for e.g. @ is used in Twitter to tag others in a post
- Removal of any non-alphabetic characters except for emojis
- Convert all text to lower case and remove any trailing white space

Next, the combined dataset will be broken up into multiple datasets to help facilitate the full training procedure. First, a subset of data will be taken from the full dataset and will be deemed as the pre-training dataset, this dataset will be fed to the BERT model for the additional pre-training to fine tune it for the domain, this aspect will be explained in a later section. The remaining data will be split into the training dataset and testing dataset, the amount of labelled data will be balanced to ensure that the model learns effectively across all categories or classes, reducing the risk of bias toward categories that may be overrepresented. The training dataset will be used to fine tune the BERT model on downstream task, in this case, to perform sentiment analysis while the test dataset will act as the unseen data which will be used to test and evaluate the performance of the BERT model after the training has been completed.

Each dataset will undergo a tokenization process using the BERT tokenizer to convert the raw text contained in the dataset into tokens and their assigned token IDs. A Torch dataset will be used to create a dataset to house the encodings provided by the BERT tokenizer along with their associated labels that can be accessed by the BERT model during training.

4.5 Model modifications and enhancements

The BERT model will undergo various changes and addons to fine tune it to perform sentiment analysis in social media. these changes include additional model pre-training for BERT, expansion of vocabulary for both the BERT model and the BERT tokenizer as well as the implementation of hidden layer aggregation, which based on key findings and conclusions of the literature review, will benefit the performance of the BERT model. The techniques mentioned will explained in the sections below.

Model pre-training

The model pre-training will help enhance BERT's understanding of the language used in the domain of social media. This process will make use of the pre-training dataset defined in section

4.2 where the dataset was split into multiple parts and have undergone the tokenization process. This pre-training process will use Masked Language Modeling, the same technique used to pre-train BERT by using data from BooksCorpus and the English Wikipedia. A defined percentage of tokens from the input data will be randomly masked (e.g. 15%) during the training phase, the model will then be required to predict these masked tokens based on the context of the input text, allowing the model to learn the contextual embeddings for the social media domain. Next Sentence Prediction will be excluded based on the key insights from the literature review where this technique could add unnecessary complexity of the model and affect performance.

With this technique, the model will be able to better adapt to the unique language structure that many social media platforms have such as slang, emojis and informal language. The implementation of model pre-training will be able to complement the expansion of the vocabulary layer of BERT to aid it in performing better when training on Masked Language Modeling.

Vocabulary expansion

The vocabulary of BERT and BERT tokenizer will be expanded with the domain specific language used in social media and online platforms, this vocabulary will include emojis and slang. To identify what vocabulary will be inserted into BERT, term frequency–inverse document frequency (TFIDF) will be used on the dataset to find to find the most significant terms and emojis. After the dataset has been processed, the results will be filtered with the help of regex and a slang dictionary to obtain the words slangs as well as the emojis.

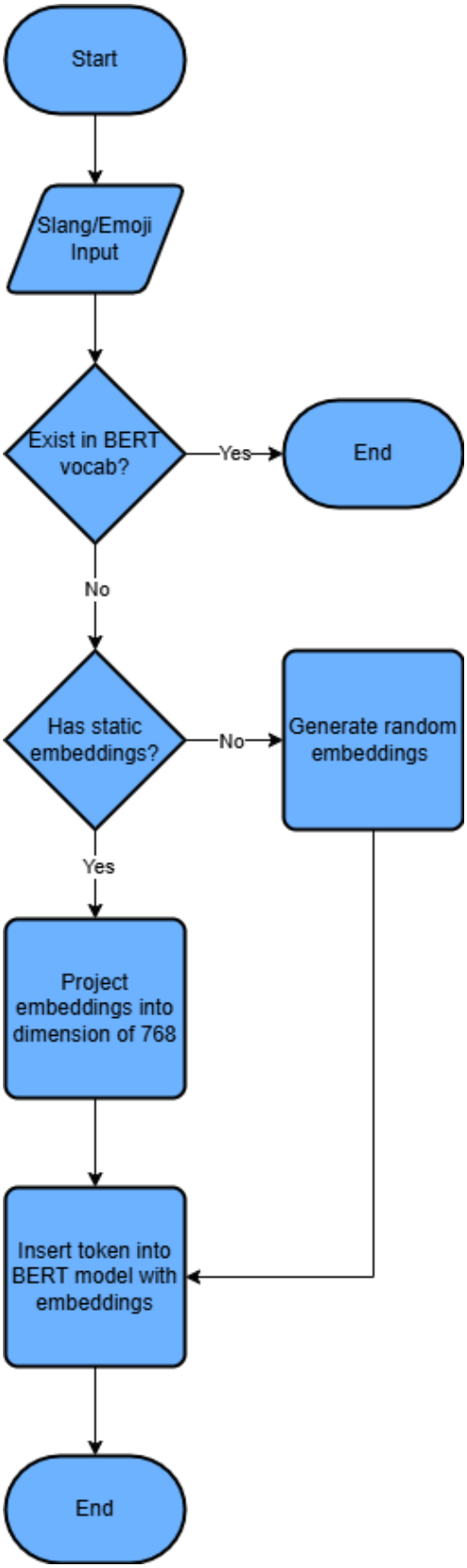


Figure 4: Flowchart of vocabulary insertion process

Before inserting the obtained vocabulary into the BERT vocabulary layer, the obtained vocabulary is compared against the BERT vocabulary layer to check if any of the vocabulary is existing inside BERT. If the vocabulary does not reside within BERT, it can be inserted into the vocabulary layer of BERT. Word embeddings can be initialized for the vocabulary using embeddings from a static word embedder like GloVe which contains pre-trained embeddings for a large vocabulary, including slang from social media platforms like Twitter. For emojis, the same can be applied by referring to a static word embedder like emoji2vec. This can provide the model better semantic awareness for the start, faster convergence and help reduce computational time. To perform this process, the vocabulary can be compared against the static word embedder to check if there are pre-trained embeddings present for this vocabulary, otherwise, the weights for the embeddings can be randomly initialized.

One issue that needs to be dealt with is the matching of dimensionality of the embeddings coming from a static word embedder. GloVe for example, has a dimension shape of 300 while the word embeddings of BERT have a dimensionality of 768. To solve this issue, a linear projection layer can be used to transform the 300-dimension GloVe embeddings into a dimensionality of 768 that can be accepted by BERT. However, as it is a one-time transformation for the word embeddings, random weights will be used for the linear projection layer and the output obtained may not capture the original semantic information provided by GloVe. Despite this drawback, it still provides the model a starting point with meaningful initializations instead of pure random embeddings. Once initialized, the vocabulary has been successfully inserted into the BERT vocabulary layer and is ready for further training.

Hidden layer aggregation

WHLA, the weighted hidden layer aggregation involves the weighted summation of the hidden layers, a variant from the HLA implemented in studies from the reviewed literature that involved a hierarchical architecture. It will be used to improve the model's ability to capture nuanced semantic representations from social media text and online platforms. This process will involve aggregating the outputs from multiple hidden layers within BERT to create a more comprehensive representation of the given text. The last four hidden layers of BERT will be selected to perform the aggregation as based on reviewed literature, helps to capture important semantic representations in text and provided the most improvements to performance.

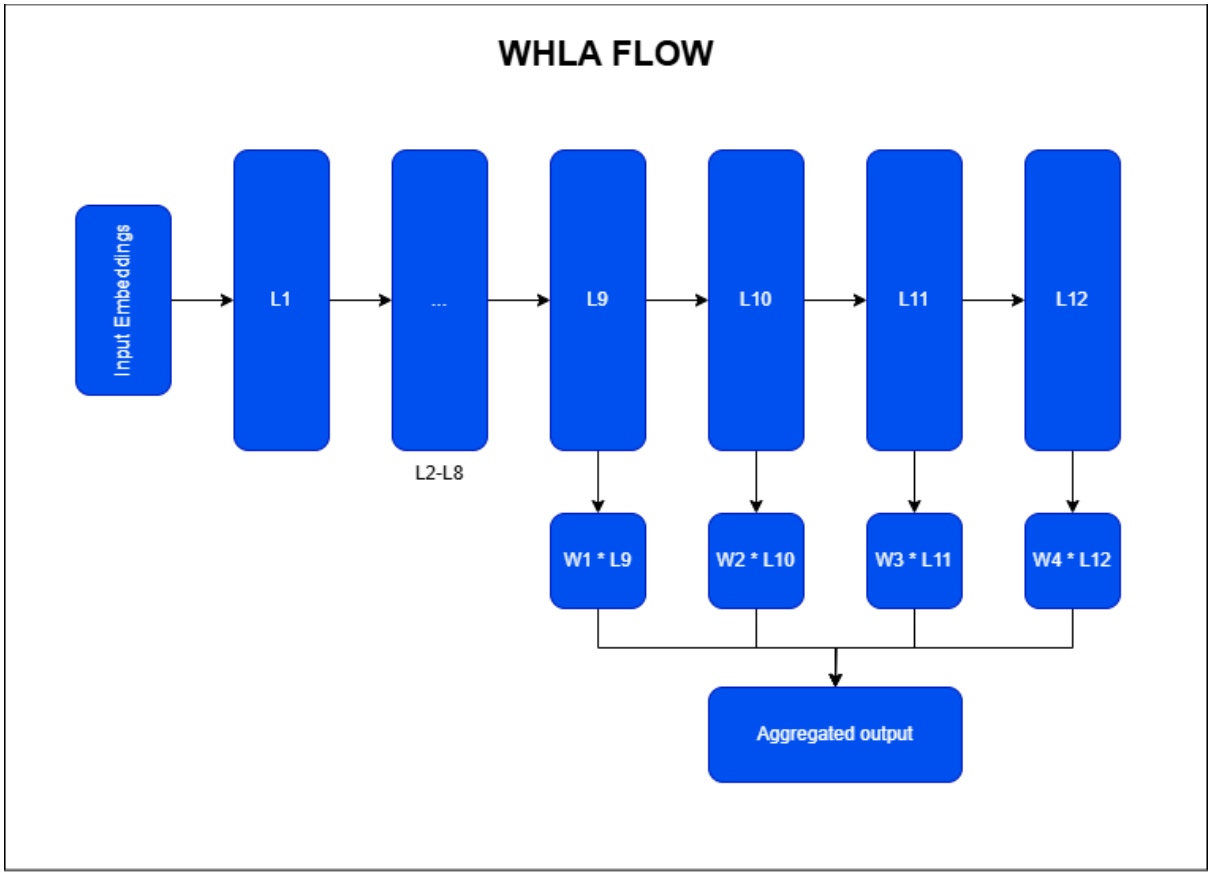


Figure 5: Weighted Sum Aggregation Across BERT's Final Layers

The figure above represents the process execution of WHLA within BERT where L_{12} to L_1 represents the hidden layers of BERT. Each hidden layer will produce a hidden state which will be passed into the next layer until it reaches the final layer. By default, without any form of aggregation, the final hidden state will be passed into a fully connected layer to perform classification. However, in this case, the model will perform a weighted summation on the last four layers of BERT to obtain the aggregated output. Proposed weights will be introduced to allow the model to learn the importance of the each of the last 4 hidden layers, enhancing the model's ability to make better use of the semantic representations captured by BERT. The implementation is expected to provide a more balanced integration of features from the different layers in BERT, allowing it to emphasize the more relevant features to the task at hand at the slight cost of additional computational cost. The aggregation of the weighted sum aggregation is represented as follows:

$$A_{final} = f(w_1 \times L_9, w_2 \times L_{10}, w_3 \times L_{11}, w_4 \times L_{12})$$

Where A_{final} represents the final output from hidden layer aggregation process and w_1 to w_4 represents the proposed weights that is assigned to each hidden layer. $f()$ represents the

summation function for all the weighted hidden layer outputs.

4.6 Model training and hyper tuning

This section will involve fine-tuning the model on the sentiment analysis downstream task. The training data defined in section 4.2 will be fed into the BERT model with defined hyperparameter settings. These settings will affect how the model will train using the training data, it involves settings such as:

- Learning Rate
    - Adjust the learning rate of the weights in the model that will affect how quickly or slowly the model converges during training
- Batch size
    - Defines the number of samples that the model will process before updating its weights
- Epochs
    - Defines the number of times the model will process through an entire dataset during training
- Dropout Rate
    - Adjust the probability of dropping nodes during training which can help reduce overfitting
- Optimizer
    - Defines the optimizer to be used during the training of the model, determining how the model's parameters are updated to minimize the loss function.

Defining optimal hyperparameter settings can greatly influence the performance of the model after training is complete and thus, careful tuning is required to ensure the model performance does not degrade. To prepare for the model evaluation section and to provide a fair evaluation of all the different techniques implemented on the model, a series of models will be trained:

- BERT without modifications (This model will act as the baseline model)
- BERT pre-trained on pre-training dataset
- BERT with expanded vocabulary
- BERT with hidden layer aggregation
- BERT with all 3 techniques implemented

4.7 Model evaluation

The model evaluation involves a performance evaluation where different metrics are used, these metrics include and are not limited to, accuracy, precision, recall and the F1 score. After training has been completed in section 4.6, the performance evaluation will be conducted against the test dataset to obtain the evaluation results. These results will be used in the results and discussion section. In addition, as mentioned in the section 4.4, the evaluations scores from existing literature can be used to benchmark against the models that were trained in this project to gain insights and ensure competitiveness.

4.8 Web Application

A 2-tier web application will be created to provide a platform for interactions with the model. The 2-tier web application will consist of a frontend, which is responsible for rendering the user interface elements for users while the backend will be responsible for handling the REST API framework along with integration with the different models. For the frontend, ReactJS with Typescript was chosen as the framework, providing a fast and efficient way to build the user interface. For the backend, Django with the Django rest framework was chosen as the framework to help build the APIs and integrate with the models. The frontend and backend will interact through various API calls, such as performing a POST request with a text user input and receiving the appropriate sentiment in the response body.

5. Results and Analysis

5.1 Overview

The results section will contain the process of the implementation of the methodology section as well as data obtained from the metrics computed from the model after training has been completed. These metrics, as stated in the previous section, will contain important information on the performance of each model. A series of tables will be presented, showcasing the model performance against each other as well as against different task during the training pipeline.

5.2 Data Preparation

To meet the requirements as stated in the methodology section for data sourcing and collection, the dataset will require data collated from different sources. Several datasets from various

sources were identified to help prepare the final dataset to be used for binary sentiment analysis training. These were the datasets identified as well as their metadata:

- Sentiment140 Twitter dataset
    - A Twitter dataset consisting of 1.6 million tweets that were extracted using the use of Twitter API. There are 2 different polarities included in the training dataset. Each tweet is labelled with a polarity indicating a positive or negative sentiment. A number is used to represent each sentiment, 0 represents negative while 4 represents positive. There is an equal distribution of both sentiments, numbering 800k records each. The labels for each tweet were created by assuming that any tweet with a positive or negative emoji/emoticon was regarded as a positive/negative tweet respectively [46].
- Reddit India General Elections Dataset
    - This dataset consisted of reddit comments that were extracted using the help of the Reddit API. These comments included people's opinions on the next prime minister of India during the India General Elections held in 2019. The dataset has a total of 37k comments covering 3 different sentiments ranging from positive, negative and neutral. Each sentiment was represented with a number, 0 representing a neutral comment, -1 representing a negative comment and lastly, 1 representing a positive comment. Each comment in the dataset was labelled with a sentiment label. There is a total of 15380 positive comments, 13142 neutral comments and 8277 negative comments. There was no indication in the process of how each comment was labelled [47].
- Reddit GoEmotions Dataset
    - The GoEmotions Dataset is a reddit dataset that consists of 58009 comments that were manually annotated by humans to an emotion range covering 28 different emotions. Each comment is labelled with 1 or more emotion, representing how a sentence could indicate 1 or more sentiments. Additionally, a training, test and validation split was provided by the author for this dataset [48].
- Instagram Covid-19 Dataset
    - The Instagram dataset consisted of a multilingual dataset with 500k records of text obtained from Instagram post over the topic of COVID-19. These records were obtained over a 5 year range from early 2020 to late 2024. There are 3 different sentiments that were covered in this dataset, positive, negative and neutral. Each post was labelled with the use of sentiment analysis tools such

as VADER and twitter-xlm-roberta-base-sentiment [49].

- IMDB movie reviews dataset
    - The dataset consists of 50k move reviews that were collected from the move review website forum IMDB, where individuals rate and provide reviews of various shows and movies that are trending. The dataset includes 50k labelled records between a positive and negative sentiment. Based on the author notes, the sentiment of the review is done based on the rating score of the movie review, a positive sentiment is provided to a review score of more than equal to 7, while a negative sentiment is provided to a review score of lower than equals to 4. Reviews with scores 5 and 6 are ignored to account for neutral ratings [50].
- YELP open dataset
    - The YELP open dataset is a dataset provided for academic use, it contains text records in the form of reviews about various businesses, pictures etc. The dataset is provided in a JSON format, including the number of review stars they provided the entity they are reviewing along with the text of the review itself. Other metric data about the review is also provided such as the user id, business id and date of the review [51].

Each data set was preprocessed and converted into a data frame with 3 columns, the polarity, the text as well as the data source it came from. During the preprocessing process, functions such as sampling are used and to ensure reproducibility of results, all preprocessing functions that use a randomizer will have their seed set to 69. To prepare the data frame, each dataset was cleaned to produce a standardized output as follows:

- Eliminate any empty or null text records
- Only 2 sentiments will be accepted, either positive or negative labelled
- For each text, there should only be 1 sentiment attached
- Each text record should be in the English language
- Each text will be cleaned with regex-based requirements as stated in the methodology section

Specific datasets like the GoEmotions dataset and the YELP reviews dataset require more specialized tuning as their sentiment labels were not explicitly shown to include positive and negative sentiments. The GoEmotions dataset for example, required the identification of positive and negative sentiments among the 28 emotions that were provided and after which, were grouped and labelled accordingly. The YELP reviews dataset on the other hand, did not

contain a label for a sentiment, instead, only included the review rating on each record. Thus, to obtain the positive and negative sentiment accordingly, the 5-star reviews were regarded as positive sentiments while the 1-star review were regarded as negative sentiments. Added together with the sheer size of the dataset, only a sample of these 5-star and 1-star reviews were obtained to be used in the combined dataset.

Each dataset was then combined to form the dataset to be used for training and evaluation purposes. The results and distribution of the collated dataset is as follows:

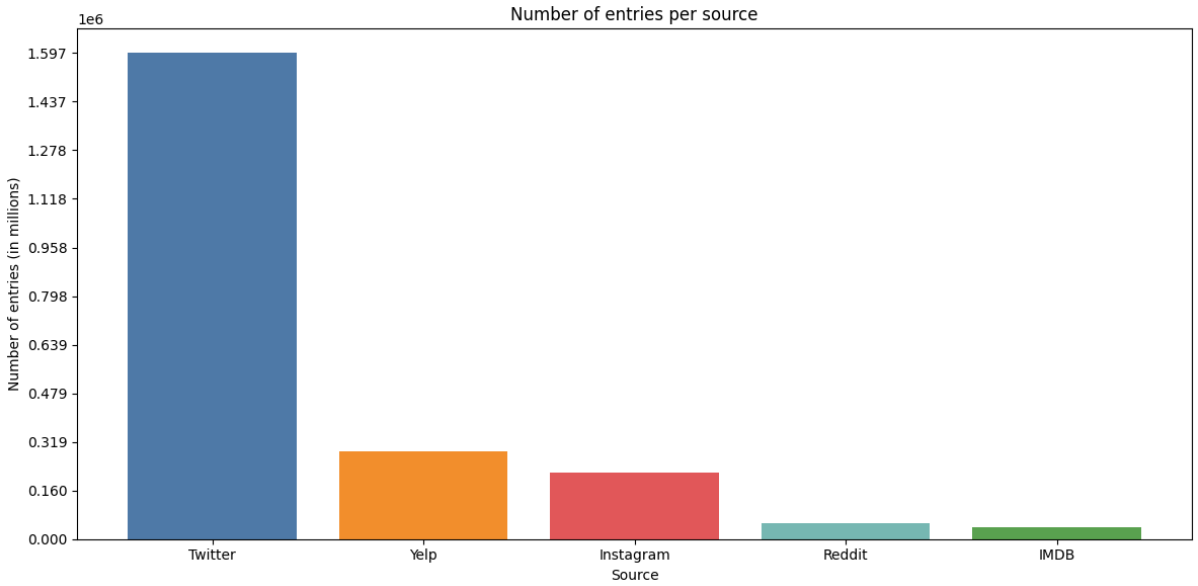


Figure 6: Bar Chart of Sourced Data Distribution

- Twitter source: 1596886
- YELP: 287150
- Instagram: 219578
- Reddit: 53142
- IMDB: 40065

The data source from Twitter covers roughly 70% of the entire combined dataset while the remaining 30% of the dataset was spread across the remaining data sources, ranging from 40k to 300k data records.

Next, the dataset was analyzed for the token limit of each record. The BERT model can only receive an input of up to a maximum amount of 512 tokens, any text record that has a token number that exceeds the 512 limits would lose its information when undergoing the tokenization

process through truncation. The tokenization process is used here to identify the token length of each text record. The tokenizer used in this process is the tokenizer of the BERT-base-uncased model with its default vocabulary and settings. The results of the process indicate the following for the dataset:

Table 1: Distribution of number of tokens per text record with default tokenizer

Tokens	Count
> 0	2221566
> 100	248629
> 200	101724
> 300	47839
> 400	24745
> 500	13634

The findings indicate a need for the removal of several text records that exceeded the max token limit. Using this opportunity, additional records exceeding a specified amount will be removed from the dataset to consider the eventual expansion of tokenizer vocabulary that would lead to an increase in token length. Examining the dataset, the number of tokens that exceeded the token number of 400 accounts to roughly 1% of the entire dataset, thus the removal of such entries would have a minimal impact on the overall dataset while ensuring that all remaining records fit within the BERT model's token limit. This approach also accounts for potential vocabulary expansion, preventing unintended truncation of text information during tokenization.

Afterwards, the dataset is analyzed for their distribution of positive sentiment and negative sentiment labels. For a model to be fine-tuned effectively, it is important that there is a balanced distribution of labels in the dataset. Uneven distributions could lead to situations where the model learns to favor a specific class, causing a reduction in model performance and its ability to generalize.

Table 2: Sentiment Distribution Across Data Sources

Source	Positive	Negative
Twitter	798446	798440
Reddit	34691	18451
Instagram	169249	50329
IMDB reviews	19935	20130
Yelp reviews	147007	140143

The table above shows the current distribution of sentiment labels for each source. The Instagram and Reddit sources for example show a considerable difference between both sentiment labels. To ensure balance in labels in the dataset used during fine-tuning, a combination of a proposed algorithm and splitting of datasets was used, explained in the following section.

The combined dataset will be used for all training performed during this project; however, not all the data is required to perform either additional pre-training or fine-tuning. Thus, a split of the combined dataset is required, assigning each dataset for their specific purpose. This split opens an opportunity to balance the sentiment labels during fine-tuning. While the fine-tuning process requires a balance in sentiment labels, the additional pre-training does not. This is because masked language modeling is an unsupervised machine learning task which focuses on allowing the model to learn contextual representations of language by predicting masked tokens within a given text, rather than classifying sentiment, thus, the sentiment labels of the pre-training dataset can be ignored.

Along with the aim to balance the sentiment label distribution, the original ratio of source distribution is also aimed to be maintained. For the algorithm, a specified difference threshold and a 70:30 data split was defined for the pre-training and fine-tuning dataset respectively for each source. The threshold value was defined to specifically target the Reddit and Instagram sources, equating to a value of 7000. Sources whose difference value between sentiment labels has crossed the threshold will ignore the 70:30 data split and instead assign the difference in value to the pre-training dataset, while the balanced ratio will be assigned to the fine-tuning dataset. The pseudo code for the data split algorithm can be found in the appendix section Appendix B Figure 14. The result of the data distribution for each dataset after the above preprocessing is as follows:

Pre-training dataset

Table 3: Sentiment Distribution for Pre-Training Dataset

Source	Positive	Negative	Total	Positive Percentage	Negative percentage
Twitter	558912	558908	1117820	50.000179	49.999821
Reddit	16240	0	16240	100	0
Instagram	118920	0	118920	100	0
IMDB reviews	13954	14091	28045	49.755750	50.244250
Yelp reviews	102905	98100	201005	51.195244	48.804756

Total pre-training dataset records: 1482030

Fine-tuning dataset

Table 4: Sentiment Distribution for Fine-Tuning Dataset

Source	Positive	Negative	Total	Positive Percentage	Negative percentage
Twitter	239534	239532	479066	50.000209	49.999791
Reddit	18451	18451	36902	50	50
Instagram	50329	50329	100658	50	50
IMDB reviews	5981	6039	12020	49.758735	50.241265
Yelp reviews	44102	42043	86145	51.195078	48.804922

Total fine-tuning dataset records: 714791

Each dataset is brought before each pipeline respectively to be used for training. To ensure maximum volume and variability for the masked language modeling process, the pre-trained dataset is used as is when injected into the pipeline. However, the fine-tuning dataset will undergo 1 additional change to address the issue of data source distribution. Out of the 714k records, roughly 70% of the data is still dominated by the Twitter data source. Thus, to bring about a more balanced ratio, only a fraction of the Twitter data source is used for training. The amount of Twitter data that was extracted is 30%, resulting in a value of 140k Twitter data source remaining in the fine-tuning dataset. This helped balance the ratio between each data source, which could potentially lead to inherent biases of those specific data sources if left unaddressed. In addition, the extracted Twitter data will be used later on to create another set to test the generalization capabilities of the model.

Aside from the above dataset created, there were other datasets sourced and created to help provide other methods of evaluation. These datasets were used to test the model on more complex tasks and benchmark it against other models in the industry to help provide insights on the generalization capabilities of the improvements suggested. The datasets identified along with their purpose were as follows:

- Twitter only dataset
    - Formed with 70% Twitter data extracted from the fine-tuning dataset in Table 4. This dataset was created to help perform an evaluation on the difference in generalization capabilities of the model when training on different data sources.
- SemEval2017 Task 4 dataset
    - This dataset was part of the SemEval 2017 event. The 4th Task of this event was to classify the overall sentiment of a Tweet between 3 different sentiments, positive, negative and neutral. 3 separate datasets were provided for this task, the training, evaluation and test dataset. The contents of the dataset did not go through any preprocessing steps. This dataset will be used to evaluate the model improvements and benchmark it against other state of the art models [52].
- Twitter Emotions Dataset (6 Label)
    - The Twitter Emotions Dataset is a dataset that consists of labelled text records of approximately 390k tweets. Each tweet is labelled across an emotion range of 6 emotions. The emotion map of each tweet is as follows, sadness – 0, joy – 1, love – 2, anger – 3, fear – 4 and lastly surprise – 5. There are no indications mentioned by the author on how the dataset was labelled and whether there was any preprocessing performed on the text. The credibility of the dataset is backed up by its engagement of views and downloads. This dataset will be used to test the performance of the model on a multi-class classification task [53].
- Single Labelled GoEmotions Dataset (28 Label)
    - This dataset is a modified version of the GoEmotions dataset sourced for the collated binary sentiment analysis dataset above. The dataset was modified to remove text records with multiple labels to allow for training of the model to perform single label classification on this multi-class classification task. This was made to further test the capabilities and effectiveness of the model and their improvements on even more complex tasks [48].

For the above datasets, preprocessing will be applied accordingly. The requirement of the preprocessing follows the rules identified earlier on in this section except for the number of labels that is tied to the task. In addition, when inspecting the SemEval 2017 dataset, Twitter emotions dataset and GoEmotions dataset, the max token length that can be found in the data records does not exceed 100, indicating that there are no long paragraphs present in these datasets.

For the SemEval 2017 dataset, Twitter emotions dataset and GoEmotions dataset, their label distributions were also recorded below:

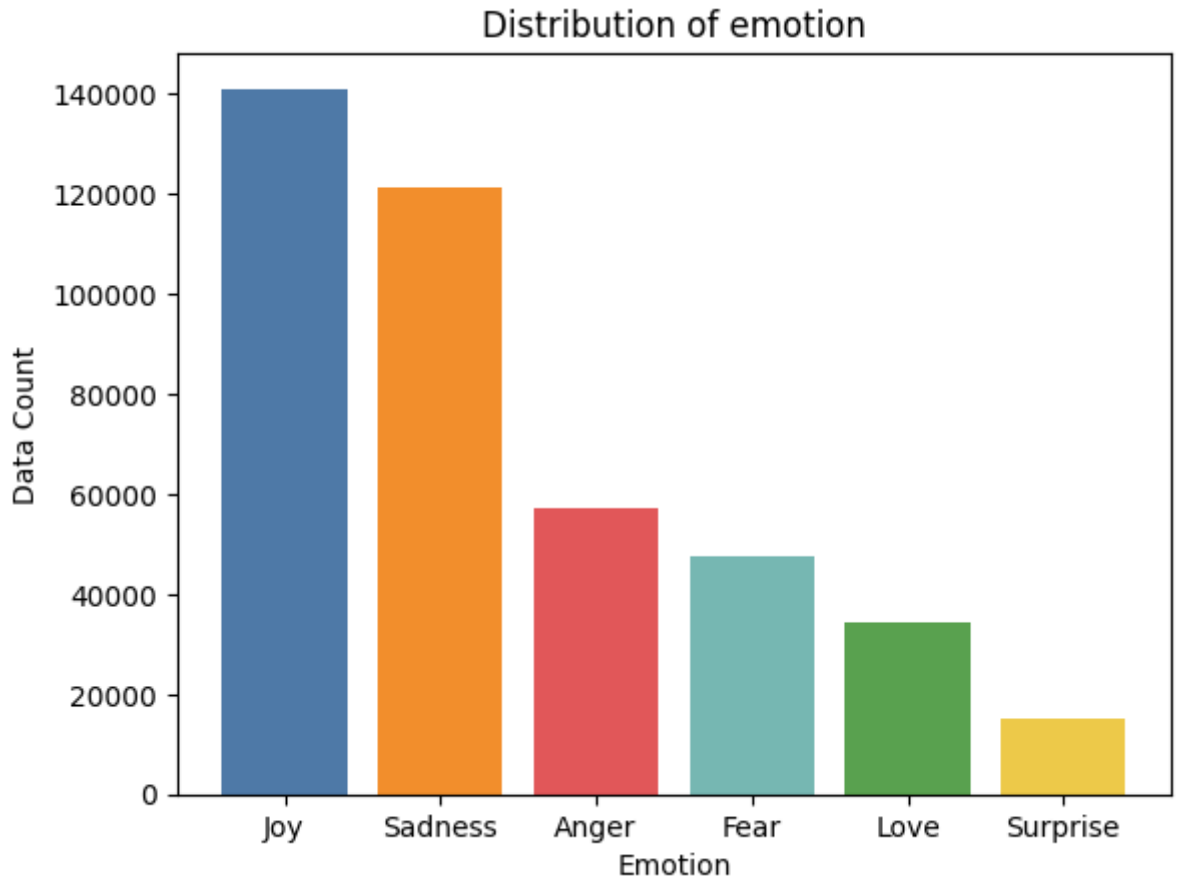


Figure 7: Bar Chart of Twitter Emotions (6 Labels) Distribution

- Joy: 141067
- Sadness: 121187
- Anger: 57317
- Fear: 47712
- Love: 34554
- Surprise: 14972

Based on the above bar chart data, for the Twitter emotions dataset, the data labels are shown to be uneven. Emotion labels such as ‘joy’ and ‘sadness’ dominate the dataset compared to ‘love’ and ‘surprise’ emotions.

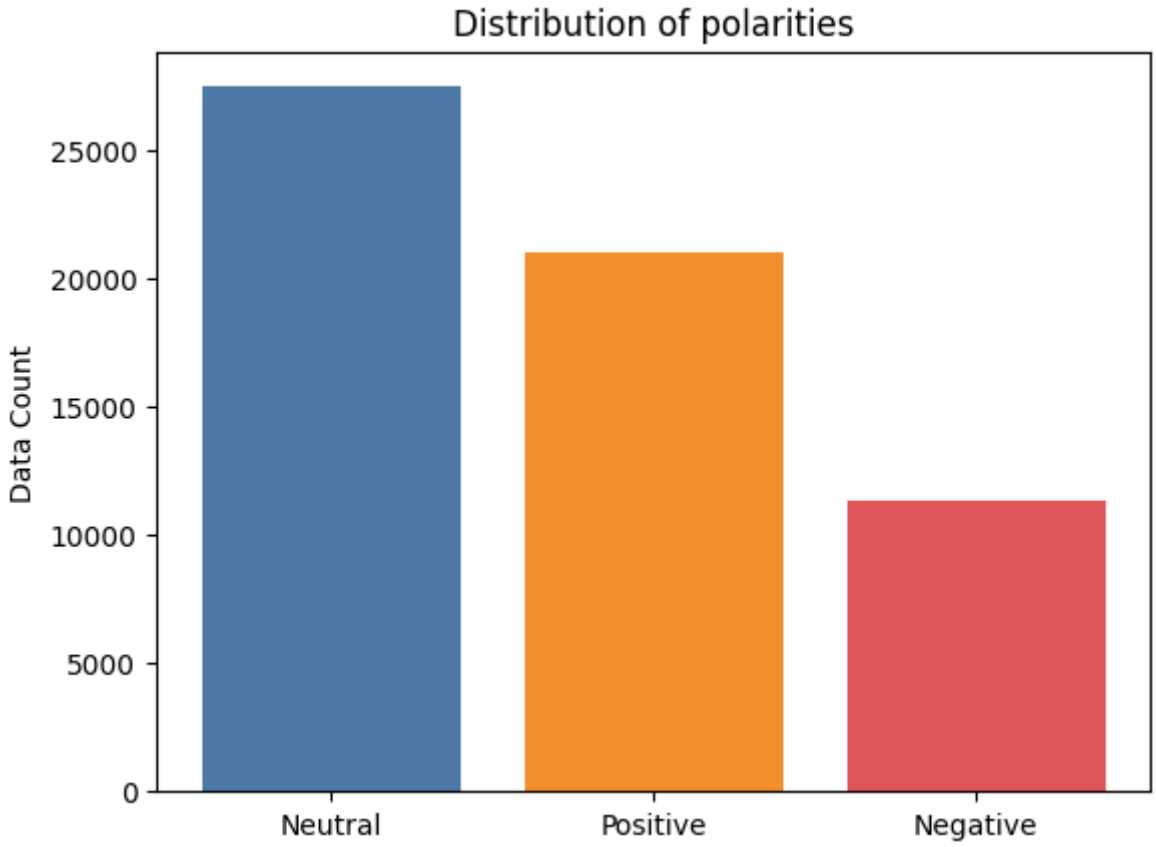


Figure 8: Bar Chart of SemEval 2017 Distribution

- Neutral: 27476
- Positive: 21043
- Negative: 11377

Based on the above bar chart data, for the SemEval 2017 dataset, the data labels show some level of imbalance with negative labels having lower data records compared to the neutral and positive records.

Lastly, the GoEmotions dataset was found to be extremely imbalanced, with a total record count of 45443, the neutral label took up approximately 16k records while the remaining 27 labels were spread around a data record count of 3k to a low count of 47.

5.3 Improvement Implementation

As mentioned in the methodology, 3 main improvements, hidden layer aggregation, additional domain pre-training as well as vocabulary expansion. These improvements will be applied individually to the base Bert model, fine-tuned and evaluated. Afterwards, these 3 improvements will be combined on top of the Bert model and evaluated as well.

Hidden Layer Aggregation

The hidden layer aggregation improvement implements a wrapper around the loaded BERT model. This wrapper extracts the last four hidden layers that are output by BERT after feeding it the text inputs. Afterwards, a weighted aggregation is performed on the last 4 layers with initialized gates. The gates represent learnable weights that the model will use during the forward pass to help determine the contribution each hidden layer has towards the final representation. To help improve the stability of the wrapper, a normalization layer was implemented on the aggregated output before extracting the CLS token representation. This token is then passed through a dropout layer to further prevent overfitting. Lastly, it is passed through to the fully connected layer to generate the logits needed for classification and loss calculation for the model to learn. The implementation pseudo code for the wrapper class can be found in the appendix section Appendix B Figure 15.

Additional domain pre-training

The domain pre-training was performed using the mask language model approach. This was implemented by loading the BERT model to be pre-trained along with a data collator for mask language modelling, where the percentage of text to mask out in each document was specified to a value of 15%.

Vocabulary expansion

The implementation of the vocabulary expansion is much more complex compared to the earlier 2 improvements. The implementation is split into 2 sections, the first section involves the identification and preprocessing of slang and emoji to be added. The second section is the insertion of the identified slang and emoji into the model and tokenizer.

1. Slang and emoji identification and preprocessing

To prepare for the vocabulary expansion of the model and tokenizer, a reference for slang and emojis is needed to successfully identify those terms from the combined dataset. Thus, a slang dataset was sourced, containing a list of slang used by individuals that belong to generation Z [54]. The dataset, publicly available on hugging face was created by sourcing public websites and social media platforms as stated by the author. The dataset contains 4 columns, the slang, description, example of usage and context of the slang. The slang dataset underwent preprocessing to eliminate the use of multiple words and numbers, however, the result of the preprocessing creates dirty slang data that has lost their meaning which would need to be removed. To ease the process of identifying those dirty data, the process of removal will be

performed after the TFIDF score has been calculated, the actual process explained in the section below. As for the identification of emojis, the standard python emoji library was used for identification.

To identify the most used terms in the dataset, the initial plan as stated in the methodology was to perform TFIDF on the entire dataset, producing a list of words which will then be further processed, however, the output resulted in too many words that cannot be processed by memory and thus a different approach was used. During the TFIDF process, tokenization was implemented to help identify each token from a document before calculating the TFIDF. The tokens that were extracted are then compared to the slang dataset prepared, before adding to the results, this method allowed for a cleaner output and lower memory usage.

The TFIDF score for the slang words and emojis were calculated separately to help simplify the pipeline. The first dataset to undergo the TFIDF calculation was the slang words, with the results producing a csv file with the slang words and scores.

At this point, the result output contains dirty data from the earlier preprocessing process and in addition, not all slang from the slang dataset was used correctly with the right context. These factors mean that the results output cannot be used as is, and manual preprocessing will need to be performed to ensure the results list is cleaned. To solve this issue, the results list, containing a list of 932 slang words was first filtered against the TFIDF value. The value of TFIDF values contains a value range as shown below:

Table 5: Distribution of TFIDF value counts for Slang

TFIDF value (Slang)	Count
> 0	932
> 50	428
> 100	353
> 1000	108
> 10000	15

The slang words that obtained a TFIDF score of above 50 covered approximately 50% of the slang words list. The top 50% of words are extracted and filtered manually by manually inspecting the list of slang words. An initial wave of words was removed by analyzing words found that appear gibberish or single alphabetical letters due to the result of the earlier slang list preprocessing. An example of this would include repeating letters such as ‘mm’ or singular letters such as ‘a’ or ‘b’. After this filtering, another check was performed on the word list by

checking that the several examples of the words being applied in sentences in the combined dataset aligns with the context description as stated in the slang dataset. Words that do not align are deemed unusable as it would contribute noise instead of useful information when it comes to performing the sentiment classification. After the manual cleaning, the number of remaining words to be added amounted to a value of 195 words and with this, the slang words list to be added is completed.

The same tokenization and filtering process was performed for the emojis to be added, producing a csv output containing all emojis used in the combined dataset. A total of 1730 emojis were identified from the dataset, with TFIDF values containing a value range as shown below:

Table 6: Distribution of TFIDF value counts for Emojis

TFIDF value (Emoji)	Count
> 0	1730
> 50	348
> 100	228
> 1000	16
> 10000	0

The number of emojis that obtained a TFIDF score of above 100 amounted to 228, roughly 13% of the entire list. To simplify the selection process, the top 200 emojis were selected to be part of the emojis to be added to the model and tokenizer.

2. Slang and emoji insertion

The implementation of the slang and emoji insertion follows closely to the proposed action as mentioned in the methodology. The GloVe and emoji2vec static embeddings were sourced and prepared for weight initialization. The BERT tokenizer was loaded first, then, each slang and emoji are added as a new token only if it does not already exist in the tokenizer vocabulary. Afterwards, the BERT model is loaded, and the token embeddings of the BERT model is resized to the length of the tokenizer. A projection layer is created for the GloVe and emoji2vec static embeddings respectively. Each new token is then identified, and their weight embeddings is updated with the GloVe or emoji2vec static embeddings after projection. If the token is not found in the static embeddings available, the weight embeddings will be randomly initialized. With that, the expanded vocab model and tokenizer is prepared and available for training and evaluation. The pseudo code for the vocabulary expansion algorithm is included in the appendix section in Appendix B Figure 16.

With the expanded tokenizer, the is a possibility that it will tokenize extra tokens given a same document due to the added vocabulary. This was accounted previously in the data preprocessing section where documents with more than 400 tokens tokenized by the base BERT tokenizer was removed. Thus, expanded tokenizer used to tokenize the cleaned combined dataset to check if there are any documents that contain tokens that exceed the 512 max token limits. The results of the tokenization process are as follows:

Table 7: Distribution of number of tokens per text record with expanded tokenizer

Tokens	Count
> 0	2196821
> 100	239265
> 200	88155
> 300	29754
> 400	4563
> 500	3

The results indicate that the tokenizer has detected additional tokens in documents due to the expanded vocabulary as indicated by the presence of entries exceeding 400 tokens. As these entries are still within limit of the 512 token limit, they will not be removed.

5.4 Web Application Implementation

As part of the project requirement, a web application was created to house the various trained models across the different task to provide a platform for an easy interaction with the models as well as to provide some level of insight into the model behaviour during the classification process.

In accordance with the frameworks mentioned in the methodology section, ReactJS with Typescript and Django were used to build the frontend and backend of the web application respectively. 3 pages were built for the user interface, a homepage, dashboard as well as an API documentation page. The home page serves as the landing page for the web application, where the user can choose to proceed to the dashboard or the API documentation page.

The dashboard contains a simple user interface that allows a user to select a specific model through a dropdown menu. There is also a text box that accepts a user input that when submitted, will perform an API call to the Django REST API framework to query the sentiment

of the user input from the model. The API returns the sentiment of the user input, along with the confidence of the prediction and the individual word contribution towards the sentiment. The word contribution towards the sentiment is calculated with a gradient-based attribution by performing a backwards pass to compute the gradients of the predicted class logits, which represents the model raw outputs, with respect to the input embeddings. This process essentially provides an idea on how much a small change in the embedding of a specific token will affect the predicted class. For this process, special tokens like the [CLS] and [SEP] tokens are ignored to focus on actual tokens of the text record. The token contribution will also be normalized to consider the missing special tokens.

The user interface also provides a button to upload a CSV file that when filled with a column of 'text' and 'label', can help to provide multiple instances of predictions at once. The output will generate a pie chart indicating the distribution of predicted classes, along with a table showcasing all the text records in the csv, along with the predicted class and the confidence of the prediction.

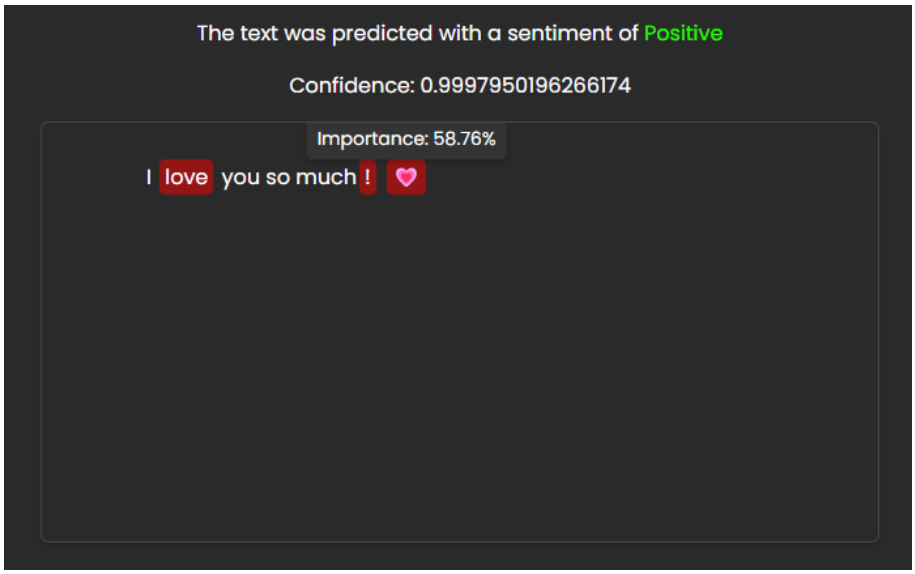


Figure 9: Binary sentiment prediction on user interface

An evaluation mode is implemented that can be toggled when uploading the csv file. With the evaluation mode enabled, the user can upload a labelled dataset to evaluate the model on the spot to test the accuracy of the model predictions. The dashboard will then show the table with an additional column of the actual label vs the predicted label performed by the model.

The API documentation page is the second page available in the web application that showcases the available API from the Django REST API framework along with the request and

response structure. The user can directly query the API endpoint with the appropriate request body to interact with the model. Screenshots of the web application UI can be found in the appendix section Appendix C.

5.5 Training Process

The training process took place on the cloud based Jupyter Notebook environment by Google. Colab premium was used to access higher quality GPUs such as the Nvidia A100 which is specialized for training machine learning and AI models. The A100 GPU provided by the Colab premium subscription provided significantly faster training times over traditional CPUs and helped accelerate the training and testing phases for the models [55].

The hugging face API trainer was also used to help simplify the training process for some of the models used. The trainer helped to provide easy access to the BERT models as well as the saving and loading of trained models. It also helped simplify the process of the definition of training arguments used for the training process and the viewing of training results.

Training Parameters

During the initial training phases of the models, the base BERT model was used to set a base pipeline for the upcoming models to be trained. With this base pipeline, various settings were used to trial and error the training process to find a standardized procedure to efficiently train our models to identify the effectiveness of these improvements. After various testing, the hyperparameter settings to be standardized and used in the pipelines were set as follows:

- Learning Rate: 3e-5
- Batch Size: 16
- Epochs: Dependent on task and experiment conducted
- Drop/weight decay: 0.01
- Warmup steps: 0.1 * total_steps, where ‘total_steps’ is calculated by (len(train_dataset) // batch_size) * epochs
- Optimizer: AdamW

These settings were obtained based on the trainer default settings, trial and testing as well as based on existing literature. The learning rate 3e-5 was chosen from the middle ground of existing identified best fine-tuning learning rates along with a weight decay of 0.01 to help prevent overfitting [56]. The batch size of 16 was arbitrary chosen, as while other BERT papers

used more powerful GPUs to perform training, a higher batch size of 32 was chosen while the resource constrained environment of this project required a lower batch size number. For training stability and improved performance, 10% of the total number of training steps was set to create a gradual increase in learning rate. Lastly, the default optimizer AdamW for transformer-based models was used for the updating of weights and biases.

Additional settings

To cater to the fact that BERT will require all inputs in a training batch to have the same sequence length, both pipelines made use of a data collator with padding. This data collator helped to enable dynamic padding per training batch, padding each input to the data entry with the longest sequence length, eliminating the need for unnecessary padding tokens, which in turn decreases computational overhead and memory usage.

5.6 Results

The results section will present the evaluation results of the various models mentioned above. The evaluation results are split based on the various tasks each model is trained upon. Each task is evaluated against a set of predefined evaluation metrics as mentioned in the methodology; the tasks include:

- Masked Language Modeling for domain pre-training
- Binary sentiment analysis (Positive and Negative) for improvement effectiveness check
- Binary sentiment analysis (Positive and Negative) for generalization check
- Ternary sentiment analysis (Positive, Negative and Neutral) for benchmark check
- Multi class sentiment analysis (6 Emotions class) for general performance check
- Multi class sentiment analysis (28 Emotions class) for general performance check

In addition, based on the methodology section, 5 variants of the BERT model will be trained on the various tasks, names will be provided to each of these 5 models to provide better identification when reviewing the results and discussions section. The 5 models that will be trained are:

- Base model – BERT model without modifications
- Pretrained model – BERT model pre-trained on pre-training dataset
- Expanded model – BERT model with expanded vocabulary

- WHLA model – BERT Model with hidden layer aggregation
- Proposed model – BERT Model with all 3 techniques implemented

For each task aside from the MLM for domain pre-training, to account for the variability in training and updating of model weights, 3 training sessions were held. Based on training results, due to low variability and similar trends for the 3 training sessions, the best results out of the 3 sessions were picked for evaluation in the results displayed for the relevant training tasks.

In addition, it is to be noted that except for the emotion sentiment analysis training pipelines, training sessions for multiple epochs were split into 2, the first session was to evaluate the performance of the model, and the second session was to test for signs of overfitting. As such, due to the total number of epochs for each sessions differing, the warmup steps under the training parameters were affected.

Masked language modeling pre-training

For the MLM task, the pre-training dataset was used to perform the training. The dataset was split into a training and evaluation set based on a 90:10 split. For both the Base Model and Expanded Model, the dataset was tokenized with the default and expanded tokenizer respectively. Both models were trained across multiple epochs to identify if the model can learn the contextual representations of language. Each epoch took approximately 3 hours to train. The training and validations loss after each epoch is listed in the table below.

Table 8: Evaluation results for MLM pre-training

Epoch	Base Model		Expanded Model	
	Training Loss	Validation Loss	Training Loss	Validation Loss
1	2.359	2.250	2.372	2.275
2	2.204	2.138	2.179	2.085
3	2.146	2.058	2.068	1.986
4	2.067	2.012	2.007	1.927
5	-	-	1.987	1.920
6	-	-	1.944	1.888

From the results of the training, both models were able to learn well, as seen from the decrease

in training and validation loss over epochs. The Expanded Model was shown to obtain a higher loss value after the first epoch but managed to continuously outperform the Base Model after the second epoch. The Expanded Model was also shown to learn better than the Base Model, as seen from the higher loss reduction per epoch. For e.g. the reduction of validation loss from the first to second epoch for the Base Model is approximately 0.1, compared to the Expanded Model which is approximately 0.19. Overall, the Expanded Model has shown to outperform the Base Model in learning the contextual representations of language in the domain of social media.

Binary Sentiment Analysis improvement effectiveness check

The effectiveness of the improvements was analyzed through the binary sentiment analysis task. The fine-tuning dataset was used to perform the training process. The fine-tuning dataset was split into a training and evaluation set based on an 80:20 split. For all models except the Expanded Model, the dataset was tokenized with the default BERT tokenizer. For the Expanded Model, the dataset was tokenized with the expanded tokenizer. All models were trained across multiple epochs to analyze the learning trends of the models. Each epoch took approximately 1 hour to train. The results include the various performance metrics along with the training loss and validation loss per epoch.

Fine-tuning results

Table 9: Evaluation results for binary sentiment analysis improvement check

Model	Epoch	Training Loss	Validation Loss	Accuracy	F1 Score	Precision	Recall
Base Model	1	0.2178	0.2103	0.9149	0.9146	0.9226	0.9067
	2	0.2264	0.2167	0.9115	0.9098	0.9322	0.8883
	3	0.1328	0.3010	0.9162	0.9162	0.9206	0.9118
WHLA Model	1	0.2205	0.2102	0.9143	0.9140	0.9218	0.9064
	2	0.2225	0.2182	0.9116	0.9096	0.9359	0.8847
	3	0.1319	0.3061	0.9150	0.9149	0.9200	0.9099
Pretrained Model	1	0.2068	0.2022	0.9184	0.9180	0.9274	0.9087
	2	0.2196	0.2093	0.9169	0.9163	0.9271	0.9058
	3	0.1258	0.3056	0.9179	0.9175	0.9268	0.9084
Expanded Model	1	0.2285	0.2180	0.9109	0.9107	0.9172	0.9043
	2	0.1080	0.3783	0.9110	0.9111	0.9140	0.9082
	3	0.0982	0.4174	0.9142	0.9138	0.9217	0.9060

WHLA layer weights

Table 10: Weight values for WHLA model in table 9

Layer	L9	L10	L11	L12
Weight	1.0003	0.9980	0.9987	0.9910

The overall results indicates that the proposed improvements do not have a significant impact, only slightly increasing and decreasing the performance of the Base Model. The most noticeable change in performance metrics can be found in the training and validation loss of the models, compared to the accuracy and performance metrics. All the models show signs of overfitting after the second epoch, as seen from the continuous decrease and increase in training and validation loss till the third epoch. The WHLA model obtained identical results to the Base Model across all the performance results, showing no effect to the model performance. The weights assigned to the final 4 hidden layers for aggregation showed minimal changes as well. The Pretrained Model, obtained from the Base Model that underwent the MLM pre-training in the previous task, obtained the best performance out of all the models, showing a marginal decrease in training and validation loss compared to the Base Model, and a slight increase in accuracy and other performance metrics. The Expanded Model obtained the worst performance out of all the models, obtaining a higher training and validation loss values compared to the Base Model, and a slight decrease in accuracy and other performance metrics.

Binary Sentiment Analysis for generalization check

The generalization capabilities of the model were analyzed by training the models on specific data sources. The Base BERT model was used to perform the experiment, and 2 models were trained with 2 different sets of data sources. The Twitter only dataset and fine-tuning dataset were used to train the models. The fine-tuning dataset was split into a training and evaluation set with an 80:20 split. The 80 split was used as a training set to train one of the models, while the Twitter only dataset was used to train the other model. Both models used the remaining 20 split test set and the binary sentiment analysis SemEval 2017 test set for evaluation. All data was tokenized using the Base BERT Model tokenizer.

Fine-tuning evaluation set

Table 11: Evaluation results for binary sentiment analysis generalization check

Data Source	Epoch	Training Loss	Validation Loss	Accuracy	F1 Score	Precision	Recall
Twitter Only	1	0.3506	0.4296	0.8089	0.8142	0.7959	0.8334
	2	0.2716	0.5027	0.8070	0.8139	0.7890	0.8404
	3	0.1933	0.7347	0.7982	0.8044	0.7836	0.8264
All	1	0.2178	0.2103	0.9149	0.9146	0.9226	0.9067
	2	0.2264	0.2167	0.9115	0.9098	0.9322	0.8883
	3	0.1328	0.3010	0.9162	0.9162	0.9206	0.9118

Binary SemEval 2017 test set

Table 12: Test results for binary sentiment analysis generalization check

Data Source	Test Loss	Test Accuracy	F1 Score	Precision	Recall
Twitter Only	0.6191	0.7029	0.6888	0.5664	0.8787
All	0.4452	0.8191	0.7836	0.7093	0.8754

The results for both evaluations showed that the model that trained on multiple data sources obtained significantly better results compared to the model that trained only on Twitter data as seen from the loss values, accuracy and even metric values.

Binary Sentiment Analysis improvement effectiveness check (Proposed model)

The binary sentiment analysis task was used to evaluate the performance of the Proposed

model. 2 variations of the Proposed model were trained to re-evaluate the effectiveness of the WHLA improvement. The fine-tuning dataset was used to perform the training and was split into a training and evaluation set of 80:20. The dataset was tokenized using the expanded tokenizer and all models were trained across multiple epochs to analyze the learning trends of the models. Each epoch took approximately 1 hour to complete.

Fine-tuning results

Table 13: Evaluation results for binary sentiment analysis (Proposed model)

Model	Epoch	Training Loss	Validation Loss	Accuracy	F1 Score	Precision	Recall
Proposed model (Pretrained Expanded Model only)	1	0.2126	0.2034	0.9177	0.9173	0.9259	0.9089
	2	0.2245	0.2202	0.9129	0.9108	0.9378	0.8854
	3	0.1293	0.2912	0.9179	0.9177	0.9238	0.9116
Proposed model (Pretrained Expanded Model + WHLA)	1	0.2102	0.2002	0.9184	0.9181	0.9250	0.9115
	2	0.2263	0.2148	0.9136	0.9126	0.9276	0.8980
	3	0.1359	0.2998	0.9172	0.9170	0.9234	0.9108

WHLA layer weights

Table 14: Weight values for Proposed model with WHLA in table 13

Layer	L9	L10	L11	L12
Weight	0.9944	0.9940	1.0102	0.9815

The Proposed model with the WHLA wrapper was shown to outperform the Proposed model without the WHLA wrapper as seen from the loss values and the accuracy score. The weights of the WHLA wrapper were also shown to have a larger change for some layers as compared to the WHLA model from the fine-tuning improvement check results.

Ternary Sentiment Analysis Benchmark Check

The Semeval 2017 dataset was used to evaluate the Proposed model and benchmark it against the other industry models used in research studies such as TweetEval [57, 58]. The SemEval

2017 dataset provided training, validation and test set for the model evaluation. 2 models, the Base model and the Proposed model were trained in this task to identify the model benchmark performance, as well as to check the effects of the proposed alterations to the Base model. The macro recall evaluation metric was used as the identifier for model performance in this task based on the provided benchmark results. The datasets were tokenized separately using the Base BERT Model tokenizer and the expanded tokenizer before feeding the inputs into the model for training. All models were trained across multiple epochs to analyze the learning trends of the models. The training time for each epoch took approximately a minute and a half for completion.

Evaluation Dataset

Table 15: Evaluation results for ternary sentiment analysis

Model	Epoch	Training Loss	Validation Loss	Accuracy	F1 Score	Precision	Recall
Base Model	1	0.6360	0.6094	0.7390	0.7214	0.7180	0.7256
	2	0.5957	0.6185	0.7305	0.7152	0.7119	0.7269
	3	0.3210	0.7305	0.7390	0.7166	0.7173	0.7158
Proposed Model	1	0.6654	0.6158	0.7260	0.7105	0.7066	0.7159
	2	0.6221	0.6239	0.7195	0.7052	0.6949	0.7235
	3	0.3326	0.7953	0.7145	0.6952	0.6906	0.7009

WHLA layer weights

Table 16: Weight values for Proposed model in table 15

Layer	L9	L10	L11	L12
Weight	0.9981	0.9980	0.9990	1.0030

Test Dataset

Table 17: M-Rec benchmark for SemEval 2017 ternary sentiment analysis

S/N	Model	Recall
1	BERTweet	73.4
2	RoB-RT	72.6
3	RoB-Bs	71.3
4	Base model	70.0
5	Proposed model	69.9
6	RoB-Tw	69.1
7	SVM	62.9
8	FastText	62.9
9	BLSTM	58.3

From the evaluation dataset, the Base model and the Proposed model obtained similar scores to each other, with the Base model outperforming the Proposed model by a small amount. The weights of the WHLA wrapper show little change from their original initialized tensor one values.

The Base model and the Proposed model also obtained relatively close macro recall scores compared to the top-ranking models of the benchmark table. They obtained the 4th and 5th rank in the benchmark table, losing to the RoBerta-Base (RoB-Bs), RoBerta pre-trained on additional Twitter corpus (RoB-RT) as well as the BERTweet models while winning against the RoBerta model trained from scratch with Twitter data (RoB-Tw) and the remaining traditional machine learning models.

Multi class sentiment analysis (6 Emotions class)

The twitter emotions dataset was used to evaluate the effectiveness of the model improvements on a multi-class classification task. The dataset was split with an 80:20 ratio into the training set and evaluation set respectively. For this task, the datasets were tokenized separately using the Base BERT Model tokenizer and the expanded tokenizer before feeding the inputs into the model for training. All models were trained across multiple epochs to analyze the learning trends of the models. The training time for each epoch took approximately 17 minutes to complete.

Fine-tuning results

Table 18: Evaluation results for emotions sentiment analysis (6 labels)

Model	Epoch	Training Loss	Validation Loss	Accuracy	F1 Score	Precision	Recall
Base Model	1	0.0980	0.0939	0.9427	0.9163	0.8972	0.9433
	2	0.0817	0.0913	0.9425	0.9162	0.9017	0.9458
	3	0.0821	0.0908	0.9424	0.9063	0.9296	0.9015
Proposed Model	1	0.1068	0.0975	0.9414	0.9064	0.9254	0.9070
	2	0.0860	0.0961	0.9428	0.9172	0.8966	0.9505
	3	0.0825	0.0918	0.9415	0.9080	0.9098	0.9158

WHLA layer weights

Table 19: Weight values for Proposed model in table 18

Layer	L9	L10	L11	L12
Weight	0.9714	0.9862	1.0116	0.9838

The Base model and the Proposed model obtained identical results across all 3 epochs from the loss values, accuracy score and metric values. There were no signs of overfitting for the emotion's classification task, as seen from the continuous decrease in both training and loss values for all 3 epochs. For the WHLA wrapper of the Proposed model, the weights have shown to have been affected much more than previous tasks as well.

Multi class sentiment analysis (28 Emotions class)

The modified GoEmotions dataset was used to evaluate the effectiveness of the model improvements on a more complex multi-class classification task. The accompanied training dataset, evaluation dataset and test dataset were used to obtain the results below. For all the datasets in this task, they were tokenized separately using the Base BERT Model tokenizer and the expanded tokenizer before feeding the inputs into the model for training. All models were trained across multiple epochs to analyze the learning trends of the models. The training time for each epoch was exceptionally fast, only taking approximately a minute and a half to complete.

Evaluation Dataset

Table 20: Evaluation results for emotions sentiment analysis (28 labels)

Model	Epoch	Training Loss	Validation Loss	Accuracy	F1 Score	Precision	Recall
Base Model	1	1.3765	1.2827	0.6214	0.4490	0.5170	0.4363
	2	1.0974	1.2438	0.6238	0.4739	0.5085	0.4627
	3	0.8100	1.3303	0.6128	0.4896	0.5284	0.4886
Proposed Model	1	1.3833	1.2480	0.6247	0.4860	0.6367	0.4760
	2	1.1135	1.2263	0.6321	0.5065	0.6120	0.4860
	3	0.8533	1.3112	0.6212	0.5134	0.5869	0.5134

WHLA layer weights

Table 21: Weight values for Proposed model in table 20

Layer	L9	L10	L11	L12
Weight	0.9838	0.9820	1.0050	1.0173

Test Dataset

Table 22: Test results for emotions sentiment analysis (28 labels)

Model	Test Loss	Accuracy	F1 Score	Precision	Recall
Base Model	1.3217	0.6202	0.4970	0.6062	0.5040
Proposed Model	1.3182	0.6248	0.5208	0.5865	0.5201

From the above results, the overall model performance on the datasets is poor, with low values of accuracy and F1 score. For comparisons between the 2 models, the Proposed model managed to outperform the Base model in terms of overall performance from both the Evaluation Dataset and the Test Dataset. The weights in the WHLA wrapper also show signs of change in certain layers. Signs of overfitting for both models only started to appear at the third epoch. It is to be noted that during the evaluation process of the model, the trainer showed warnings and indication that the model is failing to predict even a single instance of several class labels.

Compute Resource Used

In terms of compute resource, the individual improvements did not show any signs of significant

increase to the computation overhead during the training pipelines. The figure below shows the compute resource used during one of the training sessions for the binary sentiment analysis task involving the different implementations of improvements to the Base model.

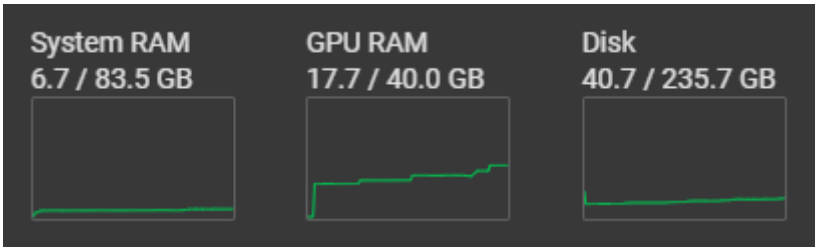


Figure 10: Compute Resource for Binary Sentiment Analysis Task

During this training sessions, multiple models were trained after one another to test the different improvements to the model. The GPU RAM shown in the figure shows a slight increase in GPU ram used as models are trained. The line graph for the GPU ram starts with the compute resource used by the Base model, utilising the lowest ram and increases as the different improvements are implemented.

5.7 Practical Testing

The testing section will involve providing real user input to the various trained models to observe the model behavior. The models that will be tested in this section will only cover the Base model and the Proposed models for the different tasks. The tasks that the Base model and Proposed model will be tested against are:

- Binary Sentiment Analysis
- Ternary Sentiment Analysis
- Emotions Sentiment Analysis (6 labels)

The emotion sentiment analysis for 28 labels will not be tested here due to its poor performance as well as its inability to perform a classification for certain labels.

For binary sentiment analysis, the Base model and the Proposed model will be compared against each other to showcase the differences in how each model arrives to their classification. For subsequent tasks, only the Proposed model will be tested as the model results between the Base model and Proposed model are similar. For both models, when provided a short text input with a clear sentiment and context, the model often confidently predicts the sentiment, usually arriving at a 90% confidence level regardless of whether it predicted the sentiment

correctly or not. For slightly longer sentences, when various slang and emoji are included, there are instances where the confidence of predictions tend to decrease. For paragraphs of text, the model performance noticeably starts to decrease when slang usage is involved, wrongly classifying the text record despite the clear intent of the sentiment delivered throughout the text.

The models also have issues classifying certain statements where sarcasm is involved, however there are instances where the confidence level of the prediction is at 50-60%, indicating that the model is aware that the statement is not directly clear on the intent of the sentiment delivered.

The difference between how the classification of the sentiment is performed lies on the identification of the tokens based on the vocabulary library of the respective model and tokenizer. Using the gradient-based attribution as mentioned in the above section for the implementation of the web application, the word contribution towards the sentiment of a text record can be identified. A clear example of how the Proposed model may use the weight embeddings of the tokens differently from the Base model is by using an example of a token that was added to the Proposed model.

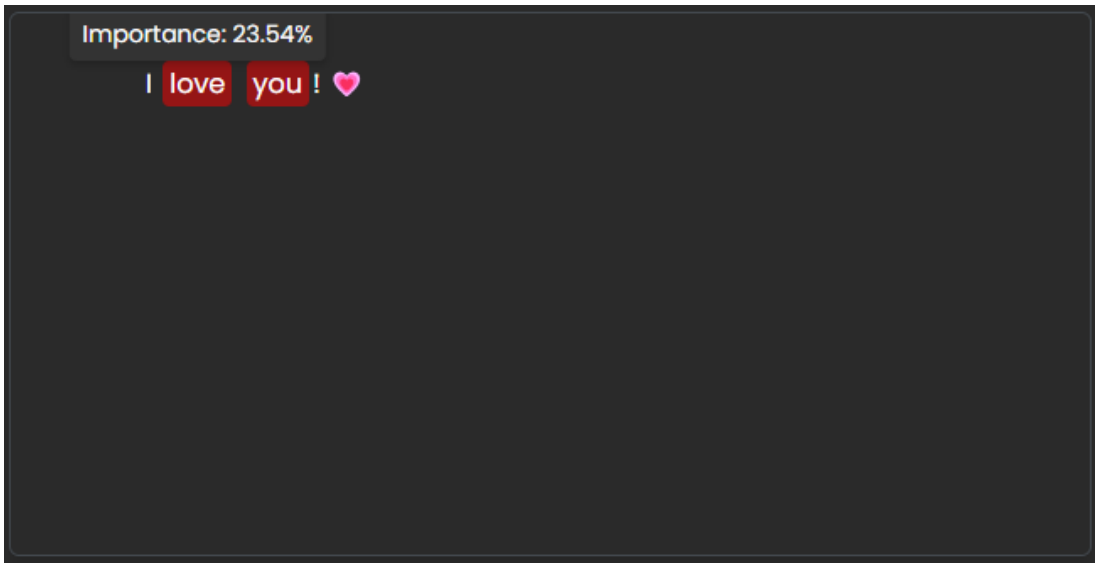


Figure 11: Word importance value with Base model

For a statement ‘I love you!’, with a heart emoji at the end, the Base model predicts the statement to be positive with a confidence rating of 99%. Looking at the tokens and their contribution returned by the API response, the token ‘love’ contributed a value of 23% to the sentiment. The Base model does not recognize the heart emoji at the end of the sentences, thus assigning it the [UNK] token, however, it recognizes that given the context of the statement,

an unknown token at the end is likely a token that reinforces the sentiment of the statement, does attributing the token to a contribution of 40%.

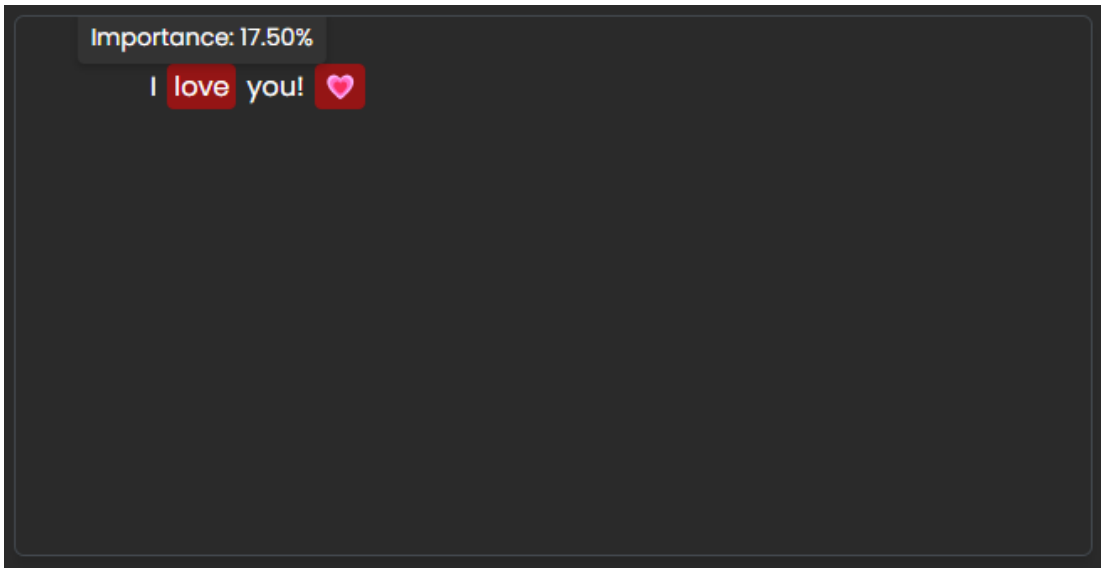


Figure 12: Word importance value with Proposed model

In contrast, when the Proposed model is used to predict the same sentence, the contribution of each token differs from the Base model. In this case, the word ‘love’ only contributed 17.4% to the sentiment of the statement compared to the Base model value of 23.5%. The Proposed model also recognized the heart emoji at the end of the statement and attributed the token with a contribution value of 60% compared to the Base model value of 40%. This showcases how the Proposed model performs its classification with tokens that it can recognize by making use of its weight embeddings.

The Proposed model for the ternary sentiment analysis task showed similar behavior when predicting text records. However, as this model was fine-tuned to be able to recognize neutral statements, when presented with the same neutral statements provided to the binary sentiment analysis model, it manages to successfully classify it as a neutral statement. For longer text, there are instances where the model fails to classify the statement as neutral or classifies the neutral statement with a lower confidence rating.

For the emotion's sentiment analysis, the model also has similar struggles when it comes to detecting sarcasm in statements. For the emotion classification itself, the model struggles with classifying certain labels such as ‘love’ and ‘surprise’. Certain statements that clearly deliver the sentiment intent of ‘love’ will be misclassified as ‘joy’ instead, or statements that deliver the ‘surprise’ intent can be misclassified between ‘anger’, ‘sadness’ and even ‘fear’. The general

confidence of the predictions performed by the model is lower as well, sometimes even reaching values of 30-40% although there are also instances where the model is extremely confident, especially towards the ‘joy’ emotion.

5.8 Discussion

The discussions section will provide an analysis of the results obtained, as well as provide potential reasonings for the model's behaviour.

Overall, the default base BERT model has proven to perform well on the sentiment analysis tasks, obtaining low loss values along with high accuracy and metrics for the binary and emotions sentiment analysis task. It also managed to closely match the top-ranking benchmarks obtained by a superior model for the ternary sentiment analysis task. The only task which the model failed to perform well was the GoEmotions sentiment analysis task which consists of 28 emotions label. The amount of compute resource used was also affected when improvements were applied to the model, which lines up with key findings from the literature review where added complexity to the model can affect computational cost. It is also noted that with sufficient hardware to cover the additional computational cost, the efficiency of training will not dwindle, as seen from how all the models have similar training times despite the added complexity of certain models.

The trend of the performance of the BERT model matches the complexity of the task relatively well. The complexity of the task is derived from various factors, some of which includes the quality of the dataset, the number of tokens for the text records, the nature of the language provided and the number of labels.

The binary sentiment analysis task, consisting of only 2 labels can be considered a low complexity task. While the nature of the language used in the dataset may include use of slang and certain nuance, it can be speculated that the sentiment conveyed by each text record is direct and straight to the point. The emotions label task, while consisting of 6 different emotions label, was analysed to contain short and simple text records, with token numbers not exceeding over 100 tokens. An overview of the language used in this emotions dataset also shows that while social media slang is used, the text records to each emotions label are not vague and nuance.

For the SemEval 2017 dataset, while the number of tokens for the text records do not exceed

100, the nature of language used is slang heavy across the 3 different sentiments. For the 28-emotion label dataset, the number of tokens does not exceed 100 as well, but the high number of classes to look through to perform classification makes it a very complex task. As such, the evaluation of model performance for the binary and 6 emotions label task can be seen to be less complex than the ternary and 28 emotions label task.

Despite the variations in the model performance across all the different tasks, the practical testing of the models show how they still struggle with nuance in language such as sarcasm. The models also struggle with longer instances of text that uses slang and emojis, often affecting the confidence and accuracy of the classification. When it comes to emotions classification, the model misclassification of the 'love' and 'surprise' labels coincide with the distribution of emotions labels. The emotions dataset was found to be unbalanced as seen from the data preparations section in 5.2 and the 'love' and 'surprise' labels were also represented the lowest counts of data records among the 6 labels. This likely caused the model to favour the classification of the 'love' and 'surprise' emotion to other emotions with similar sentiment, for example, misclassifying statements with the emotion intent of 'love' with 'joy'. This represents a case where an imbalanced fine-tuning dataset can lead to poor performance in classifying certain labels. The concept of imbalanced dataset is also likely the reason why the GoEmotions dataset performed so poorly, due to a single label dominating roughly a third of the entire dataset and some labels that only represent less than 1% of the dataset.

Next, a review of the improvements applied to the base BERT model during the binary sentiment analysis task shows that it has little effect on the performance of the model. This means that while the improvements applied to the model has improved the loss values, it was still not able to help the model detect the nuance in the language of text records that were misclassified in the base BERT model.

WHLA model

The WHLA model obtained extremely similar results to the Base model. The weights assigned to the hidden layers of the model had little to no change from the original tensor values as well. This implies that the semantic representations of the last 4 layers of the BERT model were not varied enough to provide an improved output during aggregation. The low complexity of the binary sentiment analysis task could also imply that the final layer is sufficient to accurately classify the sentiment class of the text record. The number of tokens per text record might also play a part in the effectiveness of the WHLA wrapper as a lower number of tokens present in the text record can indicate a lower amount of semantic representation to be captured in the

last 4 hidden layers.

Pretrained model

Out of the 3 improvements applied to the model, the model that underwent the additional pre-training via MLM was the only improvement that improved the model during the fine-tuning task by further reducing the loss values and increasing the accuracy scores and metrics. Based on the MLM results, the model was observed to be able to learn the contextual representations of the different domain languages used across social media by accurately predicting the masked text as seen from its loss values decreasing over epochs. With a better understanding of the domain language of the dataset, it is expected that the model will perform better than the original without any prior domain pre-training. With this result, a hypothesis was made that if the training and validation loss of the model decreases with the MLM task, the performance of the model will increase during the fine-tuning task.

Expanded model

The Expanded model was the only model out of the 3 improvements that had a reduction is performance from the base BERT model. The added vocabulary to the model and tokenizer with initialized values likely added additional noise during the classification than actual representations of the text. It was initially hypothesized that due to the methods of extraction of vocabulary to be added to the model and tokenizer, that the training dataset did not cover enough instances of the vocabulary added to learn the proper representations of the tokens and stabilize the weights. However, after the second epoch, the model was observed to significantly overfit to the training dataset as compared to the second epochs of the other improvements, proving that the model is learning the representations of the added tokens correctly. Thus, it is also possible that the added tokens used in the training dataset do not contribute much to the sentiment delivered in the text record, acting as noise instead.

Proposed model

During the implementation of the Proposed model, the model and tokenizer were expanded and underwent the additional pre-training via MLM. The results showed that the added tokens improved the ability of the model to learn the contextual representations of the domain language during MLM after the second training epoch. This was an expected behaviour of the model, this was an expected behaviour of the model, as the added tokens opened new feature embeddings for domain specific words to learn and update weights.

The hypothesis made previously regarding the direct relation that a decrease in loss value in

MLM training will result in a decrease in loss value during fine-tuning was proved wrong as the results of the Expanded pre-trained model did not outperform the Base pre-trained model as seen in Table 9 and Table 13. The contextual representations of language that the model learned during the MLM task was able to properly train the weight embeddings of the added tokens to offset the noise that it contributed to the classification but was not enough for it to outperform the Base pre-trained model. One surprising observation was that with the added WHLA wrapper, the completed Proposed model was able obtain the best performance out of all the models, beating the Base pre-trained model by a small amount despite previously showing no effect on the base BERT model. It is likely that due to the Expanded pre-trained model, the model opened additional semantic representations that allows the WHLA to capture and contribute to the final classification. This can also be seen from the final weights of WHLA wrapper, where the weights were altered more than before.

Outside of the binary sentiment analysis task, the Proposed model was also fine-tuned on the other task to evaluate its performance such as the ternary and emotions classification tasks. During these tasks, the Proposed model was evaluated against the Base model to identify if the improvements had any effect on the model. However, the results obtained from the tasks indicated that the model performed similarly to the Base model. One possible reasoning for this behaviour is that the improvements provided to the model was not able to generalize outside of the dataset it was trained on. The contextual representations that were learned from the MLM pre-training task was too tailored to the pre-training dataset. The tokens that were added to the model and tokenizer may not be used in the other datasets used for the training and evaluation of the other tasks; thus, the improvements would have no effect on the classification performance of those tasks. The WHLA wrapper applied to the Expanded pre-trained model would be reminiscent of applying the wrapper around the Base model due to the lack of contribution from the other 2 improvements, resulting in no effect as well.

Implications and insights

The 3 improvement methods extracted from the existing literature and implemented through the methodology were shown to only be partially effective across all the proposed task used for evaluation. Despite these improvements showing exceptional results when used in their own respective studies and domains, the results obtained in the above experiments indicate that these improvements do not generalize well across other task and domains.

One likely suspect that could be holding back the effectiveness these improvements could be the collated dataset sourced in the above process. The collated dataset consisting of the 5

different social media sources were not enough to provide enough generalization to the model during the MLM pre-training process, thus there was a lack of improvement to the model when it was used to perform fine-tuning on tasks that require a separate training dataset as the language used may differ. This can be seen on how the Proposed model that underwent the MLM pre-training process only improved on tasks where the fine-tuning dataset is related to the pre-training dataset. Based on the results, the Proposed model showed signs of improvements over the Base model in the binary sentiment analysis task and the 28-emotions sentiment analysis task. For the binary sentiment analysis task, it was extracted alongside the pre-training dataset from the collated dataset. For the 28-emotions label classification task, data records from the fine-tuning dataset are also present in the pre-training dataset. For the remaining tasks, the fine-tuning datasets were not related to the pre-training dataset in anyway, and there were no signs of improvements for the Proposed model compared to the Base model in those cases.

The possible cause of the lack of generalization from the collated dataset could be due to the how these datasets provided too much of a focus on specific topics. For example, the Instagram dataset was focused on social media language surrounding the covid-19 topic or the reddit dataset that was focused on the election of the next India prime minister.

Another factor to consider is the evolution of language use on the internet throughout the years with the use social media platform. With the help of social media, the rate at which various social groups on the internet converse and create new types of slang increases. Thus, a dataset of labelled text records collected from social media in 2015, is likely to have different use of slang and emojis compared to a labelled dataset collected in 2020. In addition, the type of demographic that is present on these platforms can contribute various types of slang when conversing about various topics, for example, a younger generation may converse in a more informal tone, filled with slang and references to pop culture while an older generation of users may converse in a more formal tone, with less use of slang and nuance. As such, the datasets collated may have lacked the generalization across various demographics of users, thus limiting the generalization capabilities of the model [59].

As for the improvement methods, there are certain takeaways from the results that could provide a better understanding of their effectiveness. The WHLA wrapper benefits more when it can capture more semantic representations of the text from the model embeddings and the text record itself. The 512 token limit that BERT models are limited to might not be enough for the WHLA wrapper to properly showcase its effectiveness and it is likely that the improvement

may perform better on a text record with a larger sequence length.

The additional vocabulary added to the model and tokens, despite not being able to increase the results of the model after fine-tuning, was shown to still provide a certain degree of contribution to the sentiment classification of the model as seen from the practical testing of the model. Thus, instead of relying purely relying on the number of slang and emoji occurrences in the dataset to identify what tokens to add to the model, identifying the usefulness of these tokens in terms of providing context and sentiment can prove to be more effective instead. This is especially particular for emojis, for example, the emoji of the planet Earth, a house or even a camera is unlikely to contribute to the sentiment of a text record compared to a “heart” or “thumbs up” emoji, that can usually indicate a positive sentiment or a “crying” emoji that can indicate a negative sentiment. The difficulty that comes with this method to identifying how to obtain a proper weight embedding of the token if the dataset does not provide enough occurrences of the token for proper fine-tuning.

The additional training with MLM has shown that without a proper domain dataset that has sufficient generalization capabilities, the model's ability to fine-tune afterwards on the domain may not improve. In terms of situations where the model performance does improve after fine-tuning, the correlation between the loss values after MLM training and the increase in model performance is minimal and would require a larger reduction in loss values before further improvements will show. This can be seen from how despite the Expanded model obtaining a lower validation loss value of 1.888 compared to the base BERT model of 2.012, there was no signs of increase in model performance during the fine-tuning process.

The TweetEval paper also provides some insight on the relationship between resource invested into training the model to the amount of improvement to the performance of the model regarding additional pre-training with MLM. Take BERTweet for example, the model was trained from scratch with an access to a large corpus of 850 million English Tweets. This pre-training process made use of extensive hardware such as 8 V100 GPU across a total of 40 epochs over 4 weeks [60].

The RoBERTa models under the Tweet Eval paper also had access to approximately 60 million tweets along with 8 V100 GPUS that was trained for over 9 days for both pretraining and training from scratch with MLM [58].

It can be seen from both instances of training that large amounts of data and resources were

required to help improve the performance of the model during fine-tuning. With the current project, the pre-training dataset that was sourced only amounted to a value of 1.7 million data records and trained up to a maximum of 6 epochs with a higher learning rate. With these findings, it matches up with the reason why the Proposed model did not show any improvements over the Base model.

Lastly, the results show how the performance of the model by evaluating them on the datasets of the specific task do not directly translate into the performance of the model in a real-world scenario. The models that were trained on the binary and 6-emotions label task obtained high accuracies and F1 score of over 90%, yet the practical testing shows instances of the model struggling with various text records. The SemEval dataset that was used to train the ternary sentiment analysis model only obtained an accuracy and f1 score of 70%, but this model still manages to accurately predict many positive, negative and neutral statements.

Criticisms and considerations

During the creation of the methodology and implementation phase, there are several areas that needs to be acknowledged that could have potentially affected the validity of the results obtained.

The dataset used to train and evaluate the model is a candidate that could have affected the model results. Firstly, the datasets sourced could have a potential bias during the selection process. Along with it, the quality of the dataset and accuracy of the label can only be relied on through the authors confirmation and process. For example, the sentiment140 was a dataset was that labelled through a script, by identifying positive and negative text records via the presence of positive and negative emojis respectively. Thus, the quality of the dataset cannot be confirmed as there may be edge cases where the presence of such emojis does not directly correlate to the user sentiment. For datasets that were manually labelled, especially for datasets that contain more than 2 labels, the sentiment or emotion perceived by the author of the dataset may be interpreted wrongly due to subjective reasons. These factors may influence the training of the model, leading to an inaccurate representation of the model performance.

For all the results obtained above, they were all trained using the same set of training parameters except for the warmup step value. Thus, there could be a possibility that the performance of the model along with its improvements could vary with the use of other training parameters. For example, the increase in dropout rate or the decrease in learning rate could have prevented the rate of overfitting of the models, leading to a more accurate representation

of the model results by training over more epochs. The change in warmup step value across multiple training sessions for multiple epochs could have also played a role in affecting the credibility of the results.

Future works and implementations

Data represents one area that can be worked on and improved that could potentially bring about better and more credible results. However, this also serves as a challenge as the amount of publicly available datasets from different social media platforms are limited. The datasets that are available also mainly focus on the Twitter platform, which was why many of the datasets and tasks covered in this project was covered the Twitter data source despite data generalization being a focus. Possible solutions to acquiring datasets from more data sources would include performing a web scrape of those text records and performing manual labelling, however, this would be time consuming and would require additional manpower to collate a dataset of sufficient size. The datasets procured should also undergo a certain level of review to help ensure a level of credibility of the quality of the dataset when performing the evaluation of the model. The goal should be to procure a dataset of sufficient quality that has a balanced ratio between all data sources and labels to perform a fair evaluation for the training of models.

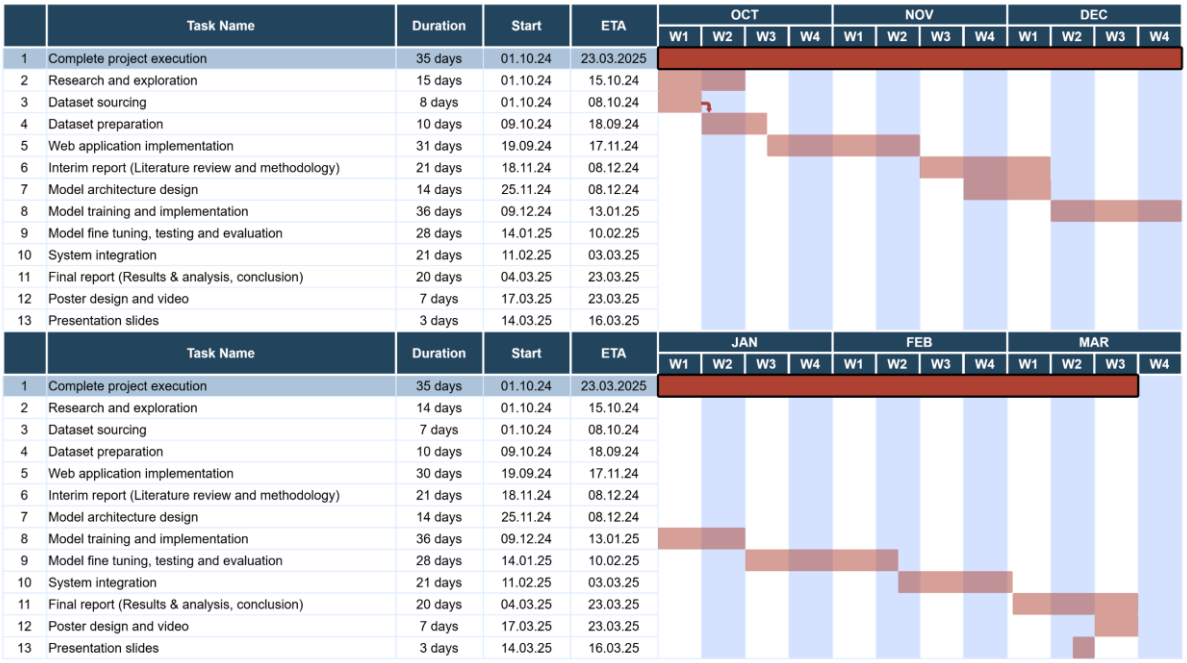
With the 3 different improvements covered in this project, there exist other forms of techniques and methods applied in other domains during the research of transformer-based text classification tasks. These other forms of improvements can also be investigated and implemented into the Proposed model such as multi-task learning with sarcasm and sentiment analysis along with hierarchical models. The insights and implications section also provided some areas that can be worked on to investigate the effectiveness of these 3 improvements further, such as longer token sequences for the WHLA wrapper or the improved process for the selection of token emojis to be added into the model and tokenizer.

If more resources were acquired, more variations of hyperparameter tuning can be implemented to aid with the evaluation and credibility of the model training results. This also means that longer training sessions can be held with more stringent training parameters, such as low learning rates. Additionally, more complex models can also be considered for the Base model to implement the improvement results on such as the RoBERTa model.

6. Project Management

To help ensure a smooth delivery of the project, effective time management and project management techniques are essential when considering the amount of time allocated to the project as well as manpower and resource. This project was set to be performed in parallel with other commitments such as internships. The amount of manpower assigned to the project is only 1 person and a fixed budget of 150 Singaporean dollars was provided for any capstone resources that needed to be bought to execute the project.

Thus, a project timeline was created to help plan the execution of the project. The project timeline serves as an estimate of the key phases, milestones, and deliverables required to successfully complete the capstone project within the allocated timeframe. The Gantt chart figures below showcase the following project task as well as their estimated time to complete each task. The allocated time given to each task is subject to change and will be largely dependent on the complexity of the task as well as any unforeseen challenges that appear during the execution of the task. The project will be operating in a low resource environment due to the lack of state-of-the-art computing resources and hardware, thus the timeline also considers the number of resources available at hand as the models used in the field of natural language processing could be complex. Testing and retraining the model are required if results are not satisfactory, which could potentially lead to more time and resource needed for model related deliverables. As mentioned in earlier sections when explaining the training process, cloud computing resources like Google Colab were purchased with the allocated capstone budget and used to improve the efficiency and effectiveness of the training process.



	Task Name	Duration	Start	ETA
1	Complete project execution	35 days	01.10.24	23.03.2025
2	Research and exploration	15 days	01.10.24	15.10.24
3	Dataset sourcing	8 days	01.10.24	08.10.24
4	Dataset preparation	10 days	09.10.24	18.09.24
5	Web application implementation	31 days	19.09.24	17.11.24
6	Interim report (Literature review and methodology)	21 days	18.11.24	08.12.24
7	Model architecture design	14 days	25.11.24	08.12.24
8	Model training and implementation	36 days	09.12.24	13.01.25
9	Model fine tuning, testing and evaluation	28 days	14.01.25	10.02.25
10	System integration	21 days	11.02.25	03.03.25
11	Final report (Results & analysis, conclusion)	20 days	04.03.25	23.03.25
12	Poster design and video	7 days	17.03.25	23.03.25
13	Presentation slides	3 days	14.03.25	16.03.25

	Task Name	Duration	Start	ETA
1	Complete project execution	35 days	01.10.24	23.03.2025
2	Research and exploration	14 days	01.10.24	15.10.24
3	Dataset sourcing	7 days	01.10.24	08.10.24
4	Dataset preparation	10 days	09.10.24	18.09.24
5	Web application implementation	30 days	19.09.24	17.11.24
6	Interim report (Literature review and methodology)	21 days	18.11.24	08.12.24
7	Model architecture design	14 days	25.11.24	08.12.24
8	Model training and implementation	36 days	09.12.24	13.01.25
9	Model fine tuning, testing and evaluation	28 days	14.01.25	10.02.25
10	System integration	21 days	11.02.25	03.03.25
11	Final report (Results & analysis, conclusion)	20 days	04.03.25	23.03.25
12	Poster design and video	7 days	17.03.25	23.03.25
13	Presentation slides	3 days	14.03.25	16.03.25

Figure 13: Gantt chart of project timeline

The details of each project task provided in the above project timeline chart is further elaborated in the section below.

Project task

- Research and exploration
    - To research and explore fundamental knowledge surrounding Natural Language Processing to better understand how to tackle the capstone project. Areas of research include and are not limited to, the creation and development of Natural Language Processing models, current existing solutions and pre-trained models available, terminologies used in Natural Language Processing.
- Dataset sourcing
    - To research and identify possible datasets that can be used to train the Natural Language Processing models and identify their feasibility to be implemented as training/testing data.
- Dataset preparation and preprocessing
    - To collate the identified and sourced datasets from the previous task and perform data preprocessing techniques on the dataset to prepare the data to be fed into the models for training as well as testing.
- Web application implementation
    - To implement a web application and API framework code structure to prepare for system integration with the NLP models once training has been completed.

- Interim report (Literature review and methodology)
    - To research on additional related studies to the capstone project and perform the write up on the literature review. In addition, to confirm the methodology and perform the write up for the implementation of the sentiment analysis model.
- Implementation of improved model
    - To prepare the improved model based on the methodology defined in the previous task.
- Model training and fine-tuning
    - To train the model on the training data and perform fine-tuning to achieve the best possible training results of the model
- Model testing and evaluation
    - To test the model and evaluate the performance based on different improvements applied to the model to identify how much the performance of the model has improved due to the applied improvements. The comparisons are made based on different metrics such as precision, recall and F1-scores. Additionally, to compare the performance of the model against other pre-trained models in the field.
- System integration
    - To integrate the model with the web application, this includes setting up the API to process the user input to perform queries to the model as well as any other information retrieval performed between the frontend and backend that may include the model.
- Final report (Results, analysis & conclusions)
    - To perform the remaining write up on the results and analysis of the actions performed based on the capstone project methodology as well as the conclusions of the project.
- Poster design and video
    - To create and design the poster based on the capstone project as well as recording video demonstrations.
- Presentation slides
    - To create and design the presentation slides to prepare for the capstone project presentation.

7. Conclusion

The capstone project explored various techniques that can be used to enhance the performance of sentiment analysis models within the domain of social media. These improvements aim to improve the ability of the model to process and understand the diverse nature of linguistics patterns in language such as slang and emojis. By studying existing literature, studies shown that existing models tend to focus on specific domains and struggle with contextual nuances in language such as sarcasm. To address this, several techniques were referenced from existing literature within the realm of transformers and NLP aimed towards improving model performance for various NLP tasks. The improvements that were identified and implemented includes the vocabulary expansion for models and tokenizers, masked language modeling for additional domain pre-training and hidden layer aggregation.

The experiments in this paper showed that while these techniques do improve the model performance for certain tasks, it did not generalize well across all the sentiment analysis tasks and benchmarks. For instance, the vocabulary expansion technique contributed to the contextual awareness of the model to a certain extent but did not improve the sentiment classification accuracy. The additional domain pre-training helped to improve the model's understanding of the domain-type language used in social media but when the model is applied to unseen data, no signs of performance increase was detected. Lastly, the hidden layer aggregation technique showed some potential in capturing the semantic representations in text among the BERT hidden layers, but did not contribute significantly to increasing the sentiment classification accuracy.

One key takeaway from this project is the importance of data quality and diversity when it comes to developing sentiment analysis models. With evolving forms of language used in social media, existing models require constant updates and retraining to maintain its robustness when performing sentiment analysis in social media.

While this project did not provide any groundbreaking discoveries in the realm on improving sentiment analysis models in social media, it managed to provide insights on the feasibility of implementing various improvements to NLP models in such a domain. This project contributes to the ongoing research and findings for NLP by identifying the strength and weakness of NLP model enhancements. Future works within this domain include exploring more complex forms of techniques used to improve NLP models such as multi-task learning and hierarchical models.

8. Knowledge and Training Requirements

The section list all the knowledge, skillsets and certifications both from the degree programme and beyond that was necessary for the successful completion of the capstone projects.

8.1 Applicable Knowledge from the Degree Programme

The prerequisite knowledge and skillsets from the degree programme that was necessary for the capstone projects are as follows:

No.	Module(s)	Knowledge(s) Applied
1	CSC3109: Machine Learning	Machine Learning provided me with essential knowledge needed to complete the capstone project. The technical details learned from the module such as the architectures of recurrent neural networks and long short-term memory aided my understanding of NLP models used currently. In addition, it helped me understand techniques used in relevant literature as well as how to alter or improve them as part of the machine learning process pipeline.
2	CSC3105: Data Analytics	Data Analytics taught me the fundamentals of executing a full machine learning process pipeline which was extremely useful in this machine learning based capstone project such as implementing data preprocessing procedures as well as model evaluation methods.
3	CSC1106: Web Programming	While the web application is not the sole focus of this project, the module provided essential knowledge on the fundamentals of building a web application which I was able to expand upon with my own learning to build the 2-tier web application as well as performing CRUD via the API framework.

8.2 Additional Knowledge, Skillsets, or Certifications Required

The following are the additional requirements and knowledge that were required for for the capstone projects:

No.	Additional Requirement(s)	Knowledge(s) Applied
1	Natural Language Processing Knowledge	Knowledge required to allow understanding of terminologies and techniques used within the field of natural language processing.

9. References

[1] A. Perrin, "Social networking usage: 2005-2015." in Pew Research Center. https://www.pewresearch.org/internet/2015/10/08/social-networking-usage-2005-2015/ (accessed Sep. 25, 2024)

[2] C. Zachlod, O. Samuel, A. Ochsner, and S. Werthmüller, "Analytics of social media data – State of characteristics and application," in *Journal of Business Research*, vol. 144, pp. 1064-1076, May. 2022. doi: https://doi.org/10.1016/j.jbusres.2022.02.016

[3] L. Sampson, L. D. Kubzansky, and K. C. Koenen, "The Missing Piece: A Population Health Perspective to Address the U.S. Mental Health Crisis," in *Daedalus*, vol. 152, no. 4, pp. 24-44, Fall. 2023. Available: https://www.jstor.org/stable/48749704.

[4] A. S. Yasegnal, "Assessment of Risk Factors and Consequences of Mental Health Problems Among Mental Health Patients Admitted in Felege Hiwot Referral Hospital, Bahir Dar, Ethiopia," in *Journal of Health Psychology*, vol. 30, no. 4, pp. 1-10, Jun. 2021. doi: https://doi.org/10.1177/10541373211020449

[5] M. De Choudhury, E. Kiciman, M. Dredze, G. Coppersmith, and M. Kumar, "Discovering shifts to suicidal ideation from mental health content in social media," in *CHI '16: Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pp. 2098-2110, May. 2016. doi: https://doi.org/10.1145/2858036.2858207

[6] P. M. Nadkarni, L. Ohno-Machado, and W. W. Chapman, “Natural language processing: an introduction,” in *Journal of the American Medical Informatics Association*, vol. 18, no. 5, pp. 544–551, Sep. 2011, doi: https://doi.org/10.1136/amiajnl-2011-000464

[7] R. Agnihotri, K. A. Bakeshloo, and S. Mani, "Social media analytics for business-to-business marketing," in *Industrial Marketing Management*, vol. 115, pp. 110-126, Nov. 2023. doi: https://doi.org/10.1016/j.indmarman.2023.09.012

[8] H. Zhang, Z. Zang, H. Zhu, M. I. Uddin, and M. A. Amin, "Big data-assisted social media analytics for business model for business decision making system competitive analysis," in *Information Processing & Management*, vol. 59, no. 1, p. 102762, Jan. 2022. doi: https://doi.org/10.1016/j.ipm.2021.102762

[9] A. Bovet, F. Morone, and H. A. Makse, "Validation of Twitter opinion trends with national polling aggregates: Hillary Clinton vs Donald Trump," in *Scientific Reports*, vol. 8, no. 8673, Jun. 2018. doi: https://doi.org/10.1038/s41598-018-26951-y

[10] A. Kumar and T. M. Sebastian, "Sentiment Analysis: A Perspective on its Past, Present and Future," in *I.J. Intelligent Systems and Applications*, vol. 10, pp. 1-14, Sep. 2012. doi: https://doi.org/10.5815/ijisa.2012.10.01

[11] M. V. Koroteev, "BERT: A Review of Applications in Natural Language Processing and

Understanding," in *arXiv*, Mar. 2021. doi: https://doi.org/10.48550/arXiv.2103.11943

[12] A. Singh, H. Srivastava, M. Aman, and G. Dubey, "Sentiment Analysis on User Feedback of a Social Media Platform," in *2023 International Conference on Sustainable Computing and Data Communication Systems (ICSCDS)*, Erode, India, pp. 1-6, Mar. 2023. doi: https://doi.org/10.1109/ICSCDS56580.2023.10105082

[13] K. N. Prasanthi, R. E. Madhavi, D. N. S. Sabarinadh, and B. Sravani, "A Novel Approach for Sentiment Analysis on Social Media Using BERT & ROBERTA Transformer-Based Models," in *2023 IEEE 8th International Conference for Convergence in Technology (I2CT)*, Lonavla, India, pp. 1-6, Apr. 2023. doi: https://doi.org/10.1109/I2CT57861.2023.10126206

[14] T. Bikku, J. Jarugula, L. Kongala, N. D. Tummala, and N. V. Donthiboina, "Exploring the Effectiveness of BERT for Sentiment Analysis on Large-Scale Social Media Data," in *2023 3rd International Conference on Intelligent Technologies (CONIT)*, Hubli, India, pp. 1-6, Jun. 2023. doi: https://doi.org/10.1109/CONIT59222.2023.10205600

[15] A. Chiorrini, C. Diamantini, A. Mircoli, and D. Potena, "Emotion and sentiment analysis of tweets using BERT," in *Workshop Proceedings of the EDBT/ICDT 2021 Joint Conference*, Nicosia, Cyprus, pp. 1-14, Mar. 2021. Available: http://www.ceur-ws.org/Vol-2818/paper5.pdf.

[16] N. Azzouza, K. Akli-Astouati, and R. Ibrahim, "TwitterBERT: Framework for Twitter Sentiment Analysis Based on Pre-trained Language Model Representations," in *Emerging Trends in Intelligent Computing and Informatics (IRICT 2019)*, *Advances in Intelligent Systems and Computing*, vol. 1073, pp. 428-437, Nov. 2019. doi: https://doi.org/10.1007/978-3-030-33582-3_41

[17] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects," in *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 12, pp. 6999-7019, Dec. 2022. doi: https://doi.org/10.1109/TNNLS.2021.3084827

[18] Y. Tianyang, "Study on Using Scenarios of Linear and Nonlinear Classifiers," in *2020 International Conference on Computing and Data Science (CDS)*, Stanford, CA, USA, pp. 1-6, Aug. 2020. doi: https://doi.org/10.1109/CDS49703.2020.00022

[19] W. S. Noble, "What is a support vector machine?" in *Nature Biotechnology*, vol. 24, no. 12, pp. 1565-1567, Dec. 2006, doi: https://doi.org/10.1038/nbt1206-1565

[20] H. Zhang and D. Li, "Naïve Bayes Text Classifier," in *2007 IEEE International Conference on Granular Computing (GRC 2007)*, Fremont, CA, USA, pp. 1-5, Nov. 2007. doi: https://doi.org/10.1109/GrC.2007.40

[21] V. Singh, H. V. Kaushik, and Reshma, "Social Media Sentiment Analysis Using Twitter Dataset," in *2024 Second International Conference on Data Science and Information System (ICDSIS)*, Hassan, India, pp. 1-6, May. 2024. doi: https://doi.org/10.1109/ICDSIS61070.2024.10594648

[22] K. S. Yogi, V. Dankan Gowda, D. Sindhu, H. Soni, S. Mukherjee and G. C. Madhu, "Enhancing Accuracy in Social Media Sentiment Analysis through Comparative Studies using Machine Learning Techniques," in *2024 International Conference on Knowledge Engineering and Communication Systems (ICKECS)*, Chikkaballapur, India, pp. 1-6, Apr. 2024. doi: https://doi.org/10.1109/ICKECS61492.2024.10616441

[23] V. Joseph, C. P. Lora, and N. T, "Exploring the Application of Natural Language Processing for Social Media Sentiment Analysis," in *2024 3rd International Conference for Innovation in Technology (INOCON)*, Bangalore, India, pp. 1-6, Mar. 2024. doi: https://doi.org/10.1109/INOCON60754.2024.10511841

[24] Y. J. Liu and A. Zeldes, "Why Can't Discourse Parsing Generalize? A Thorough Investigation of the Impact of Data Diversity," in *arXiv*, Feb. 2023. doi: https://doi.org/10.48550/arXiv.2302.06488

[25] D. Khurana, A. Koli, K. Khatter, and S. Singh, "Natural language processing: state of the art, current trends and challenges," in *Multimedia Tools and Applications*, vol. 82, pp. 3713–3744, Jul. 2022. doi: https://doi.org/10.1007/s11042-022-13428-4

[26] A. Rogers, O. Kovaleva, and A. Rumshisky, "A Primer in BERTology: What We Know About How BERT Works," in *Transactions of the Association for Computational Linguistics*, vol. 8, pp. 842-866, Jan. 2021. doi: https://doi.org/10.1162/tacl_a_00349

[27] H.-T. Duong and T.-A. Nguyen-Thi, "A review: preprocessing techniques and data augmentation for sentiment analysis," Computational Social Networks, vol. 8, no. 1, Jan. 2021. doi: https://doi.org/10.1186/s40649-020-00080-x

[28] V. Sanh, T. Wolf, and S. Ruder, "A hierarchical multi-task approach for learning embeddings from semantic tasks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, pp. 6949–6956, Jul. 2019. doi: https://doi.org/10.1609/aaai.v33i01.33016949

[29] L. Torbarina, T. Ferkovic, L. Roguski, V. Mihelcic, B. Sarlija, and Z. Kraljevic, "Challenges and opportunities of using transformer-based multi-task learning in NLP through ML lifecycle: A position paper," in *Natural Language Processing Journal*, vol. 7, Jun. 2024. doi: https://doi.org/10.1016/j.nlp.2024.100076

[30] S. Ghannay, B. Favre, Y. Estève, and N. Camelin, "Word Embedding Evaluation and Combination," in *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, Portorož, Slovenia, pp. 300–305, May 2016. Available: https://aclanthology.org/L16-1046

[31] M. E. Consens, C. Dufault, M. Wainberg, D. Forster, M. Karimzadeh, H. Goodarzi, F. J. Theis, A. Moses, and B. Wang, "To Transformers and Beyond: Large Language Models for the Genome," in *arXiv*, Nov. 2023. doi: https://doi.org/10.48550/arXiv.2311.07621

[32] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention Is All You Need,” in *arXiv*, Jun. 2017, revised Aug. 2023. doi: https://doi.org/10.48550/arXiv.1706.03762

[33] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, “RoBERTa: A Robustly Optimized BERT Pretraining Approach,” in *arXiv*, Jul. 2019. doi: https://doi.org/10.48550/arXiv.1907.11692

[34] P. He, X. Liu, J. Gao, and W. Chen, “DeBERTa: Decoding-enhanced BERT with Disentangled Attention,” in *arXiv*, Jun. 2020. Available: https://doi.org/10.48550/arXiv.2006.03654

[35] X. Amatriain, A. Sankar, J. Bing, P. K. Bodigutla, T. J. Hazen, and M. Kazi, “Transformer models: an introduction and catalog,” in *arXiv*, Feb. 2023. Available: https://doi.org/10.48550/arXiv.2302.07730

[36] J. C. Timoneda and S. Vallejo Vera, “BERT, RoBERTa or DeBERTa? Comparing Performance Across Transformer Models in Political Science Text,” in *The Journal of Politics*, Feb. 2024. Available: https://doi.org/10.1086/730737

[37] I. Alghanmi, L. Espinosa Anke, and S. Schockaert, “Combining BERT with Static Word Embeddings for Categorizing Social Media,” in *Proceedings of the Sixth Workshop on Noisy User-generated Text (W-NUT 2020)*, pp. 28–33, Nov. 2020. doi: https://doi.org/10.18653/v1/2020.wnut-1.5

[38] N. Bensalah, H. Ayad, A. Adib, and A. Ibn El Farouk, “Combining Static and Contextual Features: The Case of English Tweets,” in *Emerging Trends in Intelligent Systems & Network Security (NISS 2022), Lecture Notes on Data Engineering and Communications Technologies*, vol. 147, Springer, pp. 168–175, Sep. 2022. doi: https://doi.org/10.1007/978-3-031-15191-0_16

[39] O. Galal, A. H. Abdel-Gawad, and M. Farouk, “Rethinking of BERT sentence embedding for text classification,” *in Neural Computing and Applications*, vol. 36, pp. 20245–20258, Aug. 2024. doi: https://doi.org/10.1007/s00521-024-10212-3

[40] H. Wu, X. Ye, and S. Manoharan, “Enhancing Multi-Class Text Classification with BERT-Based Models,” in *2023 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE),* Nadi, Fiji, Dec. 2023, doi: https://doi.org/10.1109/CSDE59766.2023.10487760

[41] W. Tang, R. Chen, X. Wen, and Z. Hu, “Text Classification Based on Hierarchical Weighted Aggregation and Selective Convolution,” in *SSRN*, Jul. 2024. Available: https://ssrn.com/abstract=4881692

[42] G. Jawahar, B. Sagot, and D. Seddah, “What does BERT learn about the structure of language?,” in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL 2019)*, Florence, Italy, Jul. 2019. Available: https://inria.hal.science/hal-02131630/

[43] W. Tai, H. T. Kung, X. Dong, M. Comiter, and C.-F. Kuo, “exBERT: Extending Pre-trained Models with Domain-specific Vocabulary Under Constrained Training Resources,” in *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 1433–1439, Nov. 2020. doi: https://doi.org/10.18653/v1/2020.findings-emnlp.129

[44] Z. Wang, X. Liang, R. Du, J. Tian, and S. Zhang, “TVD-BERT: A Domain-Adaptation Pre-trained Model for Textural Vulnerability Descriptions,” in Advanced Intelligent Computing Technology and Applications (ICIC 2024), Lecture Notes in Computer Science, vol. 14875, Springer, pp. 197–208. Aug. 2024. doi: https://doi.org/10.1007/978-981-97-5663-6_17

[45] H. Tanaka and H. Shinnou, “Vocabulary expansion of compound words for domain adaptation of BERT,” in *Proceedings of the 36th Pacific Asia Conference on Language, Information and Computation (PACLIC 2022)*, pp. 379–387, Oct. 2022. Available: https://aclanthology.org/2022.paclic-1.42

[46] A. Go, R. Bhayani, and L. Huang, "Twitter Sentiment Classification using Distant Supervision," Stanford University, 2009. Available: http://cs.stanford.edu/people/alecmgo/papers/TwitterDistantSupervision09.pdf

[47] C. K. A., "Twitter and Reddit Sentimental Analysis Dataset," Kaggle, 2020. Available: https://www.kaggle.com/datasets/cosmos98/twitter-and-reddit-sentimental-analysis-dataset

[48] D. Demszky, D. Movshovitz-Attias, J. Ko, A. Cowen, G. Nemade, and S. Ravi, "GoEmotions: A Dataset of Fine-Grained Emotions," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 4040–4054, 2020. Available: https://aclanthology.org/2020.acl-main.372/

[49] N. Thakur, "Five Years of COVID-19 Discourse on Instagram: A Labeled Instagram Dataset of Over Half a Million Posts for Multilingual Sentiment Analysis," in *arXiv*, 2024. Available: https://arxiv.org/abs/2410.03293

[50] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning Word Vectors for Sentiment Analysis," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, Portland, Oregon, USA, pp. 142–150, 2011. Available: http://www.aclweb.org/anthology/P11-1015

[51] Yelp, "Yelp Open Dataset," Available: https://business.yelp.com/data/resources/open-dataset/

[52] S. Rosenthal, N. Farra, and P. Nakov, "SemEval-2017 Task 4: Sentiment Analysis in Twitter," in *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, Vancouver, Canada, pp. 502–518, 2017. Available: https://aclanthology.org/S17-2088/

[53] N. Elgiriyewithana, "Emotions," Kaggle, 2024. Available: https://www.kaggle.com/dsv/7563141

[54] MLBtrio, "GenZ Slang Dataset," Hugging Face, 2024. Available: https://huggingface.co/datasets/MLBtrio/genz-slang-dataset

[55] NVIDIA Corporation, "NVIDIA A100 Tensor Core GPU Datasheet," Jun. 2021. Available: https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/a100/pdf/nvidia-a100-datasheet-us-nvidia-1758950-r4-web.pdf

[56] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *arXiv*, 2018. Available: https://doi.org/10.48550/arXiv.1810.04805

[57] CardiffNLP, "TweetEval: Benchmark and Evaluation for Tweet Classification," GitHub, 2020. Available: https://github.com/cardiffnlp/tweeteval

[58] F. Barbieri, J. Camacho-Collados, L. Neves, and L. Espinosa-Anke, "TweetEval: Unified Benchmark and Comparative Evaluation for Tweet Classification," in *arXiv*, 2020. Available: https://doi.org/10.48550/arXiv.2010.12421

[59] M. Carrier, "Because Internet: Understanding the new rules of language (a review)," *Training, Language and Culture*, vol. 3, no. 3, pp. 107–111, 2019. Available: http://doi.org/10.29366/2019tlc.3.3.8

[60] D. Q. Nguyen, T. Vu, and A. T. Nguyen, "BERTweet: A pre-trained language model for English Tweets," in *arXiv*, 2020. Available: https://doi.org/10.48550/arXiv.2005.10200

Appendices

Appendix A: Code Repository Link

GitHub link: https://github.com/Empyrealiv/CSC3101_Capstone

Appendix B: Pseudo Codes

```
FUNCTION Split_Data_For_Pretraining_And_Finetuning:
    INPUT:
        preview_data: DataFrame containing the data
        pretrain_percentage: Percentage of data to use for pretraining (0.70)
        threshold_difference: Maximum acceptable difference in class counts (7000)
        RANDOM_SEED: Random seed for reproducibility

    positive_data = Filter preview_data where for positive labelled data
    negative_data = Filter preview_data where for negative labelled data

    Initialize empty list for pretraining_samples and finetuning_samples

    FOR each unique source in preview_data:
        positive_total = Filter positive_data where data source matches current source
        negative_total = Filter negative_data where data source matches current source

        # Count samples per class
        positive_count = Count of positive_total
        negative_count = Count of negative_total
        difference = Absolute difference between positive_count and negative_count

        IF difference < threshold_difference:
            # Case 1: Counts are closely balanced, perform 70:30 split
            positive_pretrain = Random sample 70% from positive_total
            negative_pretrain = Random sample 70% from negative_total
            positive_finetune = positive_total minus positive_pretrain
            negative_finetune = negative_total minus negative_pretrain

        ELSE:
            # Case 2: Counts are imbalanced
            # First, get the smaller count
            min_count = Minimum between positive_count and negative_count

            # Create balanced dataset for fine-tuning
            positive_finetune = Random sample min_count from positive_total
            negative_finetune = Random sample min_count from negative_total

            # Send remaining samples to pretraining
            positive_pretrain = positive_total minus positive_finetune
            negative_pretrain = negative_total minus negative_finetune

        # Add samples to respective lists
        Add positive_pretrain to pretraining_samples
        Add negative_pretrain to pretraining_samples
        Add positive_finetune to finetuning_samples
        Add negative_finetune to finetuning_samples

    # Combine all samples into final respective datasets
    pretraining_data = Concatenate all pretraining_samples
    finetuning_data = Concatenate all finetuning_samples

    pretraining_data = Shuffle pretraining_data with RANDOM_SEED and reset indices
    finetuning_data = Shuffle finetuning_data with RANDOM_SEED and reset indices

    RETURN pretraining_data, finetuning_data
```

Figure 14: Pseudo code for algorithm to perform data split

```
CLASS WHLA_BERT(Neural Network Model):
    FUNCTION Initialize(pretrained_model="bert-base-uncased", num_labels=2):
        Initialize parent class

        bert = Load BertModel from pretrained_model with output_hidden_states=True
        hidden_size = bert.config.hidden_size

        weights = Create learnable Parameter tensor of ones with size 4
        fully connected layer = Map a linear layer from hidden_size to num_labels
        dropout = Create Dropout layer with rate 0.5
        layer_norm = Create a layer normalization layer for hidden_size

    FUNCTION Forward_Propagation(input_ids, attention_mask, labels=None):
        outputs = Pass input_ids and attention_mask into the bert
        layers = Get all layers from outputs

        L9 = Get 4th-to-last layer output
        L10 = Get 3rd-to-last layer output
        L11 = Get 2nd-to-last layer output
        L12 = Get last layer output

        weighted_sum = Multiply each layer output by its corresponding weight value and sum them together
                       (weights[0] * L9 + weights[1] * L10 + weights[2] * L11 + weights[3] * L12)

        normalized_sum = Apply layer normalization to weighted_sum
        cls_representation = Extract the first token ([CLS]) representation from normalized_sum

        logits = Apply dropout to cls_representation, then pass through the
                 fully connected layer to obtain the logits output

        IF labels is provided:
            loss_function = Create CrossEntropyLoss
            loss = Calculate loss between logits and labels
            RETURN dictionary with loss and logits
        ELSE:
            RETURN logits
```

Figure 15: Pseudo code for WHLA wrapper class

```
FUNCTION Vocab_Expansion:
INPUT:
    token_list: The list of tokens to be added into the model and tokenizer
    STATIC_EMBEDDING_SIZE: The embedding size of the static embeddings used
    static_embeddings: A dictionary containing weight embeddings for different tokens

    Initialize the bert model and the tokenizer

    projection_layer = Create a torch linear layer to project STATIC_EMBEDDING_SIZE
                       to the model embedding size

    Intialize empty list for new_token_list

    FOR each token in token_list:
        If token does exist in tokenizer vocabulary:
            Add token to new_token_list

    Add new_token_list to the tokenizer
    Resize the model token embeddings to the lenght of the tokenizer

    For token in new_token_list:
        If token exist in static_embeddings:
            vector = Obtain the weight embeddings of the token from the static embeddings
            projected_vector = Apply projection_layer to the vector
            Assign the weight embeddings of the token in the model to the projected_vector

        Else:
            Assign randoms weight embeddings to the token in the model

    RETURN model and tokenizer
```

Figure 16: Pseudo code for vocabulary expansion of model and tokenizer

Appendix C: Web Application Screenshots

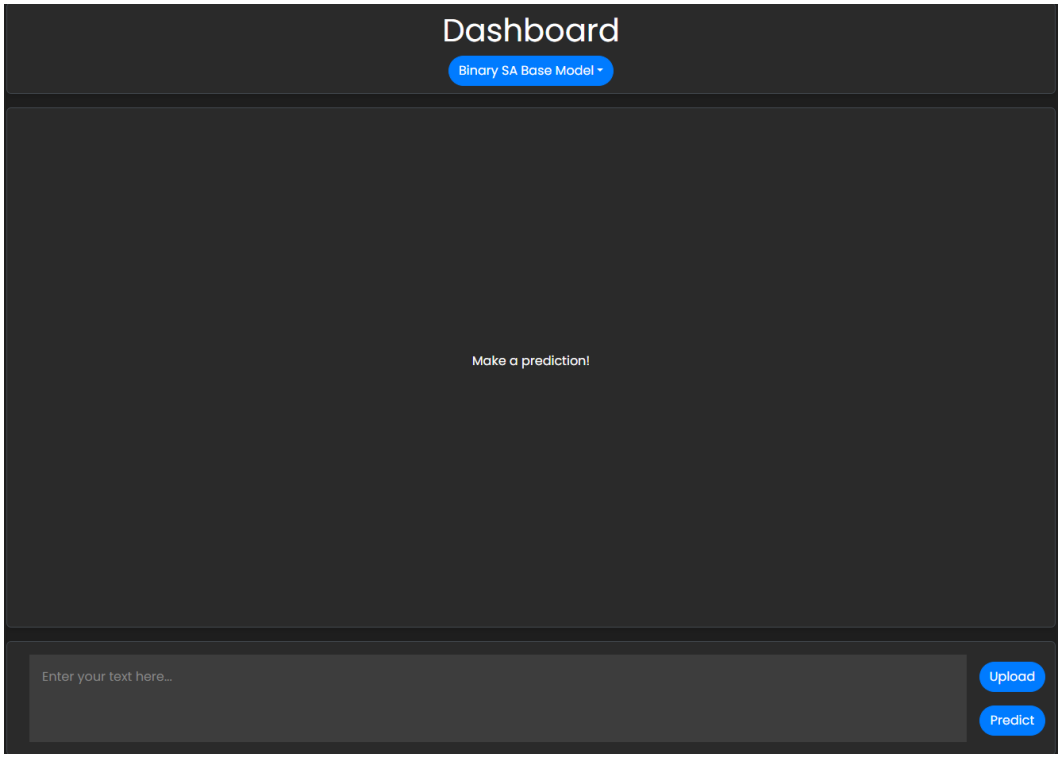


Figure 17: Layout of web application UI

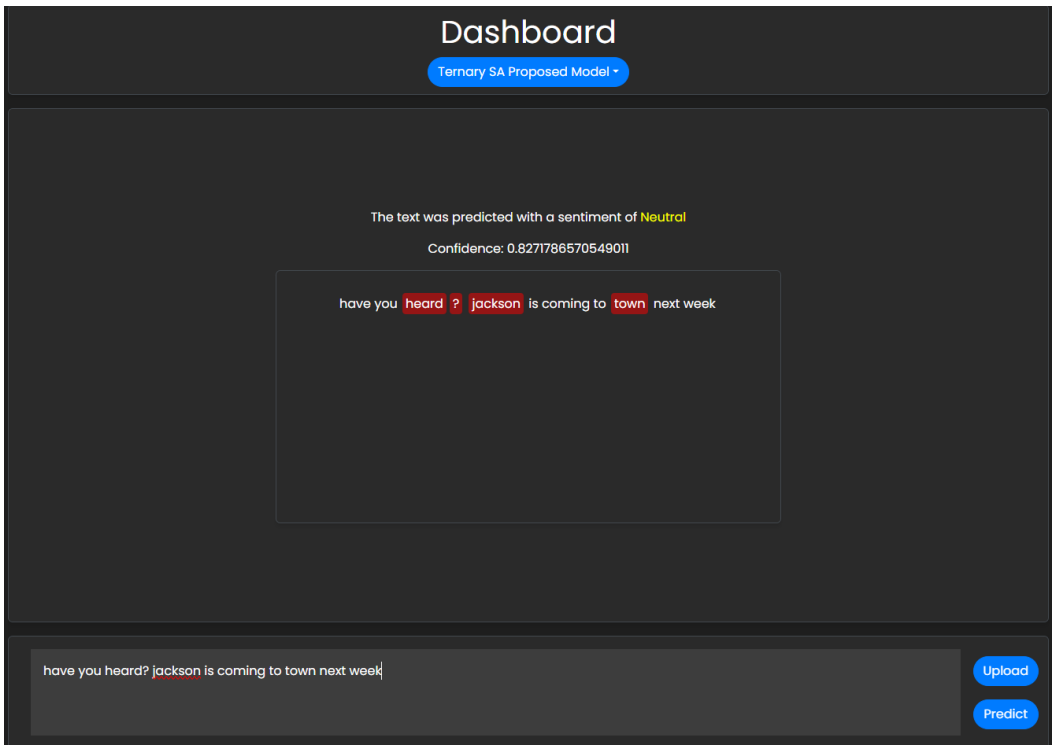


Figure 18: Ternary Sentiment Analysis with web application UI

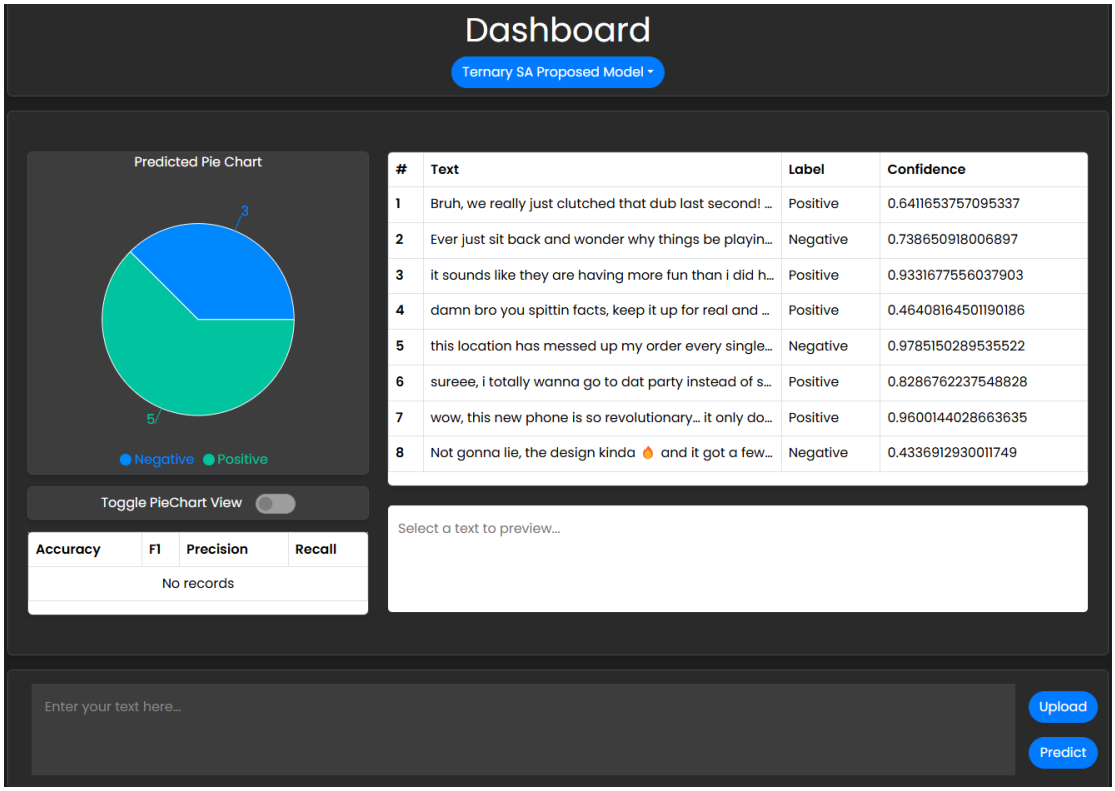


Figure 19: Layout of web application UI predicting multiple sentiments

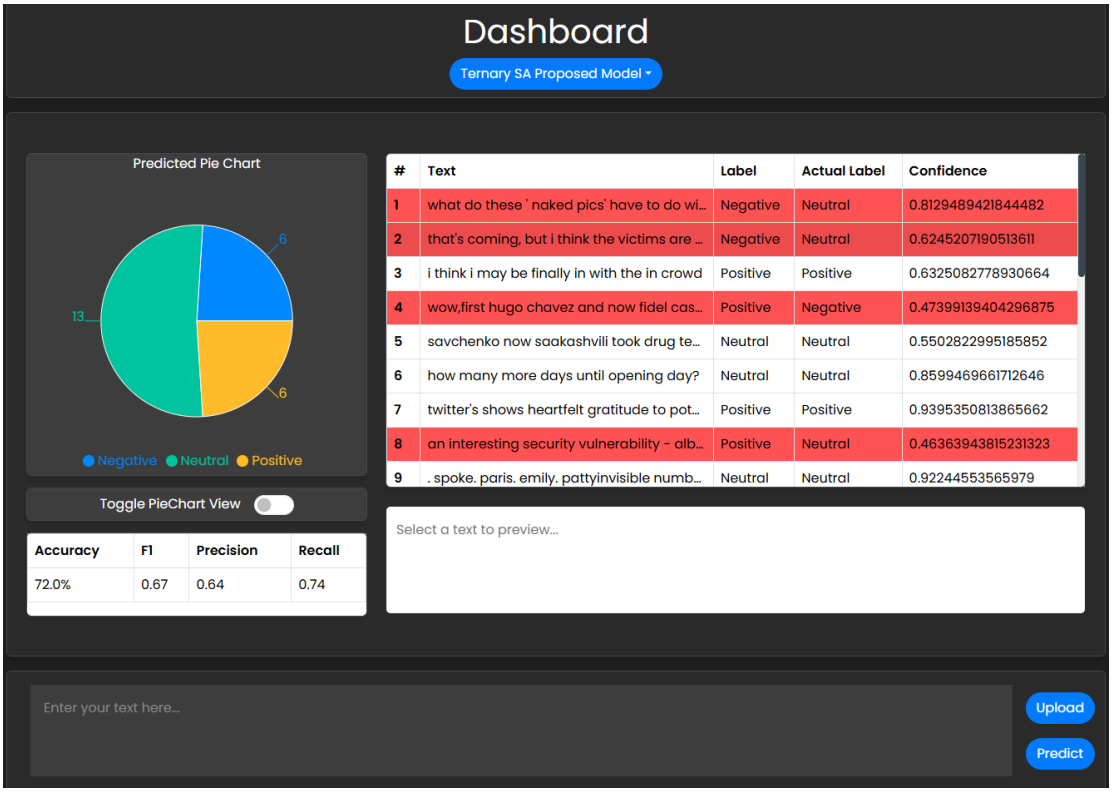


Figure 20: Layout of web application UI predicting multiple sentiments (Evaluation mode)

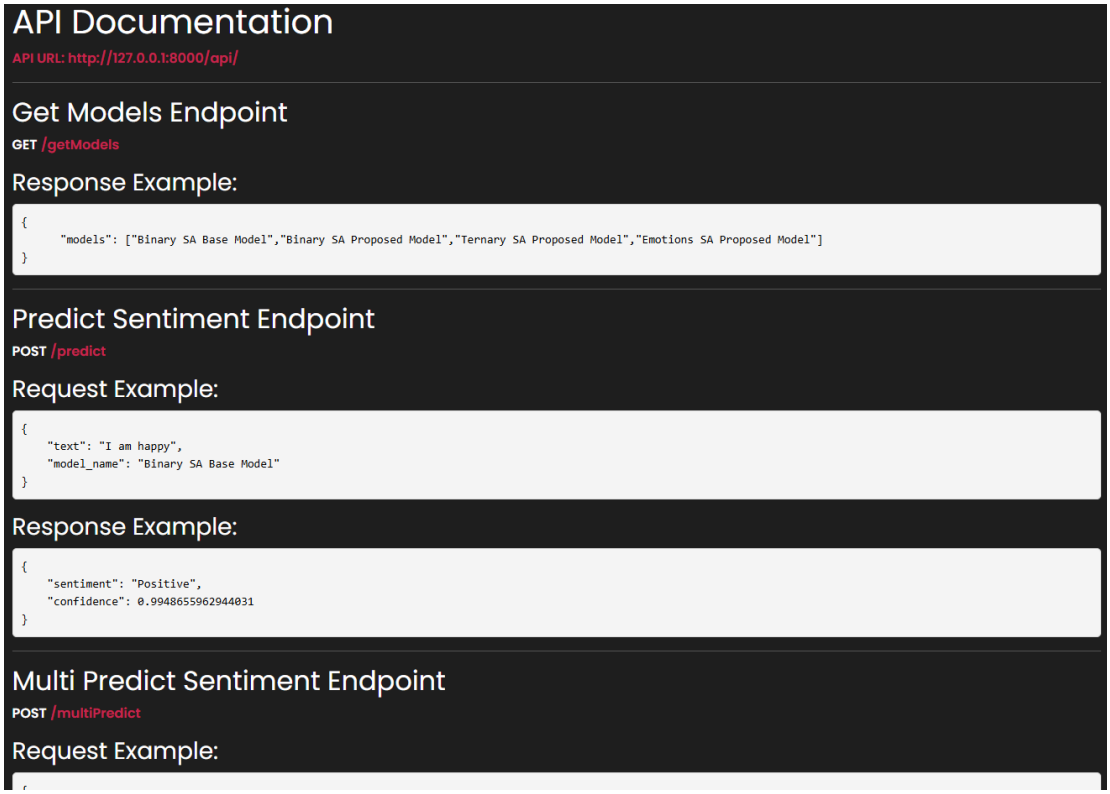


Figure 21: Web application API documentation page

END OF REPORT