UTILIZING ROBERTA INTERMEDIATE LAYERS AND FINE-TUNING FOR SENTENCE CLASSIFICATION

By

Eaint Thet Hmu Soe
B.C.Sc.

A Dissertation Submitted in Partial Fulfillment of the Requirement for the Degree of

Master of Computer Science

M.C.Sc.

University of Computer Studies, Yangon

December 2022

ACKNOWLEDGEMENTS

I would like to express my respectful thanks to **Dr. Mie Mie Khin**, Rector of the University of Computer Studies, Yangon, for her kind permission to submit this thesis.

I wholeheartedly express my sincere gratitude to **Dr. Si Si Mar Win** and **Dr. Thin Zar Thaw** for their administrative support as Deans of Master Course and Professors throughout the completion of this thesis.

I especially would like to thank my supervisor, Professor **Dr. Win Pa Pa**, for introducing and supporting me to this research area and for her advice, guidance and careful editing during the preparation of this thesis.

I would also like to thank **Dr. Aung Nway Oo** for letting my defense be an enjoyable moment, and for your brilliant comments and suggestions.

I would like to extend my deepest gratitude to **Daw Hnin Yee Aung**, Lecturer, Department of English, for editing my thesis paper from the language point of view.

I would like to appreciate the suggestions and recommendations of the teachers who attended all my seminars.

I am grateful to my parents and the rest of my family for their continuous support and encouragement during my graduate studies at UCSY.

I sincerely appreciate Bagan Innovation Technology for supporting the dataset for my thesis.

Finally, I would like to thank all of the staff, teachers from the University of Computer Studies, Yangon for their support.

ABSTRACT

Text classification becomes more and more challenging due to a scarcity of standardized labeled data in the Myanmar NLP domain. The majority of the existing Myanmar research has relied on models of deep learning that significantly focus on context-independent word embeddings, such as Word2Vec, GloVe, and fastText, in which each word has a fixed representation irrespective of its context. Meanwhile, context-based pre-trained language models such as BERT and RoBERTa recently revolutionized the state of natural language processing. In this paper, the experiments are conducted to enhance the performance of classification in sentiment analysis by utilizing the transfer learning ability of RoBERTa. Existing pretrained model based works only utilize the last output layer of RoBERTa and ignore the semantic knowledge in the intermediate layers. This research explores the potential of utilizing RoBERTa intermediate layers to enhance the performance of fine-tuning of RoBERTa. To show the generality, Myanmar pretrained RoBERTa model (MyanBERTa)[1] and multilingual pretrained model (XLM-roBERTa)[3] are also compared. The effectiveness and generality of intermediate layers were proved and discussed in the experimental result.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF EQUATIONS

CHAPTER 1

INTRODUCTION

The process of examining a piece of text to forecast how an individual's attitude toward an occurrence or perspective will be oriented is known as sentiment classification. Text polarity is usually used to analyze sentiment. A sentiment classifier typically classifies information as positive, negative, or neutral. Sentiment extraction is the foundation of sentiment categorization, and extensive research has been conducted in this area. The next critical step is sentiment mining, which has grown dramatically in recent years in tandem with the global increase in textual data. People now share their thoughts on a variety of topics electronically, such as online product reviews, book or film studies, and political commentary. As a result, assessing various points of view becomes critical for interpreting people's intentions.

Previous attempts to implement sentiment analysis in Myanmar relied on non contextualized word embeddings (Word2Vec and fastText), which present a series of static word embeddings without taking into account the many other contexts in which they could occur. However, the recent emergence of the Bidirectional Encoder Representations from Transformers such as BERT, RoBERTa and AlBERT phenomenon greatly amplifies the contextualization strategy. BERT has established itself as the most impressive NLP model capable of performing superbly in any NLP operation with proper fine-tuning for specific downstream tasks as the trend shifted toward transformer based architectures consisting of attention heads. BERT is a highly bidirectional state-of-the-art (SOTA) language model that has been trained on a large English Wikipedia corpus.

BERT (Bidirectional Encoder Representations from Transformers) is a powerful language model developed by Google that has been widely used for various natural language processing tasks, including sentiment analysis.

1.1 Overview of the Research

To use BERT for sentiment analysis, the text data is preprocessed to convert it into a format that BERT can understand. This typically involves tokenizing the text into individual words or subwords, and encoding each word with a unique integer ID. This preprocessed data is fed into a BERT model, which will return a fixed-length encoding for each input text.

Next, a classification model is defined that takes the BERT encodings as input and predicts the sentiment of the input text. This can be done using a fully-connected (dense) layer on top of the BERT encodings, followed by a softmax activation function to output a probability distribution over the possible classes (e.g., positive, neutral, or negative).

To train the classification model, a labeled dataset is provided that consists of text examples and their corresponding sentiments. This dataset is used to fine-tune the BERT model and the classification layer together, using an optimization algorithm such as stochastic gradient descent (SGD) or Adam.

Once the model is trained, it is used to predict the sentiment of new text examples by providing them as input to the model and interpreting the output probabilities. It's worth noting that while BERT is a very effective model for many natural language processing tasks, it may not always be the best choice for sentiment analysis. There are many other approaches and models that can be used for this task, and the choice of which one to use will depend on the specific requirements and characteristics of the problems.

1.2 Problem Statement

The success of deep learning relies on the huge amount of labeled data in many applications. Large quantities of tagged data are typically difficult or expensive to gather. Researchers have turned to transfer learning to solve this problem. Transfer learning takes into account the situation where little labeled data is from the target domain for a particular task but has many relevant tasks with a lot of data from other domains (also known as out-of-domain data). The objective is to transfer knowledge from the high-resource domains to the low-resource target domain.

The lack of resources is the main issue in resolving new NLP research in the Myanmar language, a low resource language. Analysis task of classification is labeled by domain experts and this manual labeling process is intensively expensive. Pre-trained language models can leverage large amounts of unlabeled data to learn the universal language representations, which provide an effective solution for the above problem. Pre-trained transformer-based masked language models such as BERT, RoBERTa and ALBERT had a dramatic impact on the NLP landscape in recent years. A pre-trained model is often trained on a supervised downstream dataset for a few epochs, which is known as fine-tuning, in the common recipe for using such models.

1.3 Objective of the Research

In general, the goal of sentiment analysis is to understand and classify the emotional content of a given piece of text. The main objectives of this research are

- To reduce the data requirement using pre-trained RoBERTa models
- To improve the generalization ability of the model
- To choose the best pooling approach for the fine-tuning
- To analyze the results between language specific RoBERTa model and multilingual RoBERTa model.

1.4 Contribution of the Research

In this research, the best approach is explored for classification using the pre-trained contextualized language model RoBERTa and compare the results with the language specific model and multilingual model. The proposed architecture can be used in any low resource classification problem to increase the accuracy than the traditional machine learning and data intensive deep learning approach. The training pipeline and prediction pipeline is developed.

In the training pipeline,

- Develop preprocessing, word tokenization and normalization on the dataset
- Analyze the dataset nature and visualize the data distribution
- Define the hyperparameters and training arguments
- Training and Evaluation the six different models using GPU

In the prediction pipeline,

- Deploy the backend models mapping into GPU into CPU
- Develop GUI and connect with the backend models
- Store the record into mysql database
- Show the analysis results on the GUI

CHAPTER 2

RELATED WORK

Numerous models for fine-tuning have been put forth for the sentiment analysis problem. They create two efficient information pooling strategies (1. They use an LSTM network to connect all intermediate representations of the [CLS] token, and the output of the last LSTM cell is used as the final Representation. Aspect Based Sentiment Analysis explores the possibility of utilizing intermediate layers of BERT. [10]. They employ a dot-product attention module to dynamically aggregate all intermediates, and 2. Attention Pooling, to resolve aspect-based sentiment analysis tasks. On three ABSA datasets, they demonstrate their techniques for optimizing BERT-based models. They also do trials on a significant and well-known NLI problem to show generality.

Weibo Text Sentiment Analysis, a public dataset of Weibo text collected during the COVID-19 epidemic was based on BERT and Deep Learning. [4]. The pre-train layer, the BiLSTM layer, the Attention layer, the CNN layer, and the Fully Connected layer are the five layers that make up their model. They performed their experiment using a Weibo text collection pertaining to the COVID-19 outbreak. The experimental findings for the suggested model and various rival models are shown. The competition structure can be divided into three sections. The first component is the old and existing deep learning model, which contains CNN and BiLSTM, the second stage is based on transformer models such as BERT, and the final component is the combined model.

End-to-End Aspect-Based Sentiment Analysis (E2E-ABSA) is an approach that concentrates on the aspect term-level. [9]. It is possible to describe this issue as a sequence labeling problem. Their approach uses the Design of Downstream Model, which generates linear, RNN, and CRF models, and BERT as an embedding layer. In this research, they examine how well BERT embedding components perform End-to-End Aspect-Based Sentiment Analysis (E2E-ABSA). They carry out in-depth studies on two benchmark datasets while examining how to combine the BERT embedding component with other neural models.

There are several sentiment analysis approaches in the Myanmar NLP area. Sentiment Analysis of Comments on a Public Facebook Page in Myanmar Text is one of them that uses the Word Vector Representation Techniques with Machine-Learning Classifiers[1]. The performance of three different Machine Learning (ML) techniques, including Logistic Regression, SVM, and Random Forest, is compared while using Myanmar sentiment analysis based on the word vector representation method. The dataset that was used for sentiment analysis was a collection of Myanmar Facebook page comments.

CHAPTER 3

LITERATURE STUDY

In the beginning of this research, the literature review is the most important chapter in the research, and although it turned out to be cumbersome, without the literature study further work will not be clear. With this knowledge and some experience with transformers and transfer learning, further work does not require much effort, only time to train networks. Main subjects covered here:

- Levels of Sentiment Analysis
- Sentiment Analysis Approaches
  - Rule-based Approach
  - Machine Learning Approach
  - Deep learning Approach
  - Transformer-based Approach
- Word Embedding Techniques

3.1 Levels of Sentiment Analysis

Sentiment analysis has primarily been studied at three stages. [6]. Identifying whether a whole opinion document communicates a favorable or negative mood is the primary task at the document level. This level of analysis is based on the assumption that each document expresses opinions about a single entity. The main task at the sentence level is to determine whether each sentence expresses a positive, negative or neutral opinion. This level of analysis is closely related to subjectivity classification, which distinguishes objective sentences expressing factual information from subjective sentences expressing sentiment views and opinions. Document level and sentence level analyses do not reveal what people favored and disliked. Aspect level does more precise analysis. Instead of focusing on language constructs (documents, paragraphs, sentences, clauses or phrases), aspect level examines the opinion itself.

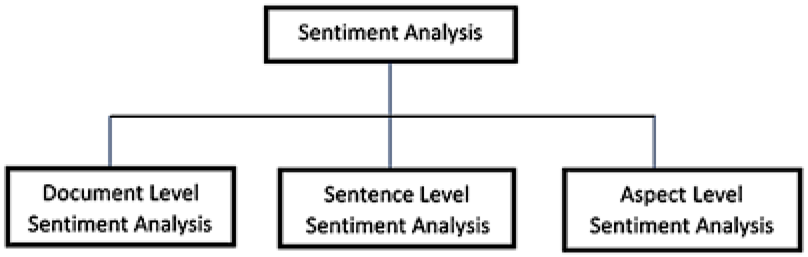

Sentiment Analysis
Document Level Sentiment Analysis
Sentence Level Sentiment Analysis
Aspect Level Sentiment Analysis

Figure 3.1 Levels of Sentiment Analysis

3.2 Sentiment Analysis Approaches

There are several approaches to sentiment analysis, which is the process of identifying and extracting subjective information from text data. Some common approaches include rule-based approach, machine learning approach, deep learning approach and transformer-based approach.

3.2.1 Rule-based Approach

To detect the sentiment of a sentence, the primary technique is rule-based and makes use of a lexicon of words labeled by sentiment. [7]. Sentiment scores should frequently be combined with additional guidelines to remove dependent clauses, negations, and sarcasm from sentences. The following NLP methods are covered under the rules:

- Stemming, tokenization, part-of-speech (POS) tagging
- Lexicons.

Since the sequential merger of words is ignored, rule-based systems are quite straightforward. To enable newer forms of expression and vocabularies, better processing techniques can be applied, and the most recent rules can be added. New rules, however, have the potential to change previously discovered outcomes and make the system as a whole quite complex. Rule-based systems necessitate continuous maintenance and fine-tuning, which necessitates financing on a regular basis.

3.2.2 Machine Learning Approach

Before using an algorithm on the real data set, machine learning algorithms train it on a training set of data. In order to eventually be able to deal with fresh, unknowable data, machine learning techniques first train the algorithm using a set of specific inputs and known outputs. The following list includes some of the most well-known machine learning-based works.

3.2.2.1 Support Vector Machine

An extensive training set is needed for this non-probabilistic classifier. Using a (d-1)-dimensional hyperplane, points are categorized. The greatest margin-possible hyperplane is found through SVM. Decision planes serve as decision boundary definitions and are used by support vector machines. A decision plane is a diagram that divides up a collection of things into several classes. The dividing line establishes the boundary between the items' classes, which are either red or green. This is a case of mapping or transformation, where the original objects are reorganized or mapped using a kernel-like mathematical function. After transformation, the mapped objects can be linearly separated, which allows for the creation of complex structures with curves to divide the objects. After transformation, the mapped items can be divided linearly, which allows for the avoidance of complex structures that require curves to do so.

3.2.2.2 Naïve Bayes Method

This probabilistic classifier is typically employed when the size of the training set is small. It belongs to the category of Bayes theorem-based example probabilistic classifiers in machine learning. By applying the Bayes rule, the equation (1) determines the conditional distribution that an event X occurs given the probability Y.

$$P(\frac{X}{Y}) = \frac{P(X)P(\frac{Y}{X})}{P(Y)} \qquad \textbf{3.1}$$

Where X is an event and Y is the evidence. So for finding the sentiment the equation is transformed into the below equation (2).

$$P(\frac{senti}{sent}) = \frac{P(senti)P(\frac{sent}{senti})}{P(sent)} \quad \textbf{3.2}$$

Where senti means sentiment and sent means sentence. P(sent/senti) is calculated as the product of P (token /senti), which is formulated by the equation (3) .

$$Count(Thistokeninclass) + \frac{1}{Count(Alltokensinclass)} + Count(Alltokens) \quad \textbf{3.3}$$

Where $Thistokeninclass$ means the tokens in one class, alltokensinclass means all the tokens in the one class and alltokens means all tokens.

3.2.2.3 Decision Tree (DT)

When non-linear data sets need to be handled effectively, another supervised learning algorithm, such as Decision Tree, is used. The decision tree method is fairly efficient at creating classification models from the data that is given. Different types of decision trees are frequently employed with logical procedures. The Decision Tree is arranged in a way that resembles a flowchart and resembles a tree. Each leaf node or terminal node serves as a representation of the class label, while the node at the decision tree's tip serves as a representation of the root node. Equation (4) shows the equation for the decision tree's entropy:

$$E(D) = n\sum i = 1 - pc(i)(pc(i)) \quad \textbf{3.4}$$

Where, $pc(i)$ is the probability of class C(i) in a node. E(D) or also called entropy of D is the measure of disorder of the considered samples.

3.2.3 Deep learning Approach

Deep learning commonly is one of many machine learning techniques built on artificial neural networks and referred to as deep structured learning. Each degree of deep learning learns how to change the incoming data into a tad more abstract and composite representation. In an application for image recognition, the initial input could be a matrix of pixels; the first representational layer could abstract the pixels and encode edges; the second layer could compose and encode arrangements of edges; the third layer could encode a nose and eyes; and the fourth layer could recognize that the image contains a face. Importantly, a deep learning phase is capable of discovering on its own which traits are most suited for each level. This does not eliminate the need for manual adjustment; for instance, different layer counts and size can offer various levels of abstraction.

3.2.3.1 Long Short-Term Memory Model (LSTM)

Recurrent Neural Network (RNN) units comprise Long Short-Term Memory Model (LSTM) units (RNN). On the basis of time series data, LSTM networks are utilized for classification or prediction. Exploding and vanishing gradient issues can be resolved. Long-term reliance is one of RNN's biggest issues. Such long-term dependency can be avoided using LSTM. A cell, input gate, output gate, and forget gate make up a standard LSTM unit. The amount of the previous sample that is saved in memory is set by the input gate. The forget gate governs the rate of loss of stored memory and establishes what data can be removed from the state of the cell, while the output gate controls the quantity of data transferred to the following layer. The input gate, forget gate, and output gate are where the weights of LSTMs are changed, preventing gradient disappearing or ballooning gradient problems. Figure 3.2 depicts how LSTM cells function.

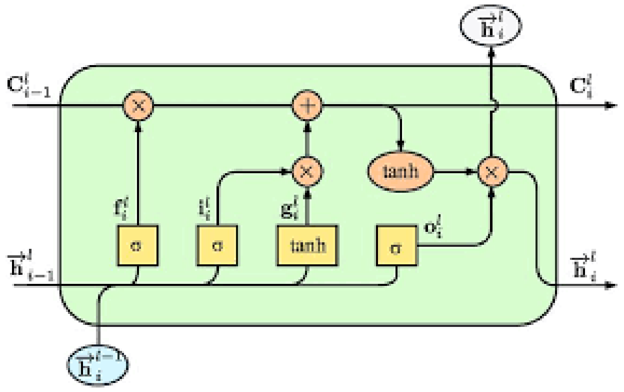


Figure 3.2 LSTM network

3.2.3.2 Gated Recurrent Unit (GRU)

A streamlined version of the Long Short Term Memory (LSTM) paradigm is the Gated Recurrent Unit (GRU). [7]. GRU can effectively capture long-term dependencies between sequences while having fewer parameters. As a result, in terms of effectiveness and computational efficiency, GRU is similar to LSTM. The two gates known as update and reset gates, which regulate the information flow via each hidden unit, are calculated by the Gated Recurrent Unit (GRU). Figure 3.3 depicts how GRU cells function.

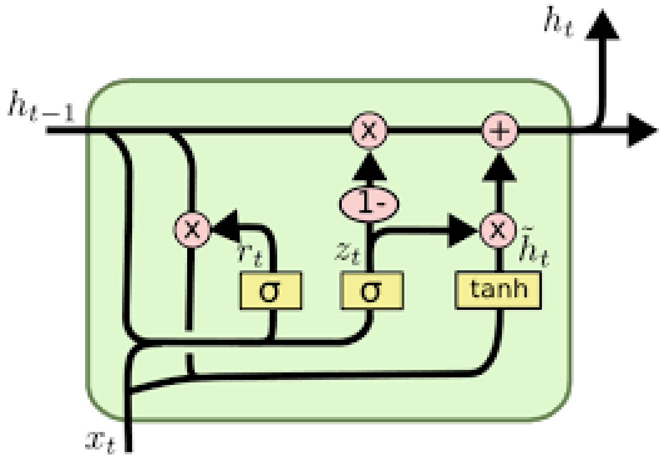


Figure 3.3 GRU network

3.2.4 Transformer-based Approach

A revolutionary architecture called NLP's Transformer aims to solve problems sequentially while resolving long-distance dependencies with ease. It uses only self-attention and does not use sequence-aligned RNNs or convolutions to compute the input and output representations.

3.2.4.1 BERT

The BERT (Bidirectional Encoder Representations from Transformers) Natural Language Processing Model was invented by Google Research researchers and released in 2018. [5]. In addition to semi-supervised training, OpenAI transformers, ELMo Embeddings, ULMFit, and Transformers are just a few of the earlier NLP methods and architectures that BERT leverages. BERT is a transformer architecture encoding stack. An encoder-decoder network using self-attention on the encoder side and attention on the decoder side is known as a transformer architecture. Pre-training and fine-tuning are the two stages of BERT. Unlabelled data is utilized to train the model via a variety of pre-training tasks during pre-training. When fine-tuning, BERT is first initialized with pre-trained parameters, then all of the parameters are adjusted using labeled data from the subsequent jobs. Despite being initialized with similar pre-trained parameters, each downstream task has its own fine-tuned models.

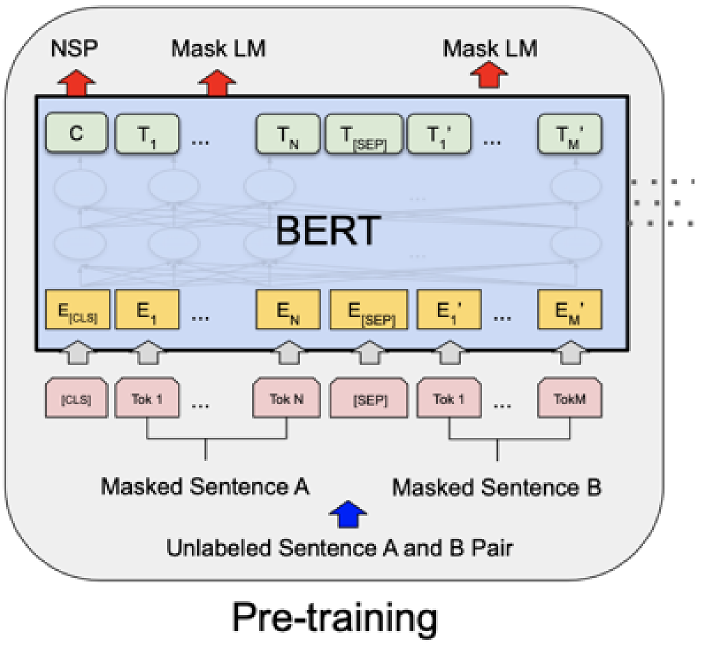


Figure 3.4 BERT Architecture

3.2.4.2 ALBERT

The ALBERT model, which was introduced in ALBERT(A Lite BERT for Self-supervised Learning of Language Representation), offers two parameter-reduction strategies to reduce memory usage and speed up BERT training:

- Creating two smaller matrices from the embedding matrix.
- Dividing up using repeating layers into groupings.

- As an absolute position embedding model, ALBERT, it is typically recommended to pad the inputs on the right rather than the left.
- ALBERT employs repeating layers, which reduces its memory footprint, but because it iterates through the same number of (repeating) levels as a BERT-like design with the same number of hidden layers, its computational cost is comparable. [11].

3.2.4.3 RoBERTa

In RoBERTa, A Robustly Optimized BERT Pretraining Approach, the RoBERTa model was put forth. [9]. It is inspired by the 2018 BERT model from Google. Although comprehensive comparison across various methods is difficult, language model pretraining has significantly improved performance. Training is a computationally intensive process that frequently uses private datasets of various sizes, and the selection of the hyperparameters has a big effect on the outcomes. They describe a replication study of BERT pretraining in which the effects of numerous important hyperparameters and the amount of the training data are extensively examined. They find that BERT was significantly undertrained, and can suit or surpass the performance of every model published after it. Their top model produces cutting-edge outcomes on GLUE, RACE, and SQuAD. These findings cast light on hitherto undervalued design considerations and cast doubt on the origin of recently reported advances.

3.3 Word Embedding Techniques

Word embeddings are numerical representations of words or phrases in a lower-dimensional space that capture the semantic and syntactic relationships between words. There are several types of word embeddings, including:

Count-based embeddings: These embeddings are based on the raw frequency counts of words in a corpus. Examples include TF-IDF.

Predictive embeddings: These embeddings are trained to predict the surrounding context of words in a corpus, using techniques such as skip-gram or continuous bag-of-words (CBOW). Examples include word2vec and FastText.

Contextual embeddings: These embeddings are trained to incorporate contextual information, such as the relationships between words in a sentence or the context of a word in a document. Examples include BERT and ELMo.

3.3.1 TF-IDF

TF-IDF (Term Frequency-Inverse Document Frequency) is a method of representing text data as a numerical vector. Each word in a document is represented by a numerical score that reflects its importance within the document. The score is calculated as the product of two factors: the term frequency (TF), which reflects the number of times the word appears in the document, and the inverse document frequency (IDF), which reflects the rarity of the word across a collection of documents.

The frequency of words in a document determines the TF score. The number of times a word appears in the files is counted. The total frequency (TF) of a word is determined by dividing its frequency I by the total number of words (N) in the document (j).

$$TF\ (i) \ = \ \frac{log\ (frequency\ (i,j))}{log\ (N\ (j))} \qquad \textbf{3.5}$$

Where TF means term frequency, i means word and j means the document.

The rarity of the terms is determined by the IDF score. It is significant since TF prioritizes terms with higher frequency of occurrence. Words that are infrequently used in the corpus, however, could contain important information. This data is gathered by the IDF. One way to determine it is to divide the total number of documents (N) by the number of documents that include the word (i).

$$IDF\ (i) \ = \frac{log\ (N\ (d)}{frequency\ (d,i))} \qquad \textbf{3.6}$$

Where IDF means inverse term frequency, d means document and i means word.

3.3.2 Word2vec

It was created in 2013 by Google researchers Tomas Mikolov and others. [8]. An advanced NLP problem-solving method called Word2Vec uses word embedding. To discover word dependencies or correlations, it can repeatedly read through a vast corpus of text.

Word2Vec uses the cosine similarity metric to identify word similarities. Words are overlapping if the cosine angle is 1. In the case of a cosine angle of 90, words are autonomous or lack contextual resemblance. Similar vector representations are given to related words by this system

Continuous Bag of Words (CBOW) and Skip-gram are two neural network-based variations that Word2Vec offers. Using a variety of words as input, the neural network model in CBOW predicts the target word that is highly related to the input words' context. The Skip-gram architecture, on the other hand, uses one word as input and predicts all of the closely associated context words.

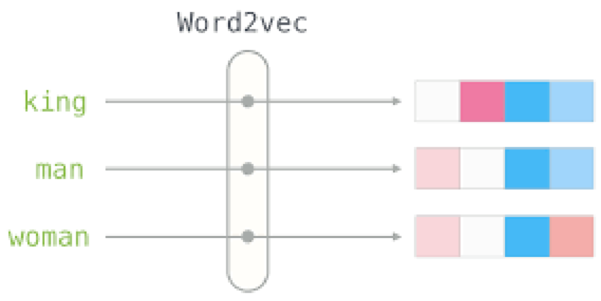


Figure 3.5 Word2vec Representation Illustration

CHAPTER 4

Methodology

Methodology Overview

Methodology Overview contains explanation of the transformer, BERT and RoBERTa architecture, description of the transfer learning, learning rate scheduler and loss function, visualization algorithms explanation of PCA and word2vec, and the system architecture.

4.1 Explanation of the Transformer, BERT and RoBERTa Architecture

The transformer neural network is a novel architecture that solves sequence-to-sequence issues while expertly handling long-range relationships. It was first proposed in the article "Attention Is All You Need," and it is currently an innovative technique in the field of NLP research [9]. This section provides an explanation of the transformer-based architecture.

4.1.1 Transformers

The Transformer architecture follows an encoder-decoder structure but does not rely on recurrence and convolutions in order to generate an output.

Encoder: The encoder is made up of N = 6 unique layers stacked together. Two sublayers make up each layer. The first is a multiple-head self-attention mechanism, and the second is a straightforward feed-forward network that is fully coupled according to position. Around each of the two sub-layers, a residual connection is used, and then layer normalization. In other words, each sub-output layer is LayerNorm(x + Sublayer(x)), where Sublayer(x) is the function that the sub-layer itself implements. All model sub-layers as well as the embedding layers generate outputs with dimension model = 512 to assist these residual connections.

Decoder: Decoder: A stack of N = 6 consecutive layers also makes up the decoder. The decoder in bert is responsible for generating the output sequence of words or tokens based on the encoded representation of the input text it does this by using a combination of attention mechanisms and multi-headed self-attention to process the encoded representation and generate the output sequence in most cases the output sequence is generated one word or token at a time starting from the first word and proceeding sequentially until the end of the sequence is reached the output sequence is generated based on the encoded representation of the input text and the previous words or tokens in the output sequence so that the output is coherent and semantically meaningful

Attention : A query and a set of key-value pairs can be mapped to an output by an attention function, where the output, the keys, and the values are all vectors. The result is calculated as a weighted sum of the values, with the weights assigned to each value determined by how well the query matches the key in question. **Multi-Head Attention** and **Scaled Dot-Product Attention** are two different kinds of attention mechanisms.

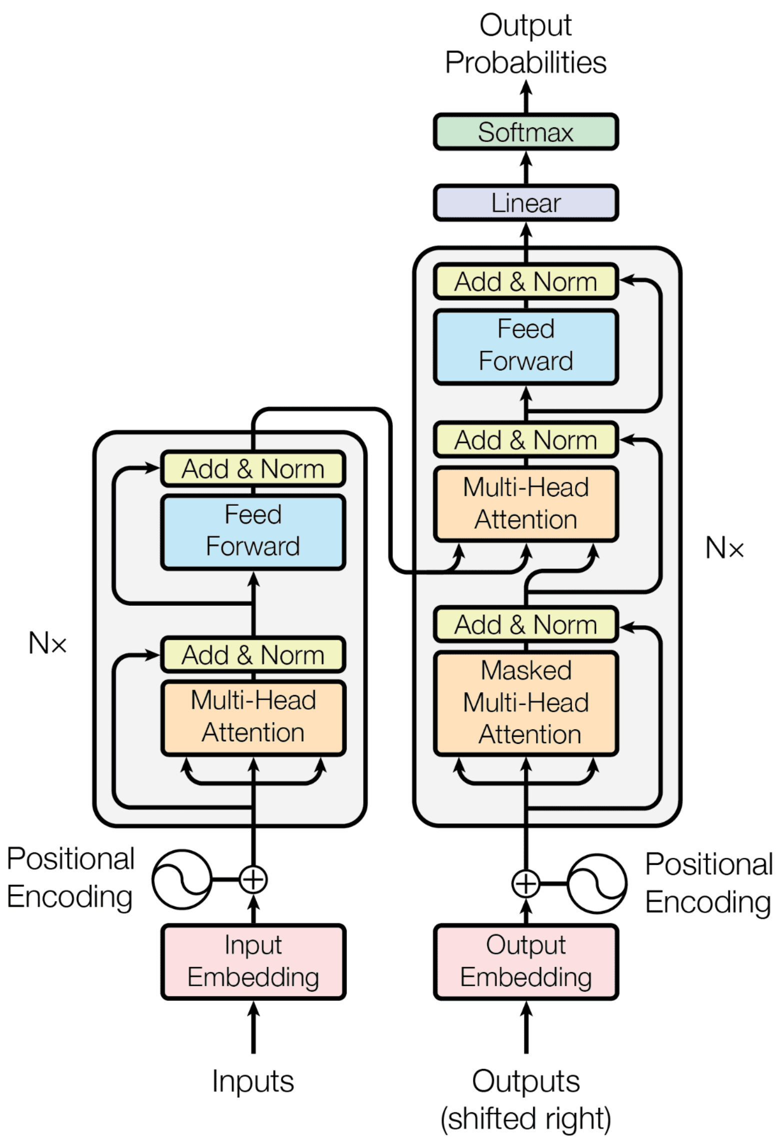


Figure 4.1 Transformer Architecture

4.1.2 BERT

By concurrently conditioning on both left and right context in all layers, Bidirectional Encoder Representations from Transformers is intended to pretrain deep bidirectional representations from unlabeled text. Two unsupervised tasks, Masked Language Modeling and Next Sentence Prediction, are used to pre-train BERT.

4.1.2.1 Masked Language Modeling

Word sequences are changed with a [MASK] token for 15% of the words in each sequence before being fed into the BERT. Based on the context offered by the other, non-masked, words in the sequence, the model then makes an attempt to forecast the original value of the masked words.

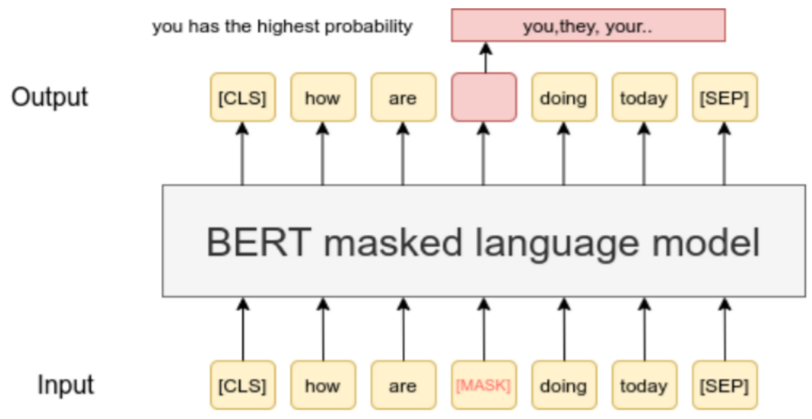


Figure 4.2 Masked LM (MLM) Approach

4.1.2.2 Next Sentence Prediction

A task known as Next Sentence Prediction (NSP) is used to train and test the effectiveness of natural language processing algorithms. The NSP challenge asks a model to determine if the second sentence in a pair is connected to or follows from the first statement. The goal of this job is to evaluate the model's comprehension of the context and relationships between phrases in a text as well as its capacity to produce coherent and semantically relevant outputs. The input is handled as follows before entering the model to aid the model in differentiating between the two sentences during training:

1. The first sentence has a [CLS] token at the start, and each subsequent sentence has a [SEP] token at the end.

2. Each token has a sentence embedding that designates Sentence A or Sentence B. Token embeddings with a vocabulary of 2 and sentence embeddings share a similar notion.
3. Each token receives a positional embedding to denote its place in the sequence. The Transformer paper presents the theory and practice of positional embedding.

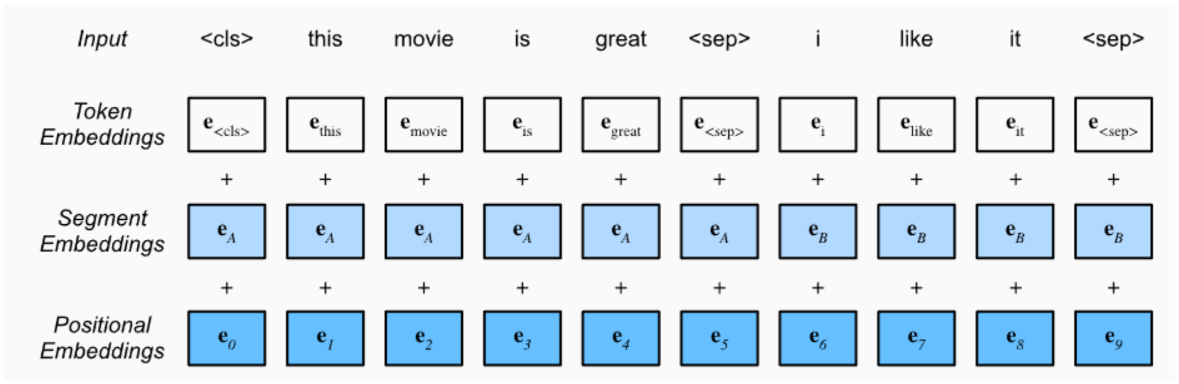

Input
<cls>
this
movie
is
great
<sep>
i
like
it
<sep>
Token Embeddings
$e_{<cls>}$ e_{this} e_{movie} e_{is} e_{great} $e_{<sep>}$ e_{i} e_{like} e_{it} $e_{<sep>}$
Segment Embeddings
e_A e_A e_A e_A e_A e_A e_B e_B e_B e_B
Positional Embeddings
e_0 e_1 e_2 e_3 e_4 e_5 e_6 e_7 e_8 e_9

Figure 4.3 Next Sentence Prediction Approach

4.1.2.3 Multi-Head Self Attention

Self-attention can encode (understand) each word based on the words that are present in the context of the current word. One word might have distinct meanings in different sentences (context).

- "The boy ate the **apple**" defines an **apple** as a fruit.
- "The boy went to **Apple**" defines **Apple** as a brand or store.
- "**ပန်းသီး** ဆို အစိမ်းမှ ကြိုက်တာ" defines an **ပန်းသီး** as a fruit.
- "ကျွန်တော်က **ပန်းသီး** မဟုတ်ရင် မကိုင်ဘူး" defines **ပန်းသီး** as a brand.

The attention mechanism refers to a weighted average of (sequence) elements, where the weights are dynamically determined using the keys of the elements and an input query.

- **Query :** The input word vector for the token is the query, which is a feature vector.
- **Keys:** Here is a key, which is also a feature vector, for each input element. This feature vector provides a general description of the element's "offering" or potential significance. Designing the keys in such a way that the element is defined to pay attention based on the query is recommended.
- **Values**: A value vector exists for each input element. The feature vector over which the average should be calculated is this one.

In math, the dot product attention is calculated as follows:

$$Attention(Q,\ K,\ V)\ =\ softmax\ (\frac{QKT}{\sqrt{dk}})V \qquad \textbf{4.1}$$

Where Q means query, K means key, V means value and d means dimension.

A network can watch over a sequence with the scaled dot product attention. However, a sequence element frequently wishes to focus on several distinct issues, so a single weighted average is not the best solution for it. In order to address numerous query-key-value triplets on the same features, or several heads, the attention methods are expanded.

$$MultiHead(Q,\ K,\ V)\ =\ Concat(head_1,\ ..,\ head_h)W^Q \qquad \textbf{4.2}$$

$$where\ head_i\ =\ Attention(QW_i^Q,\ KW_i^K,\ VW_i^V)$$

Where Q means query, K means key, V means value and i means the number of the attention head.

4.1.3 RoBERTa

RoBERTa (Robustly Optimized BERT Pre-Training Approach) includes the following steps:

- training the model longer, with bigger batches, over more data

- removing the next sentence prediction objective
- training on longer sequences and
- dynamically changing the masking pattern applied to the training data.

4.1.3.1 Static vs. Dynamic Masking

Prior to training the model, specific words or tokens in the text are hidden or obscured, and the masking is maintained during training. This implies that, regardless of the context or input that the model gets, the same words or tokens are always masked in the same way. Static masking is frequently used to assess a model's capacity to discover patterns and correlations in the data as well as its capacity to produce outputs that are coherent and have semantic significance.

Different words or tokens in a text are obscured or hidden throughout the training process with dynamic masking, and the masking might change based on the input the model gets. This implies that depending on the context or input that the model is analyzing, distinct words or tokens may be masked in various ways. Dynamic masking is frequently used to assess a model's flexibility in responding to various scenarios and settings as well as its capacity to provide consistent and semantically significant outputs under various circumstances. The decision of which type of masking to utilize depends on the individual aims, and both static and dynamic masking are frequently used in natural language processing to assess the effectiveness of machine learning models.

The initial BERT implementation only used a single static mask because masking was only done once during data preprocessing. Training data was copied 10 times, resulting in each sequence being masked 10 different ways throughout the course of the 40 training epochs. This was done to prevent utilizing the same mask for every training instance in every epoch. As a result, each training session was observed four times while wearing the same mask. We refer to this as dynamic masking.

4.2. Description of the Transfer Learning

A machine learning approach called transfer learning entails training a model on a big dataset and then utilizing the trained model as a starting point to optimize it on a smaller dataset for a related job. Transfer learning aims to enhance the model's performance on the smaller dataset by utilizing the information and patterns that the model has learnt from the bigger dataset. Transfer learning may be especially helpful when a smaller dataset is inadequate to train a model from start or when it is impossible to collect a big enough dataset for the particular purpose. The information and patterns discovered by the model on the bigger dataset may be used by applying transfer learning.

There are several different approaches to transfer learning, including fine-tuning the entire model, fine-tuning only certain layers of the model, or using the trained model as a feature extractor. The choice of which approach to use depends on the specific goals and characteristics of the dataset and the task.

Transfer learning is widely used in a variety of machine learning applications. It can be a powerful tool for improving the performance of ML models on a wide range of tasks.

Answers to the following questions are essential for the transfer learning technique.

- Which knowledge may be conveyed from the source to the target to enhance the effectiveness of the target task?
- When should a transfer be made and when shouldn't it be made so as to enhance rather than impair the performance or outcomes of the target task?
- How can one apply the information that learned from the source model to the current domain or task?

Fine-tuning: The concept of fine-tuning and transfer learning are very closely related. Transfer learning happens when someone transfers the information they learned from addressing one challenge to another that is unrelated but nonetheless challenging. For instance, the ability to identify vehicles might be used to solve the difficulty of identifying trucks.

Applying or using transfer learning requires some adjusting. Specifically, fine-tuning is the act of improving or modifying a model that has already been trained to carry out a single, specified job in order to have it carry out a second, related task. Using a previously developed and trained artificial neural network to take use of what the model has learned without having to create it from scratch, providing that the original. The following method have to choose in fine-tuning :

- Number of layers
- Types of layers
- Order of layers
- Number of nodes in each layer
- How much regularization to use
- Learning rate

4.3. Learning rate scheduler and loss function

A machine learning and statistical optimization strategy that selects the step size at each iteration while attempting to minimize a loss function using the learning rate as a tuning parameter.

4.3.1 Learning Rate Scheduler

The amount of change or update to model weights throughout the backpropagation training process is described as the learning rate (or step-size). The learning rate is often defined as a positive value less than 1.0 as a customizable hyperparameter.

The algorithm learns quickly when the learning rate is high (on the right), but it also runs the risk of oscillating or perhaps jumping over the minima. Even worse, a high learning rate results in huge weight changes, which could lead to an overflow of the weights.

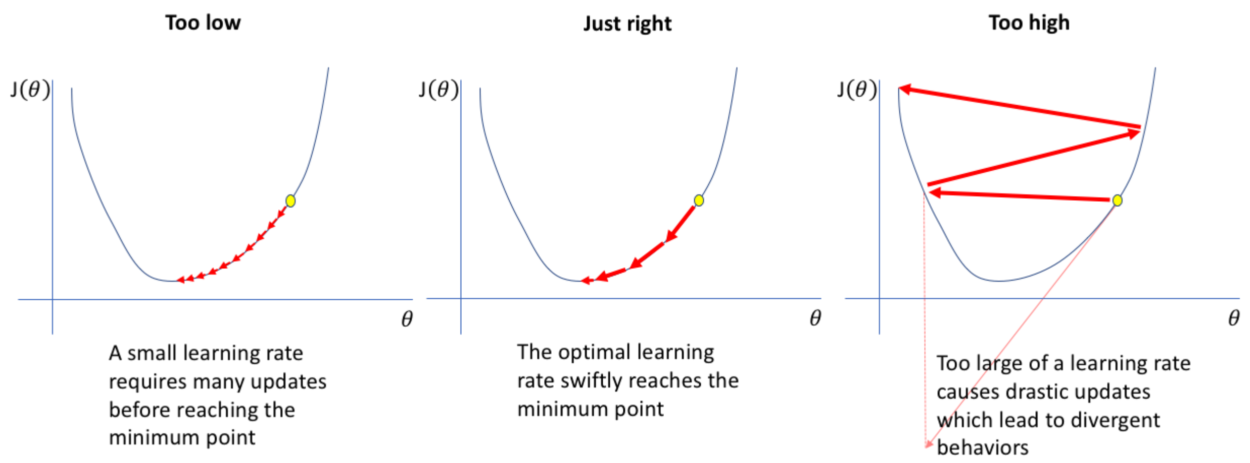

Too low
Just right
Too high
J(θ)
θ
A small learning rate requires many updates before reaching the minimum point
The optimal learning rate swiftly reaches the minimum point
Too large of a learning rate causes drastic updates which lead to divergent behaviors

Figure 4.4 Understanding the optimal learning rate

On the other hand, a lower learning rate (on the left) results in small updates to the weights, which will gradually lead the optimizer to the minima. However, the optimizer could take too long to converge or could become trapped in an unfavorable local minima or plateau.

A good learning rate strikes a balance between overshooting and coverage rate (in the middle). It is just the right size to allow the method to converge quickly while also preventing backtracking and skipping over the minima.

As the training develops, a learning rate schedule is a predetermined framework that modifies the learning rate between epochs or iterations. The two most popular methods for learning a rate schedule are:

- **Constant learning rate**: as the name suggests, during training, a learning rate is initialized and remains constant,refers to the process of choosing an initial learning rate and then gradually lowering it in line with a scheduler.
- After a warm-up period during which it climbs linearly from 0 to the initial lr set in the optimizer, the **Linear Learning Rate Scheduler** generates a schedule with a learning rate that falls linearly from the initial lr set in the optimizer to 0.

4.3.2 Loss Function

Cross-entropy loss, also described as log loss, is a measure of how well a ML model performs when its outcome is a probability value between one and zero. It occurs when the predicted probability deviates from the labeled probability. Making a forecast with a probability of 0.12 when the observation label is really 1 would be bad and have a large loss value because cross-entropy loss develops. An ideal model would have a log loss of 0.

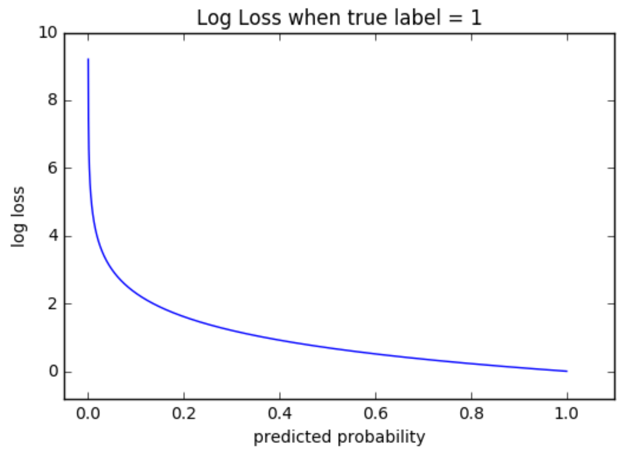


Figure 4.5 Cross-entropy loss

4.4. Visualization algorithms explanation of PCA and word2vec

To make data easier for the human brain to understand and draw conclusions from data visualization is the process of providing information into a visual context such a map or graph data visualizations major goal is to make it clearer to see patterns trends and outliers in big data sets in order to examine the nature of the class over the sentences the distribution of sentences is visualized with PCA and word2vec in this part.

4.4.1 Principal Component Analysis (PCA)

Reducing the dimension is a standard method, and PCA, an unsupervised learning algorithm, is frequently employed for this. Data visualization is also crucial, and when data has multiple dimensions, it is much harder to store vast amounts of data, run computations, and visualize. Dimensions are therefore decreased. The real goal is to make the model simpler and less complex while maintaining its usefulness and performance. Measures for PCA:

- Data can be standardized or normalized as a way to modify numbers to a common or defined scale. The "process" involves dividing by the range of values (max-min) and removing each value from the mean of that characteristic. When the standard deviation is used as the denominator, that is standardization.
- Get the covariance matrix. If it's positive, the two variables increase or decrease together (correlated). If it's negative, one variable rises as the other falls (Inversely correlated).
- Find the Eigenvalue and Eigenvector from the covariance or correlation matrix.
- Get the index by sorting the Eigenvalue now in decreasing order.
- Sorting Eigenvalue and Eigenvector simultaneously
- To convert the sorted Eigenvector matrix, use the n components dimension.
- Are a product of Equivalent Eigenvector Matrix Transpose with n Component Mean normalized matrix with a resultant transposition.

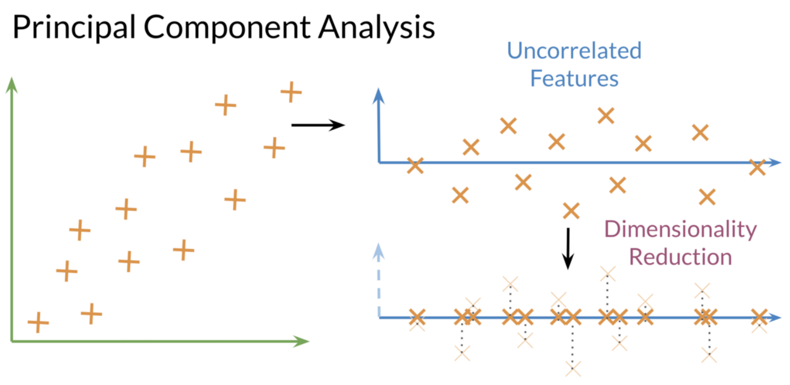


Figure 4.6 PCA Visualization

4.4.2 Word2vec

Individual words are converted into a numerical representation of the word using a process called word embeddings (a vector). Each word is converted into a single vector, which is then trained in a manner resembling a neural network. The vectors make an effort to depict multiple aspects of the word in relation to the whole text. The word semantic relationship, definitions, context, and other elements can be included in these characteristics. These numerical representations make it possible to determine how similar or dissimilar two words are.

The ability of Word2Vec to assemble vectors of related words gives it its effectiveness. Word2Vec can provide accurate predictions about a word's meaning based on its usage in the text when given a sizable enough dataset. With other words in the corpus, these estimations produce word associations. Words like "King" and "Queen," for instance, sound quite similar to one another. One can find a good approximation of word similarity by performing algebraic operations on word embeddings. As an illustration, the vector produced by multiplying the two-dimensional embedding vectors of "king," "man," and "woman" is extremely similar to the vector produced by the two-dimensional embedding vector of "queen." Please take note that the values below were selected at random.

4.5 Pooling Approaches

Main contributions of this research can be summarized as follows:

- Two pooling approaches - LSTM pooling and weighted pooling
- Two pre-trained models - language specific MyanBERTa and multilingual XLM-RoBERTa-base

4.5.1 Baseline approach

The baseline approach utilizes the CLS token of the final layer.

4.5.2 LSTM pooling approach

Depiction of hidden states An abstract-to-specific sequence, hCLS is distinct from other sequences. All intermediate representations of the [CLS] token are connected using the LSTM network, which is inherently suited to processing sequential information. The output of the final LSTM cell is then used as the final representation.

$$o = h^{L}_{\text{LSTM}} = \overrightarrow{\text{LSTM}}(h^{i}_{\text{CLS}}), i \in [1, L] \quad \textbf{4.3}$$

Where CLS means the first token of the sentence and L means the layer of the RoBERTa model.

4.5.2.1 LSTM Architecture

Long short-term memory networks, or LSTMs, are employed in deep learning. Many recurrent neural networks (RNNs) are able to learn long-term dependencies, particularly in tasks involving sequence prediction. Aside from singular data points like photos, LSTM has feedback connections, making it capable of processing the complete sequence of data. This has uses in machine translation and speech recognition, among others. A unique version of RNN called LSTM exhibits exceptional performance on a wide range of issues.

A memory cell known as a "cell state" that preserves its state over time plays a key role in an LSTM model. The horizontal line that passes through the top of the diagram below represents the cell state. It can be pictured as a conveyor belt across which data simply and unaltered passes.

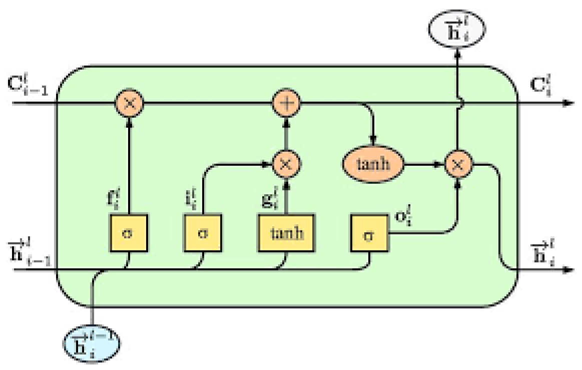


Figure 4.7. LSTM Architecture

Gates in LSTMs control the addition and removal of information from the cell state. These gates may allow information to enter and exit the cell. It has a sigmoid neural network layer and a pointwise multiplication operation that help the mechanism.
Between zero and one, the sigmoid layer outputs a number, with zero denoting "nothing should be let through" and one denoting "everything should be let through."

4.5.3 Weighted pooling approach

Intuitively, attention operations can learn the contribution of each CLS. A weighted average module is used to dynamically combine the last four intermediates.

$$\text{Weighted Average} = \frac{\sum_{i=1}^{n} X_i w_i}{\sum_{i=1}^{n} w_i} \tag{4.4}$$

Where i means the layer of the RoBERTa, X means the weight of the RoBERTa and w means the weighted sum of the vector.
Finally, the pooled output o is passed to a fully connected layer for label prediction:

$$y = softmax(W_o^T o + b_o) \quad \textbf{4.5}$$

Where o means the pooled output o to a fully connected layer.

4.6 The System Architecture

RoBERTa is capable of being utilized as a feature extraction model, however the fine-tuning method is preferred. Three alternative pooling techniques are used to enlarge the XLM-RoBERTa and MyanBERTa in order to compare them.

Each word's contextualized embedding vectors produced by RoBERTa are fed through one of the poolers; the architecture with the baseline pooler is shown in Figure 4.8. The weighted pooler for each word from the intermediary layer is calculated to create the feature vector. The final reduced vector is then classified using softmax. Connecting the intermediate representations created by RoBERTa's Transformers is the responsibility of the pooling module.

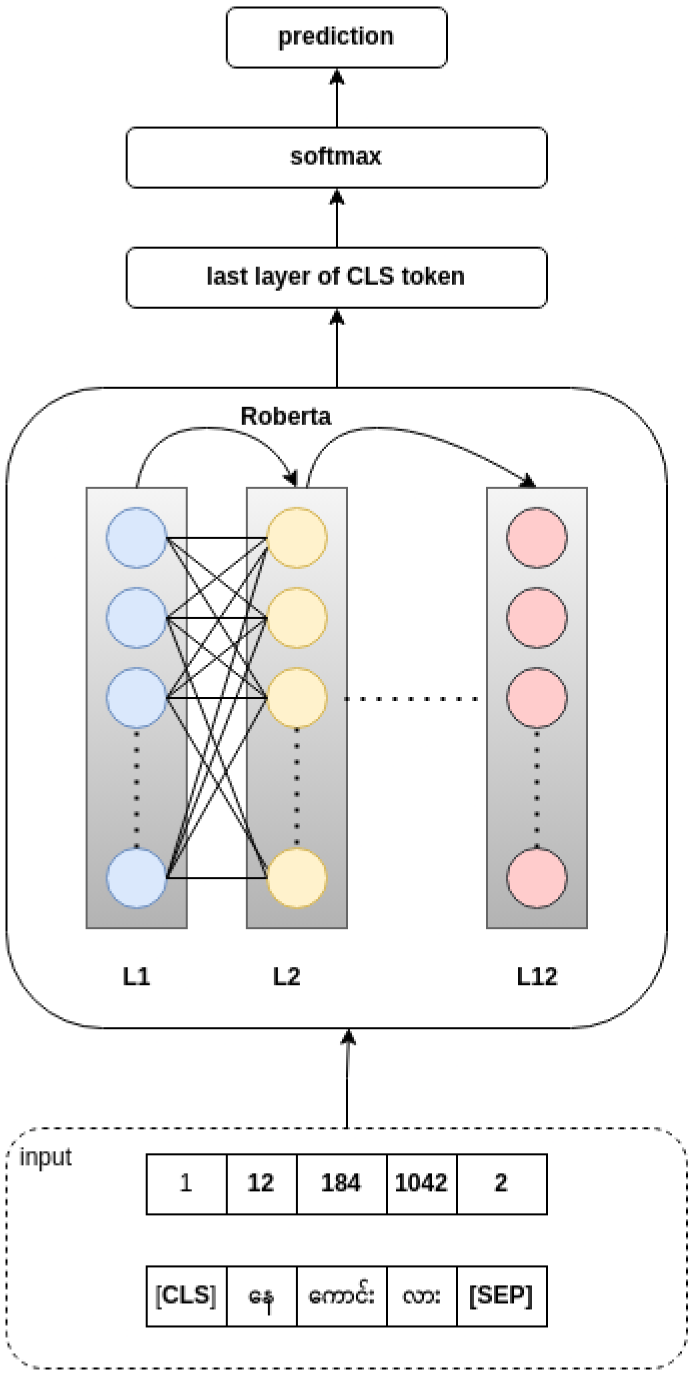


Figure 4.8 Baseline Pooler Architecture

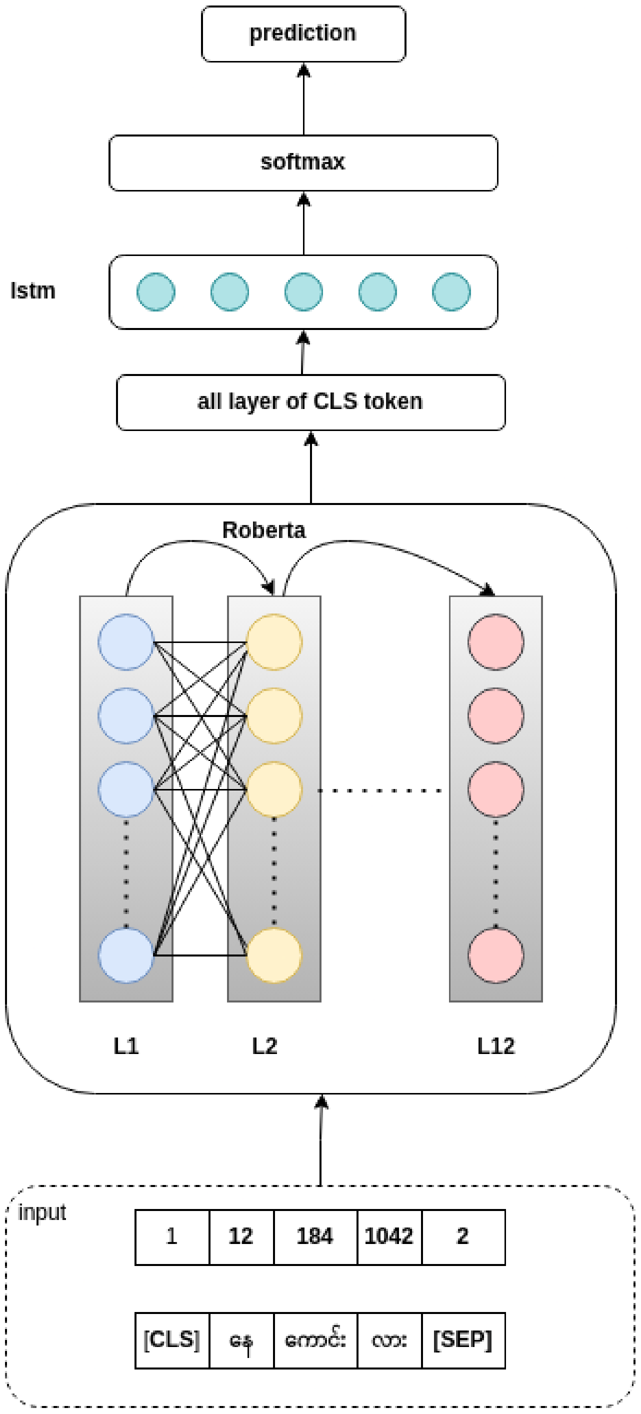


Figure 4.9 LSTM Pooler Architecture

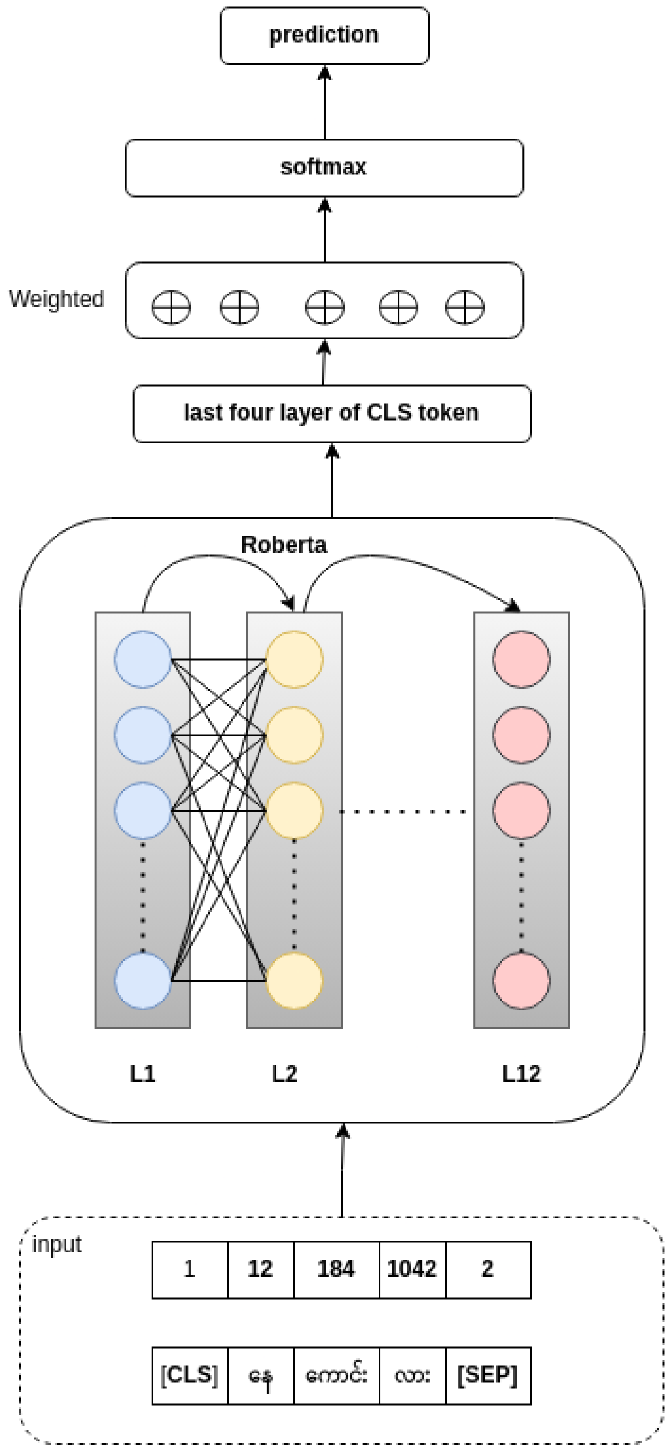


Figure 4.10 Weighted Pooler Architecture

CHAPTER 5

EXPERIMENTS AND RESULTS

The activities that constitute the RoBERTa representation and pooling approaches analysis for Myanmar Sentiment Dataset are described in this chapter. It contains the preprocessing, experimental setup and the results.

5.1 Experiments

The methods for RoBERTa based model fine-tuning on the datasets are presented. To show the generality, the experiments are also conducted on the Myanmar Language Specific model and the Multilingual Model.

5.1.1 DataSet

The first phases in this categorization job were data collection and labeling. The social media data was meticulously tagged. The manual labeling was approved by a subject matter expert. The information was gathered from the Facebook comments on the Myanmar Celebrity page. The positive, neutral, and negative tags are manually added to this dataset after it has been crawled. The sample of the sentiment dataset is shown Table 5.1. The dataset contains 72K.

Table 5.1 Sample sentences before preprocessing

Sentence	**label**
ဖြူစင်တယ် \ufeff စကားပြောကြည့် တာနဲ့သိတယ်	pos
အောင်မလေး ချစ်စရာ\u202dလေး😙	pos
ကြော်ငြာ•လေး လျှော့ပေးပါ လား	neutral
စာလုံးပေါင်း မှန်အောင်ရေး ပါ အစ်မ :')	neutral
ကိုယ့်ထမင်းကိုယ်စားပြီး သူများဝေဖန်\u200bနေ တာမင်းတို့ရုပ်တွေအရင်မှန်ထပြန်ကြည့်အလကားအကုသိုလ်ကောင်တွေ	negative
စောက်ရူးကောင်\u200b ကြောင်တောင်တောင် နဲ့	negative
ကျက်သရေအဖြာဖြာတုံးတယ်🙂	negative

5.1.1.1 Data Preprocessing

In machine learning based classification, data preparation is crucial since the model's performance is highly reliant on the quality of the input data. The following steps are used for the **data preprocessing** :

1. Dropping the None row
2. Deleting duplicate row based on the sentence column
3. Removing white-space character
4. Removing emoji
5. Deleting link
6. Cleaning the punctuation
7. Removing the numeric sentence [like phone number]
8. Detect and change into the unicode
9. And finally Deleting duplicate row based on the preprocessed sentence column

Table 5.2 Sample sentences after preprocessing

Sentence	**label**
ဖြူစင်တယ်စကားပြောကြည့်တာနဲ့သိတယ်	positive
အောင်မလေးချစ်စရာလေး	positive
ကြော်ငြာလေးလျှော့ပေးပါလား	neutral
စာလုံးပေါင်းမှန်အောင်ရေးပါအစ်မ	neutral
ကိုယ့်ထမင်းကိုယ်စားပြီးသူများဝေဖန်နေတာမင်းတို့ရုပ်တွေအရင်မှန်ထဲပြန်ကြည့် အလကားအကုသိုလ်ကောင်တွေ	negative
စောက်ရူးကောင်ကြောင်တောင်တောင်နဲ့	negative
ကျက်သရေအဖြာဖြာတုံးတယ်	negative

5.1.1.2 Word Tokenization

The trained word break model is used to tokenize the preprocessed sentences. The word break model is trained with word2vec, LSTM and CRF. This is an in-house model and the accuracy is 90%. In order to train and distribute cutting-edge sequence labeling, text classification, and language models, FLAIR, an NLP framework, is employed. Additionally, the system employs common model training and hyperparameter selection procedures.

Before word tokenization	-	ဖြူစင်တယ်စကားပြောကြည့်တာနဲ့သိတယ်
After word tokenization	-	ဖြူစင်, တယ်, စကားပြော, ကြည့်, တာ, နဲ့, သိ, တယ်

Figure 5.1 A Sample for Word Tokenization

5.1.1.3 Name Normalization

The trained NER model is used to get the person name from the tokenized sentences. The NER model is trained with word2vec, flair, LSTM and CRF. This is an in-house model and the accuracy is 95%. For the NER model, the same open-source framework FLAIR is used as the word break model. A "model zoo" of pre-trained models is also included with FLAIR so that academics may employ cutting-edge NLP models in their applications. First, the NER is parsed to obtain the person's name, and then, for the sake of normalization, the name is substituted with the "NAME" tag.

Before name normalization	-	အမြန်ဆုံး နေကောင်း ပါစေ ကိုစိုင်း
After name normalization	-	အမြန်ဆုံး နေကောင်း ပါစေ NAME

Figure 5.2 A Sample for Name Normalization

5.1.1.4 Data Analysis

Data analysis is the process to drive intuition and begin to formulate testable hypotheses. This process typically makes use of descriptive statistics and visualizations.

DataSet Class Visualization

The class frequency is plotted with the seaborn module to get better understanding of data. The number of samples can be looked for each class. If the sample of one class is considerably higher than the rest, then the model will learn to predict that class more often than others and hence leading to overfitting. The distribution shows the dataset has the normal distribution of the Myanmar Celebrity Facebook Page Comments. The dataset contains 72K sentences.

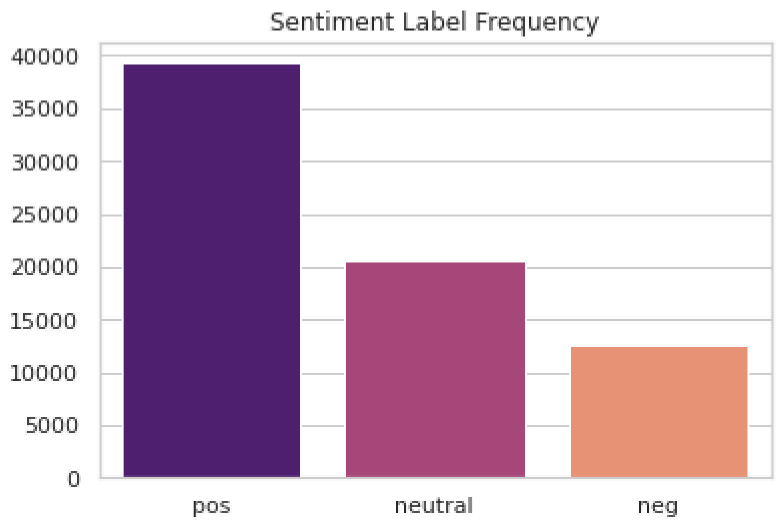


Figure 5.3 Sentiment Class Frequency

DataSet Sentence Visualization

The nature of the dataset distribution over class .i.e. negative , positive and neutral. One approach is selected to visualize the data distribution. The techniques employed are PCA and word2vec embedding. Principal component analysis is a statistical approach which allows us to compress the information content in huge data tables by using a smaller collection of "summary indices" which are simpler to show and analyze. The underlying data may consist of measurements specifying the 2-dimensional and 3-dimensional characteristics of the classes.

The three classes are each represented by a different 1K phrase. To comprehend the nature of the data, the entire dataset of 3K is used. With the use of a neural network model, the word2vec technique (context-independent word embeddings) learns word associations from a substantial body of text. PCA serves as a dimension reduction technique and serves as the foundation for multivariate data analysis based on projection methods. In order to identify trends, leaps, clusters, and outliers, PCA's most crucial use is to summarize multivariate data into a smaller number of variables (summary indices). This overview may reveal the connections between the variables in the context-independent word embeddings and the observations.

To analyze the nature of the dataset, word2vec and PCA (Principal component analysis) is used. The 1K sentences are selected from each class and the amount of information from word2vec 100 dimension feature space reduces with PCA components 3. The classes are separated with colors; yellow as positive, green as negative and purple as neutral.

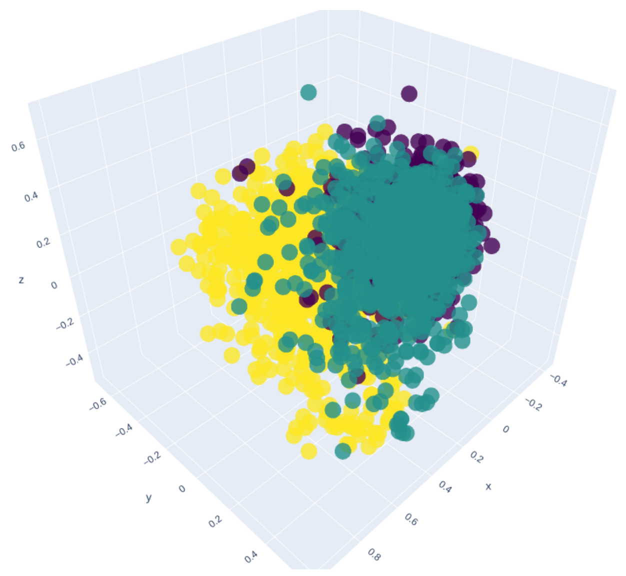


Figure 5.4 Visualization with Word2vec embedding into PCA over the selected sentiment dataset

5.2 Representation from the Pretrained Model

The data set is ready for training after preprocessing, word tokenization, and normalization. Two primary processes are present in the pre-trained RoBERTa model. Tokenization is the initial step, followed by transformation. The results from the tokenizer are shown in Figure 5.5 and Figure 5.6. The data is transformed into the format that the RoBERTa model requires in order to function. The output representation from the model, which is part of the pre-trained RoBERTa model, is presented in Figures 5.7.

Table 5.3 The Explanation of RoBERTa Default Token

Token	**Index**	**Use**
<s>	0	Beginning of sequence (BOS) or classifier (CLS) token
</s>	2	End of sequence (EOS) or separator (SEP) token
<unk>	1	Unknown token
<pad>	3	Padding token

```
Tokenizer Result : MyanBERTa
________________________________________________________________
Example sentence :
 အောင်မြင် ပါစေ သမီး ပြော တဲ့ အတိုင်း သာ လုပ် အောင်မြင် ရ မယ် အားပေး တယ်

After parsing MyanBERTa tokenizer :
 ['áĢ¡', 'áĢ±áĢ¬', 'áĢĦ', 'áĢº', 'áĢĻ', 'áĢ¼', 'áĢĦ', 'áĢº', 'ĠáĢķ', 'áĢ«', 'áĢħ', 'áĢ±', 'ĠáĢŀáĢĻ', 'áĢ®áĢ¸', 'ĠáĢķ', 'áĢ¼áĢ±áĢ¬', 'ĠáĢIJ', 'áĢ²áĢ·', 'ĠáĢ¡áĢIJ', 'áĢŃáĢ¯', 'áĢĦ', 'áĢºáĢ¸', 'ĠáĢŀ', 'áĢ¬', 'ĠáĢľ', 'áĢ¯', 'áĢķ', 'áĢº', 'ĠáĢ¡', 'áĢ±áĢ¬', 'áĢĦ', 'áĢº', 'áĢĻ', 'áĢ¼', 'áĢĦ', 'áĢº', 'ĠáĢĽ', 'ĠáĢĻáĢļ', 'áĢº', 'ĠáĢ¡', 'áĢ¬áĢ¸', 'áĢķ', 'áĢ±áĢ¸', 'ĠáĢIJáĢļ', 'áĢº']

After changing decoded token into Myanmar characters :
 ['အ', 'ော', 'င', '်', 'မ', 'ြ', 'င', '်', ' ပ', 'ါ', 'စ', 'ေ', ' သမ', 'ီး', ' ပ', 'ြော', ' တ', 'ဲ့', ' အတ', 'ို', 'င', '်း', ' သ', 'ာ', ' လ', 'ု', 'ပ', '်', ' အ', 'ော', 'င', '်', 'မ', 'ြ', 'င', '်', ' ရ', ' မယ', '်', ' အ', 'ား', 'ပ', 'ေး', ' တယ', '်']

After padding token into index :
 [0, 317, 277, 269, 263, 287, 272, 269, 263, 292, 303, 288, 267, 437, 313, 292, 326, 282, 309, 353, 271, 269, 275, 293, 265, 297, 266, 283, 263, 284, 277, 269, 263, 287, 272, 269, 263, 294, 434, 263, 284, 291, 283, 306, 347, 263, 2]
```

Figure 5.5 The transformed process of the word into index using MyanBERTa

```
Tokenizer Result : XLM-RoBERTa-base

Example sentence :
 အောင်မြင် ပါစေ သမီး ပြော တဲ့ အတိုင်း သာ လုပ် အောင်မြင် ရ မယ် အားပေး တယ်

After parsing XLM-RoBERTa-base tokenizer :
 ['_', 'အောင်မြင်', '_ပါ', 'စေ', '_', 'သမီး', '_', 'ပြော', '_', 'တဲ့', '_', 'အတိုင်း', '_', 'သာ', '_လုပ်', '_', 'အောင်မြင်',
'_ရ', '_', 'မယ်', '_အား', 'ပေး', '_', 'တယ်']

After changing decoded token into Myanmar characters :
 ['', 'အောင်မြင်', 'ပါ', 'စေ', '', 'သမီး', '', 'ပြော', '', 'တဲ့', '', 'အတိုင်း', '', 'သာ', 'လုပ်', '', 'အောင်မြင်', 'ရ', '', 'မယ်',
'အား', 'ပေး', '', 'တယ်']

After padding token into index :
 [0, 6, 205907, 40843, 31782, 6, 53156, 6, 37079, 6, 4674, 6, 154421, 6, 9806, 51861, 6, 205907, 21035, 6,
82701, 59057, 12077, 6, 38258, 2]
```

Figure 5.6 The Transformed Process of the Word into Index using XLM-RoBERTa

The RoBERTa model requires two tensors, therefore the next step can be to create them.

- input ids: our token ids with 15% of the tokens masking with mask> token.
- The position of "actual" tokens/padding labels—token ids without masking—are indicated by the attention mask, a tensor of 1s and 0s.

```
tensor([[[ 0.3354,  1.4686, -0.2191,  ..., -0.0976,  1.2770, -1.3788],
         [-1.0617,  1.0412,  0.3350,  ...,  1.5748,  2.0985,  1.0544],
         [-0.4626,  1.2569, -1.0778,  ..., -0.9197, -0.9546, -1.2346],
         ...,
         [ 0.2952,  0.3133, -0.6785,  ..., -1.7023,  0.5148,  1.3101],
         [-1.6553,  0.5398, -1.0872,  ..., -0.6277, -0.7761,  0.0071],
         [-0.0169,  0.7438,  0.4398,  ...,  0.3911,  0.0814, -0.3744]]],
       grad_fn=<NativeLayerNormBackward0>)
```

Figure 5.7 The Representation from the Pre-trained Model

5.3 Experiment Setting

In experiment setting, the overview design of the system, the hyperparameters of the models and the training hyperparameters are described.

5.3.1 System Design

After data collection and tagging, the data underwent pre-processing that included feature extraction, dimension reduction, noise removal, missing value removal, and spelling correction. The entirety methodology of this research is shown in Figure 5.8. The data set was then split into training and test parts using an 8:2 ratio.

The model is trained and evaluated using supervised learning. The trained model was then fed the testing data, and the prediction accuracy was compared to the ground truth. The whole methodology of this work has been depicted in Figure 5.8. All experiments are conducted with MyanBERTa and XLM-RoBERTa-base with different pooling strategies. In order to compare the different models, the dataset is divided 20% on a test set and the same training set and testing set is used for all experiments. For training parameters, Adam optimizer and linear learning rate scheduler are set up.

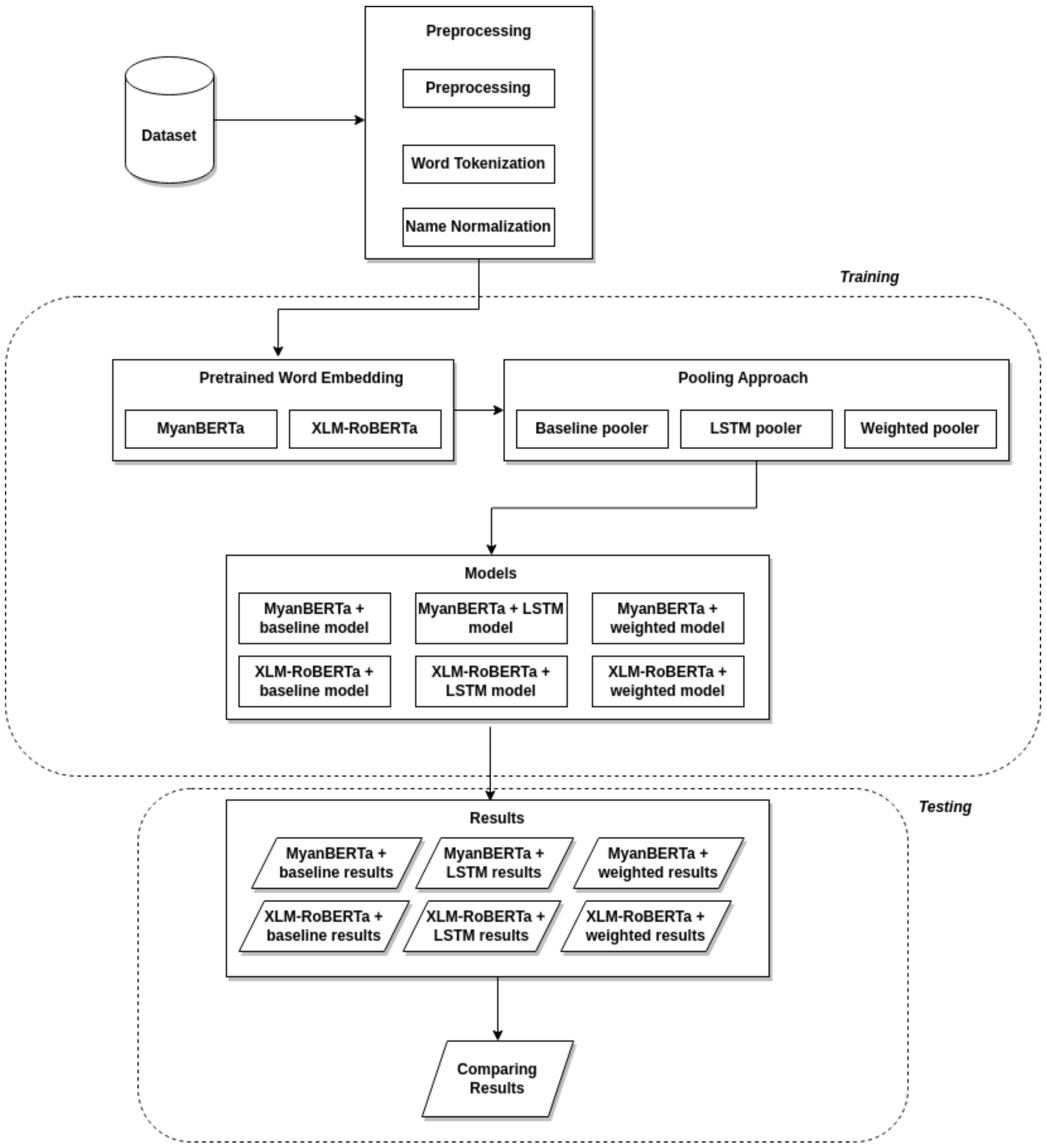

Dataset
Preprocessing
Preprocessing
Word Tokenization
Name Normalization
Training
Pretrained Word Embedding
MyanBERTa
XLM-RoBERTa
Pooling Approach
Baseline pooler
LSTM pooler
Weighted pooler
Models
MyanBERTa + baseline model
MyanBERTa + LSTM model
MyanBERTa + weighted model
XLM-RoBERTa + baseline model
XLM-RoBERTa + LSTM model
XLM-RoBERTa + weighted model
Testing
Results
MyanBERTa + baseline results
MyanBERTa + LSTM results
MyanBERTa + weighted results
XLM-RoBERTa + baseline results
XLM-RoBERTa + LSTM results
XLM-RoBERTa + weighted results
Comparing Results

Figure 5.8 The Overview Design of the System

5.3.2 Experimental Setup

According to the research, the six models are created during the model training procedure. Only the upper-level models introduced for a particular job are learned during the first phase, during which the parameters of the pre-training model are fixed. The same hyperparameters used with the pretrained models are used for fine tweaking. Table 5.3 first displays the two RoBERTa model hyperparameters. The huge training process' hyperparameters were then configured.

The hyperparameters for MyanBERTa and XLM-RoBERTa are displayed in Table 5.4. The same layers, hidden size, attention heads, dropout, and learning rate exist in both models. The batch size, dataset size, and number of languages are the main distinctions between XLM-RoBERTa and MyanBERTa.

Table 5.4 The Hyperparameters of the two RoBERTa Model

Hyperparameters	**MyanBERTa**	**XLM-RoBERTa-base**
Number of Layers	12	12
Hidden Size	768	768
Attention Heads	12	12
Dropout	0.1	0.1
Attention Dropout	0.1	0.1
Batch Size	8	8192
Learning Rate Decay	1e-5	1e-5
Dataset Size	2.1G	2.5T
Number of Languages	1	100

The experimental setting is set up before training the model in order to obtain reliable results from the model comparison. There are three pooling algorithms and two pretrained models in Table 5.5. Depending on the pooler, different layers are employed, and the LSTM pooling strategy only uses all of the embedding model's layers. The weighted pooling approach uses the final four layers, with the final layer's CLS embedding serving as the baseline.

Table 5.5 The Parameters of the Six Models

Pretrained Model	Model Type	Layers	Optimizer	Scheduler	Learning Rate	Epochs	loss function
myanBERTa	baseline	last	Adam	linear	2.00E-05	5	Cross Entropy
	lstm pooling	all	Adam	linear	2.00E-05	5	Cross Entropy
	weighted pooling	4 layers	Adam	linear	2.00E-05	5	Cross Entropy
xlm-RoBERTa-base	baseline	last	Adam	linear	2.00E-05	5	Cross Entropy
	lstm pooling	all	Adam	linear	2.00E-05	5	Cross Entropy
	weighted pooling	4 layers	Adam	linear	2.00E-05	5	Cross Entropy

Adam is chosen as the optimizer and linear scheduler to modify the training's learning rate. The five epochs and cross entropy are chosen as the loss function based on the type of fine-tuning. At the first and last epochs of the training process, the representation weight is saved into the files. The confusion matrices and evaluation report are saved for later analysis.

5.4 Experiment Results

In order to visualize how different pooling strategies benefit from sequential representations of final layers. The weight of the training dataset is saved. The principal component analysis (PCA) is used to visualize the intermediate representations of the [CLS] token to show how the different pooling strategies benefit from sequential representations of intermediate layers. Because the task-specific information is primarily extracted from the last layers of RoBERTa, the layer is depicted that gets from the pooler. The initial epoch and final epoch are compared to make it visible the effect of the transfer learning ability with a small amount of epoch. It is simple to conclude that the model divides distribution into three sentiment classes. The following figures are the trained model weight comparison for first epoch 1 and last epoch 5.

In Figure 5.9, MyanBERTa and baseline approach does not well separate the training dataset into the sentiment classes in epoch 1. But the model well clusters the training instance into the labels at epoch 5 the right side of the figure.

The advantage of the LSTM is shown in Figure 5.10, the distribution is slightly dense into the classes in epoch1. The LSTM layer learns the RoBERTa embedding to separate the label at the first epoch. At the final epoch 5, the model obviously discriminates the instance into the classes.

The weighted model tells us another assumption. In Figure 5.11, the initial epoch is not special but the distribution at the epoch 5 shows us the model learns the label without being noticeably dense into the cluster.

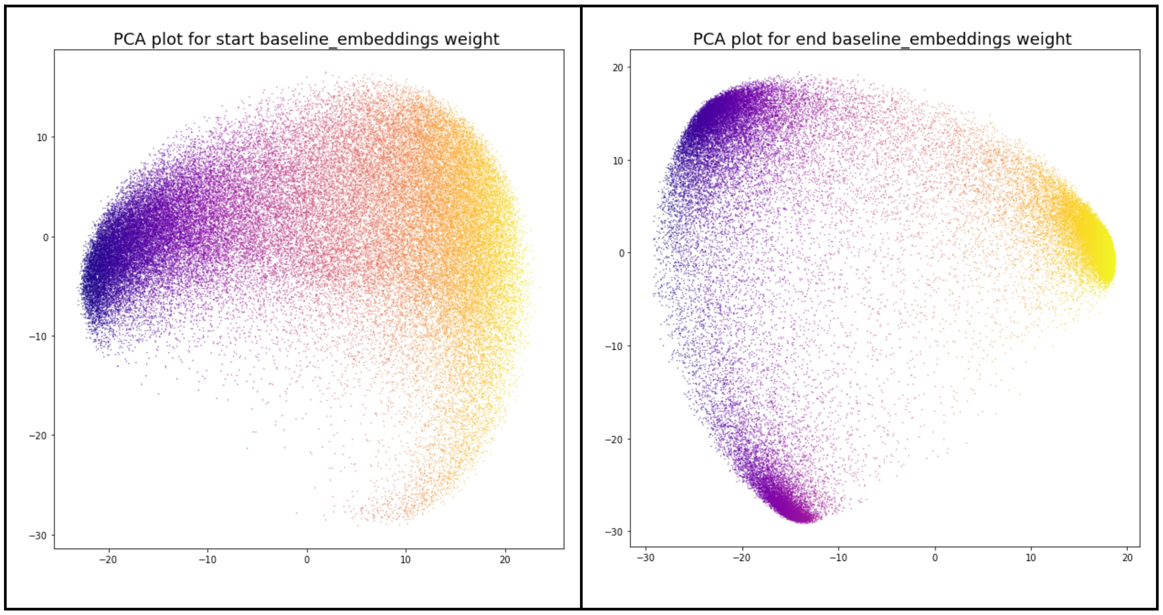


Figure 5.9 MyanBERTa + Baseline Pooling Approach

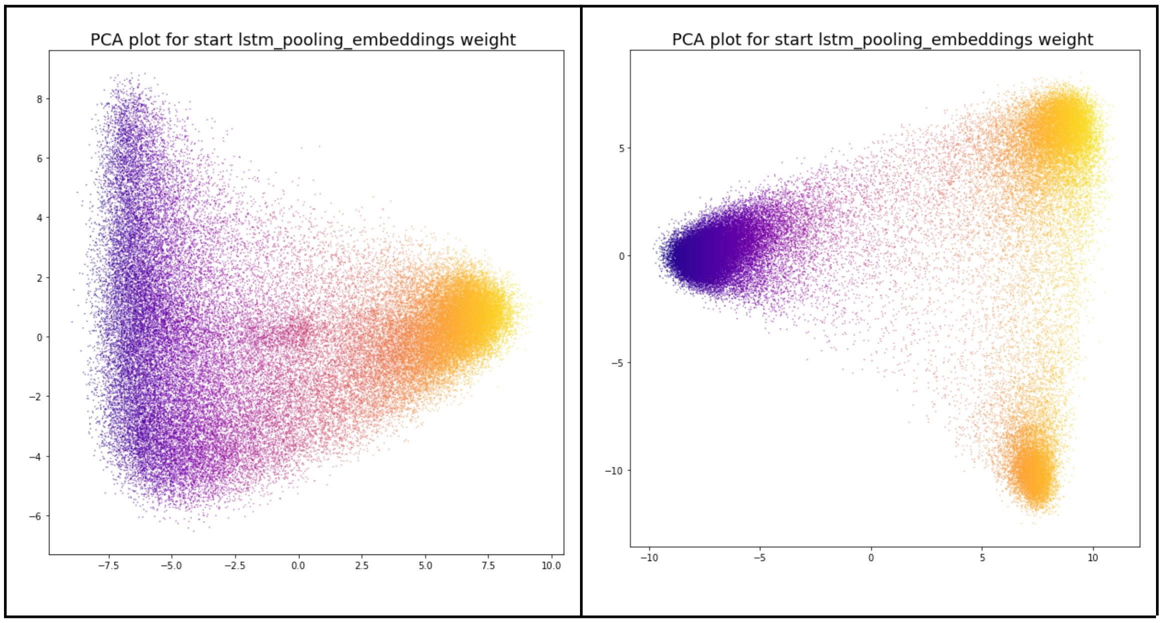


Figure 5.10 MyanBERTa + LSTM Pooling Approach

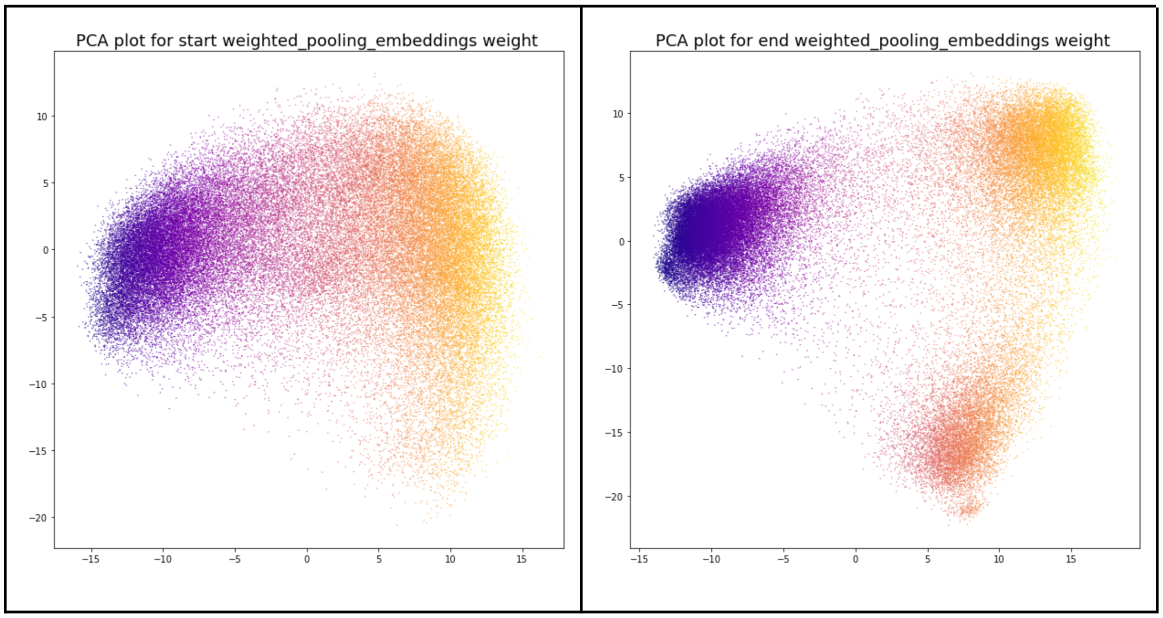


Figure 5.11 MyanBERTa + Weighted Pooling Approach

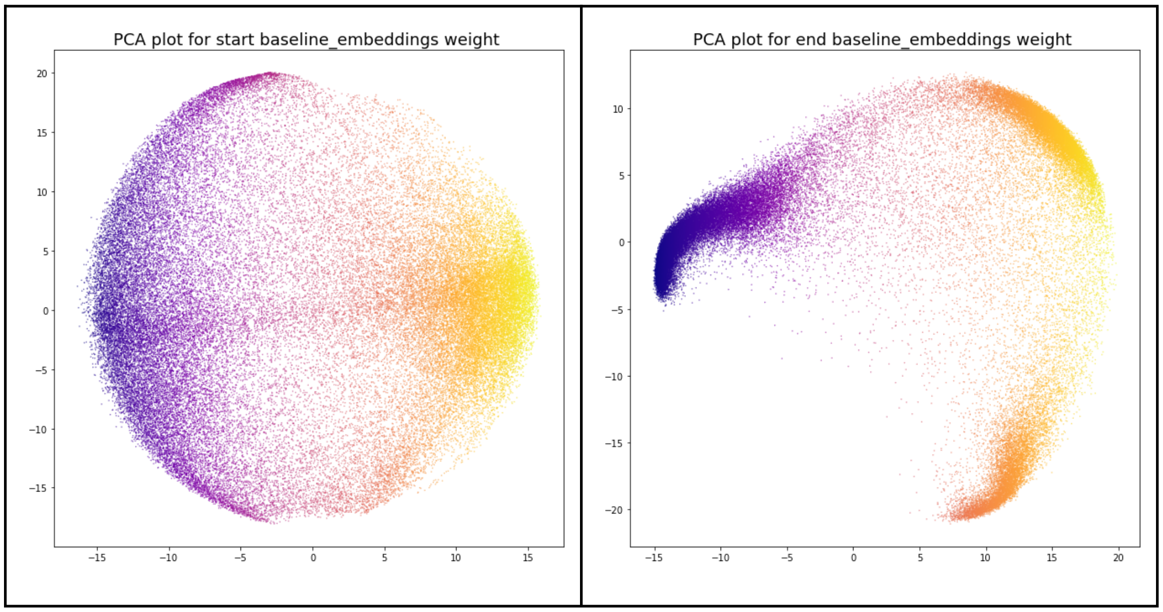


Figure 5.12 XLM-RoBERTa-base + Baseline Pooling Approach

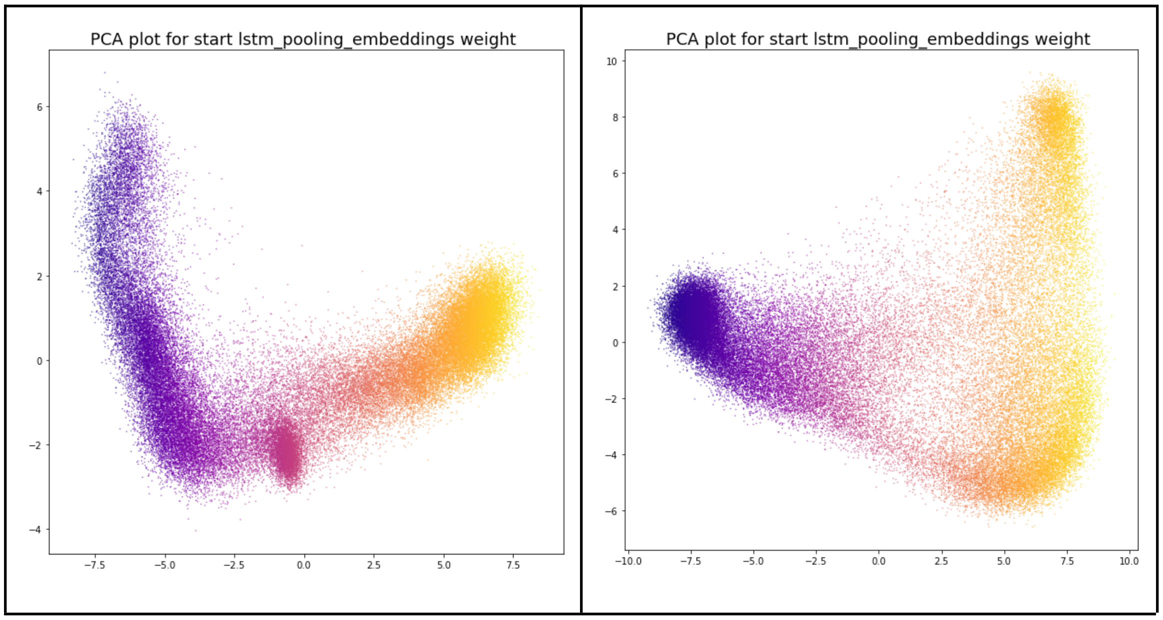


Figure 5.13 XLM-RoBERTa-base + LSTM Pooling Approach

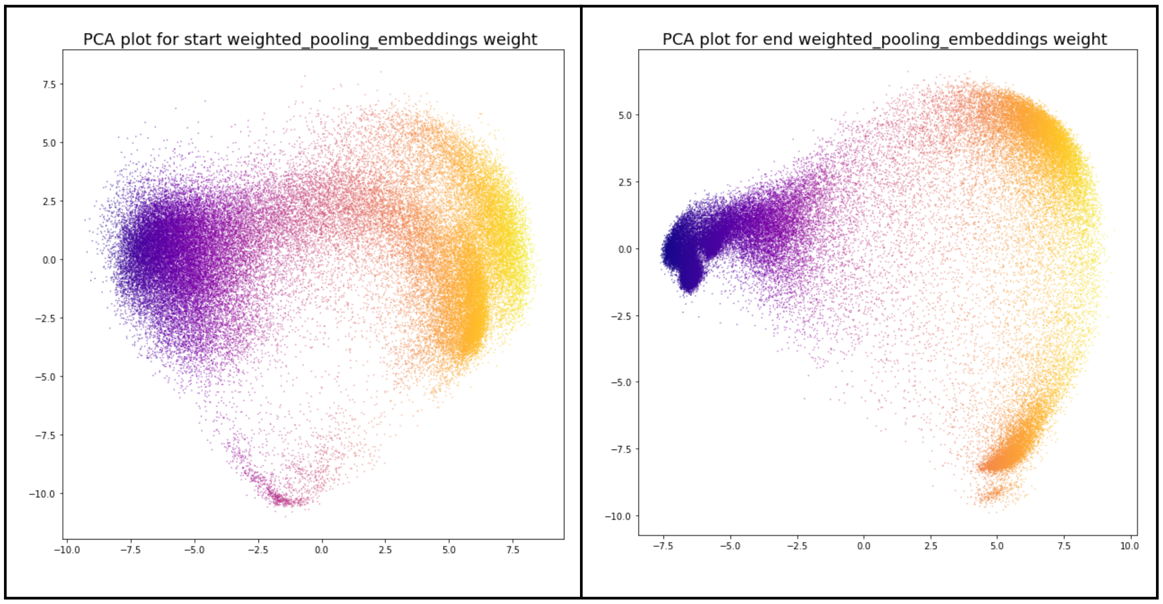


Figure 5.14 XLM-RoBERTa-base + Weighted Pooling Approach

The XLM-RoBERTa models' initial distribution reveals the various clustering. In Figure 5.12, the clustering completely scatters the instance at epoch 1, which prevents learning for the XLM-RoBERTa and baseline models at the beginning. However, the model was denser than the MyanBERTa model at the most recent period. As the MyanBERTa model, the XLM-RoBERTa and LSTM model start to learn at the first epoch that is shown in Figure 5.13.

The weighted approach is obviously different in clustering at the initial stage. In Figure 5.14, the distribution shows us the model starts to learn the label at the first epoch and then well clusters into the classes in the final epoch.

5.5 Performance Criteria

The training dataset is splitted 20% on testset and the models are evaluated with the same training set and test set to get the consistent results .To evaluate the performance of the models, accuracy, recall, precision and F1-score are used. The formula of the evaluations are as follow:

$$Accuracy = \frac{TN + TP}{TN + FP + TP + FN} \quad \textbf{5.1}$$

$$Recall = \frac{TP}{TP + FN} \quad \textbf{5.2}$$

$$Precision = \frac{TP}{TP + FP} \quad \textbf{5.3}$$

$$F1\ Score = 2 * \frac{Precision * Recall}{Precision + Recall} \quad \textbf{5.4}$$

Where TP - True Positive TN - True Negative, FP - False Positive, FN - False Negative.

5.6 Evaluation

To validate the generality of the methods, the experiments are conducted on sentiment dataset and apply the different pretrained models and pooling strategies. The results are shown in Table 6.1 and Figure 5.15, from the results, the baseline performs better than the LSTM and weighted pooling strategies with the MyanBERTa. Furthermore, the xlm-RoBERTa-base weighted pooling outperforms the others. The pooling strategies generally improve with the xlm-RoBERTa-base model. The benefits appear to be negligible, and the adaptability of the approach makes it simpler to use for a range of other tasks.

Table 6.1 The Classification Results of the Models on Accuracy, Recall, Precision and F1-score

Pretained Model	**Model Type**	**Accuracy**	**Recall**	**Precision**	**F1-Score**
myanBERTa	baseline	**84.935662**	84.935662	85.166369	85.027487
	lstm pooling	84.779412	84.779412	84.923734	84.837495
	weighted pooling	84.246324	84.246324	84.403939	84.314696
xlm-RoBERTa-base	baseline	86.047794	86.047794	86.033751	86.037324
	lstm pooling	83.823529	83.823529	84.08864	83.935806
	weighted pooling	**86.424632**	86.424632	86.494176	86.445854

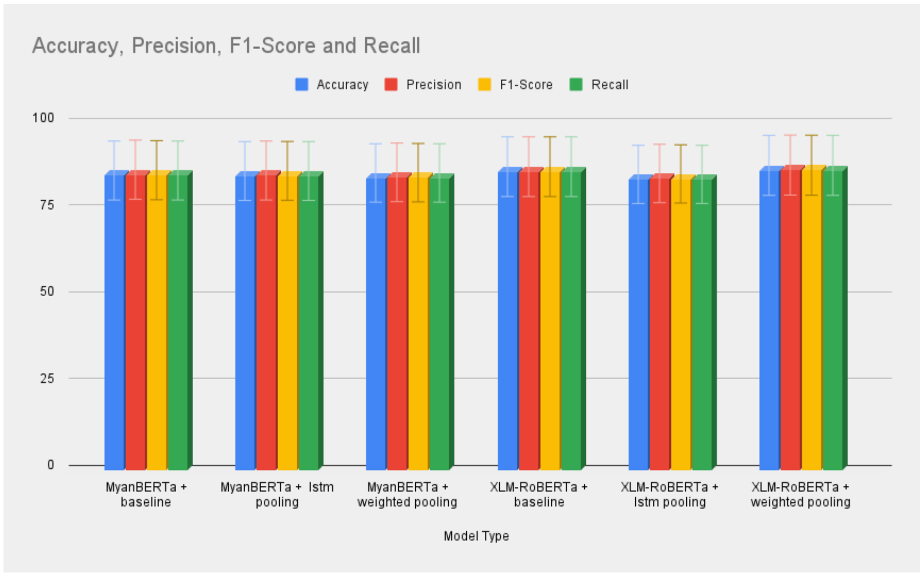
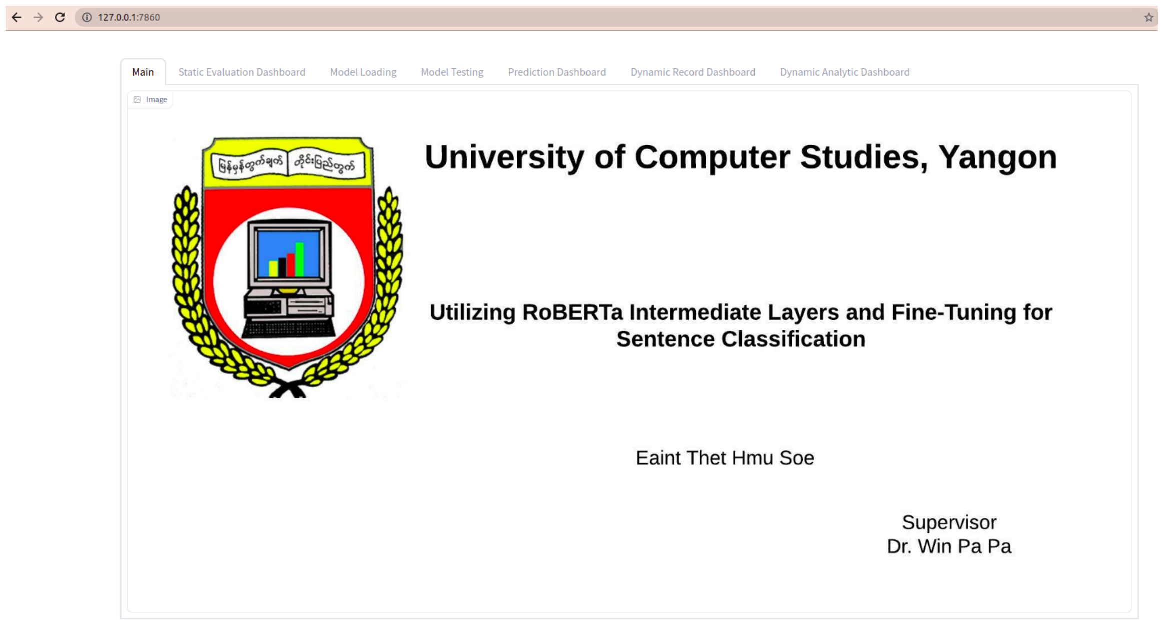


Figure 5.15 Comparison on the Classification Matrices

The proposed LSTM and weighted pooling are compared with the baseline and all models use two RoBERTa models as the backbone and are trained separately for comparison. Results are shown in weight visualization and evaluation tables. The effect of the RoBERTa model is analyzed in the generalization and the pooling strategies are determined to make the model perform better. The different evaluation methods are shown to get better analysis of the models. The LSTM pooling method improves the language specific RoBERTa than the multilingual model. One possible reason for this phenomenon is that the bigger model multilingual RoBERTa and LSTM pooling can overfit. For baseline approach, the results increase just slightly with the both RoBERTa models.By examining the evaluation table, the xlm-RoBERTa-base and weighted pooling strategies outperform the others.

5.7 User Interface Design of the System

The system includes the main dashboard, model selection and loading, live prediction, static evaluation dashboard, record dashboard and dynamic analytic dashboard. The main page of the system is shown in Figure 5.16.



University of Computer Studies, Yangon

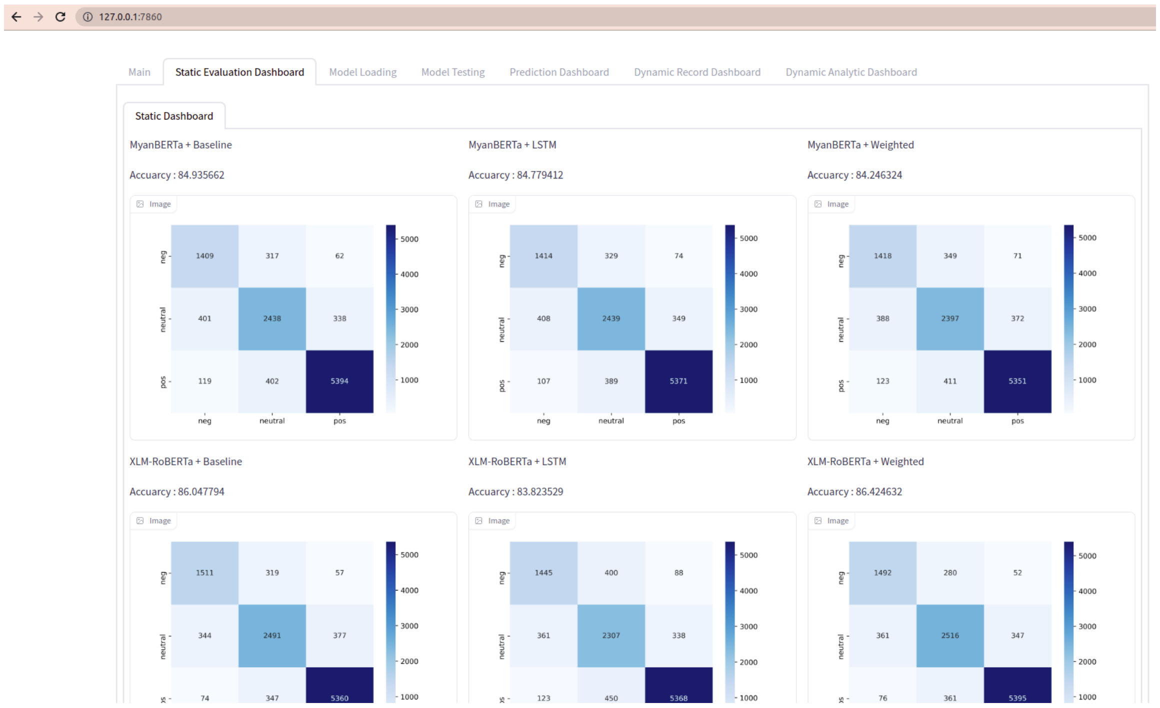
Utilizing RoBERTa Intermediate Layers and Fine-Tuning for Sentence Classification

Eaint Thet Hmu Soe

Supervisor
Dr. Win Pa Pa

Figure 5.16 Main Dashboard of the system

After the model is trained, the classification report includes accuracy, precision, recall and F1-score is saved. In Figure 5.17, the comparison of the confusion matrices and accuracy score is shown.



MyanBERTa + Baseline

Accuracy : 84.935662

	neg	neutral	pos
neg	1409	317	62
neutral	401	2438	338
pos	119	402	5394

MyanBERTa + LSTM

Accuracy : 84.779412

	neg	neutral	pos
neg	1414	329	74
neutral	408	2439	349
pos	107	389	5371

MyanBERTa + Weighted

Accuracy : 84.246324

	neg	neutral	pos
neg	1418	349	71
neutral	388	2397	372
pos	123	411	5351

XLM-RoBERTa + Baseline

Accuracy : 86.047794

neg	1511	319	57
neutral	344	2491	377
	74	347	5360

XLM-RoBERTa + LSTM

Accuracy : 83.823529

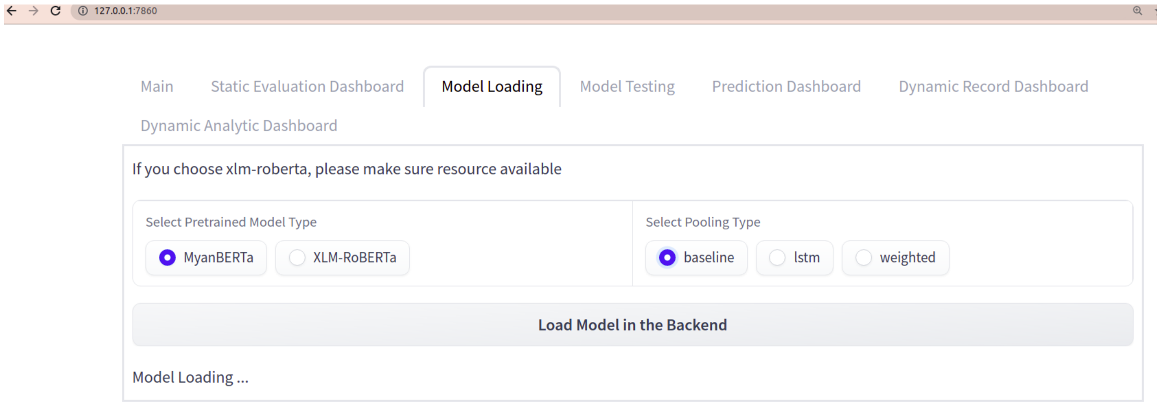
neg	1445	400	88
neutral	361	2307	338
	123	450	5368

XLM-RoBERTa + Weighted

Accuracy : 86.424632

neg	1492	280	52
neutral	361	2516	347
	76	361	5395

Figure 5.17 Evaluation Result Comparison



If you choose xlm-roberta, please make sure resource available

Select Pretrained Model Type: [✓ MyanBERTa] XLM-RoBERTa

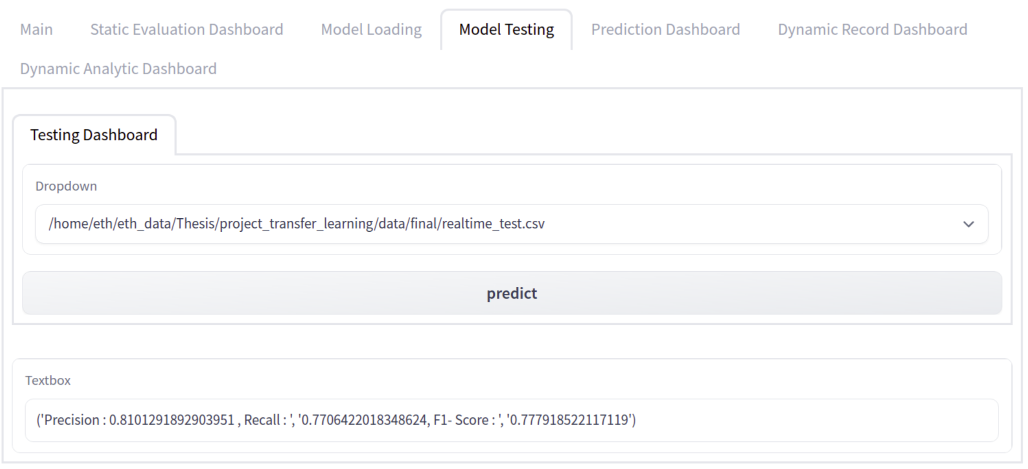
Select Pooling Type: [✓ baseline] lstm weighted

Model Loading ...

Figure 5.18 Model Selection and Loading

Figure 5.18 offers options for the system models. The MyanBERTa and XLM-RoBERTa models can be chosen from the pretrained model column. Selecting the baseline, LSTM, and weighted radio buttons will bring up the pooling type in the left column. After pressing the button, the selected model's flask service is loaded in the backend and prepared for model prediction.

Figure 5.19 shows a dropdown row where the text file can be chosen. Once the predict button has been pressed, the loaded model is parsed to load the selected file, which is then loaded, and the loaded model predicts the data to determine the Precision, recall, and F1-score.

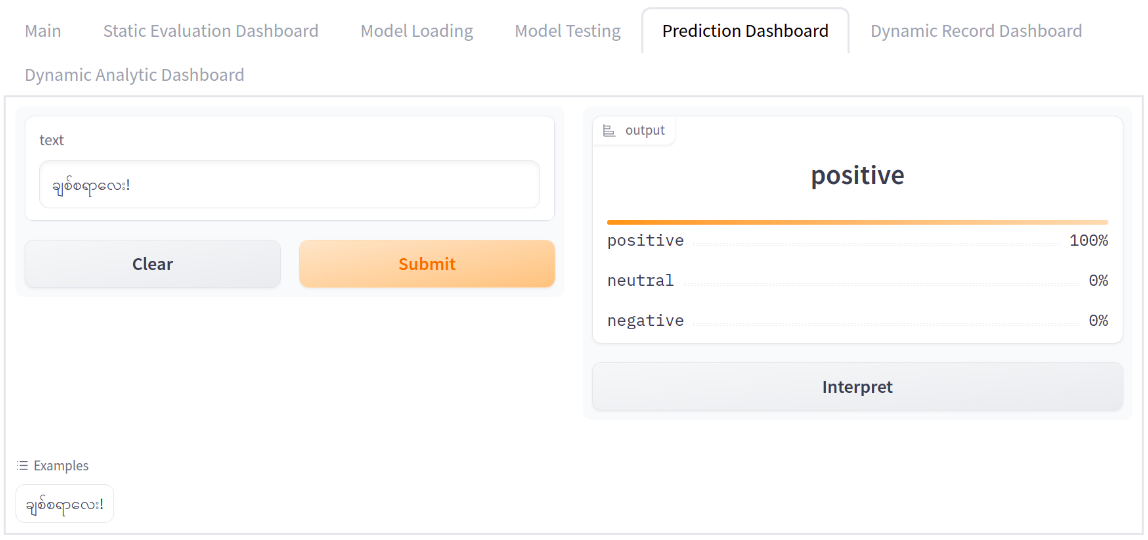


```
/home/eth/eth_data/Thesis/project_transfer_learning/data/final/realtime_test.csv
```

```
('Precision : 0.8101291892903951 , Recall : ', '0.7706422018348624, F1- Score : ', '0.777918522117119')
```

Figure 5.19 Real Time System Testing with the 100 Random Selected Sentences Text File

In the Figure 5.20 prediction dashboard, the left side is the sentence input column and the right side is the result predicted by the model in the backend. A clear button and a submit button are located on the left. The clear button deletes the sentence in the input text box and the submit button calls and predicts the loaded model to get the result that is shown on the right column. The probability distributions across the classes are examined in the right column, and the interpret button will re-index the text. The prediction is depicted with a neutral sentence in Figure 5.21. Figure 5.22 illustrates the prediction using a negative sentence.



text: ချစ်စရာလေး!

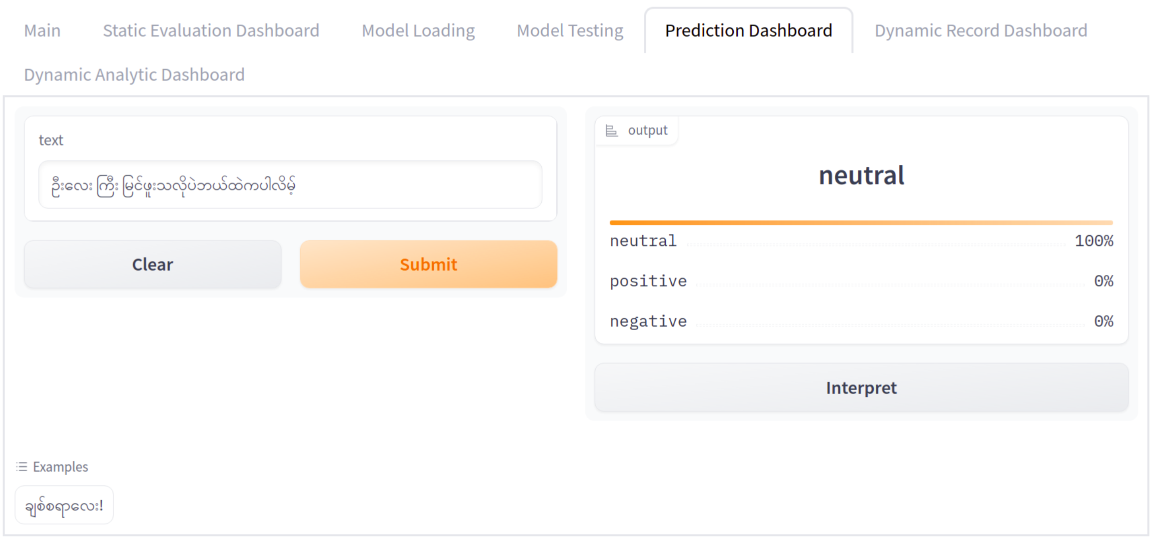
output

positive

```
positive    100%
neutral       0%
negative      0%
```

Examples: ချစ်စရာလေး!

Figure 5.20 Real Time System Testing with Positive Sentence



text: ဦးလေး ကြီး မြင်ဖူးသလိုပဲဘယ်ထဲကပါလိမ့်

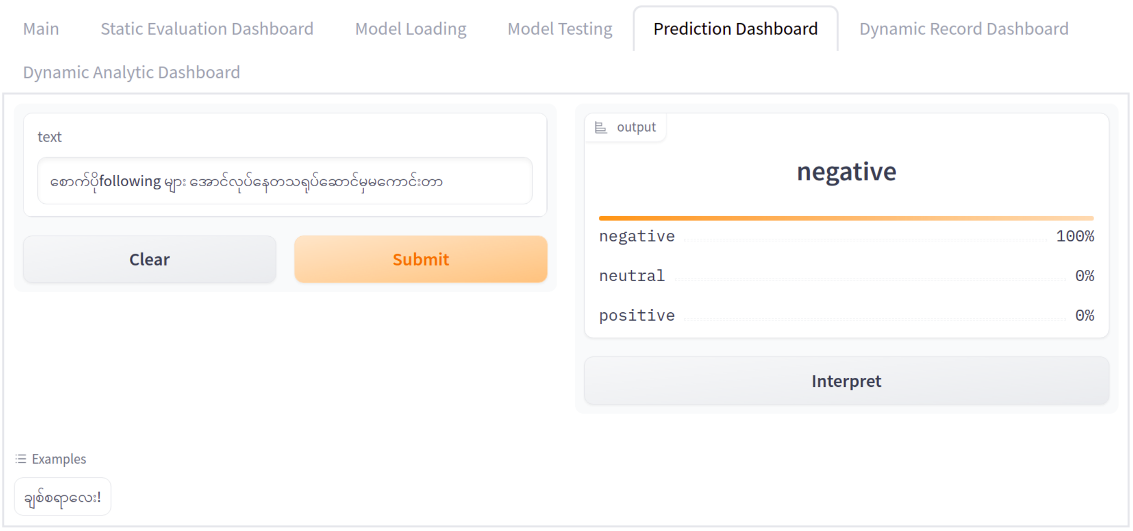
output

neutral

```
neutral     100%
positive      0%
negative      0%
```

Examples: ချစ်စရာလေး!

Figure 5.21 Real Time System Testing with Neutral Sentence



text: စောက်ပိုfollowing များ အောင်လုပ်နေတသရုပ်ဆောင်မှမကောင်းတာ

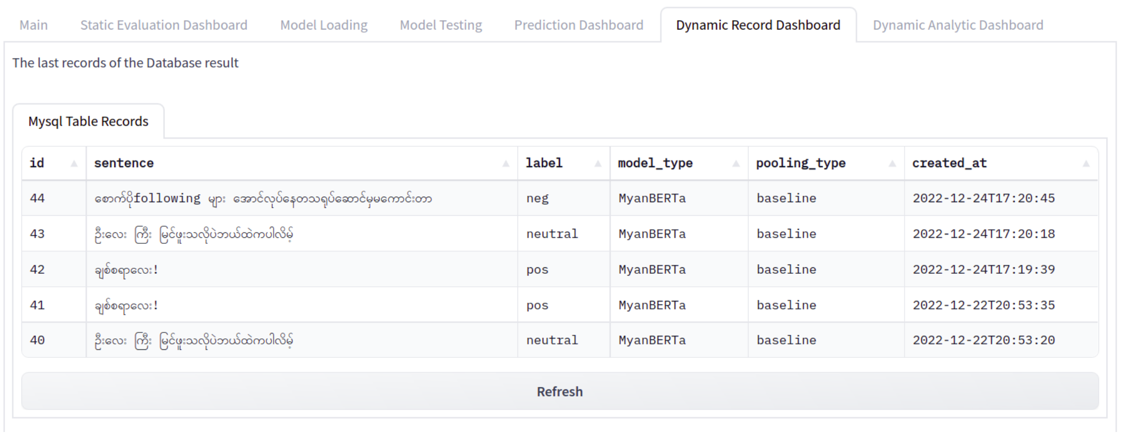
output: negative

negative	100%
neutral	0%
positive	0%

Examples: ချစ်စရာလေး!

Figure 5.22 Real Time System Testing with Negative Sentence

The record that was chosen from the mySQL database is shown in Figure 5.23. The most recent record is updated if the refresh button is hit. The four bar plot, which represents the label frequency, model frequency, and pooling type frequency across the sentence count and the pooling type and pretrained model frequency, is part of the dynamic analytical dashboard seen in figure 5.24.



The last records of the Database result

id	sentence	label	model_type	pooling_type	created_at
44	စောက်ပိုfollowing များ အောင်လုပ်နေတသရုပ်ဆောင်မှမကောင်းတာ	neg	MyanBERTa	baseline	2022-12-24T17:20:45
43	ဦးလေး ကြီး မြင်ဖူးသလိုပဲဘယ်ထဲကပါလိမ့်	neutral	MyanBERTa	baseline	2022-12-24T17:20:18
42	ချစ်စရာလေး!	pos	MyanBERTa	baseline	2022-12-24T17:19:39
41	ချစ်စရာလေး!	pos	MyanBERTa	baseline	2022-12-22T20:53:35
40	ဦးလေး ကြီး မြင်ဖူးသလိုပဲဘယ်ထဲကပါလိမ့်	neutral	MyanBERTa	baseline	2022-12-22T20:53:20

Figure 5.23 Dynamic Record Dashboard

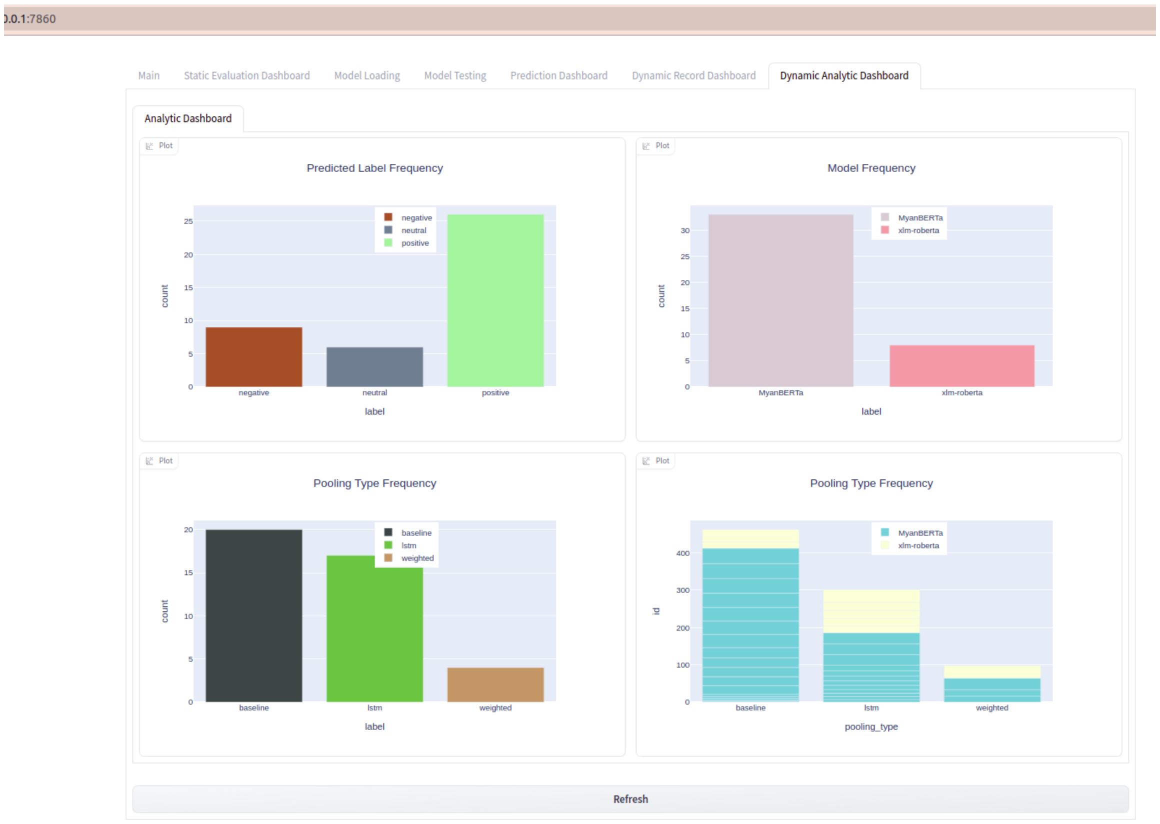


Figure 5.24 Dynamic Analytic Dashboard

CHAPTER 7

CONCLUSION

Sentiment analysis involves identifying and extracting subjective information from text data. It is often used to determine the overall sentiment of a piece of text, whether it is positive, negative, or neutral. Sentiment analysis has a wide range of applications, including social media analysis, customer service, and market research. However, there are also limitations to the technique, including the subjectivity of language and the difficulty in accurately identifying the sentiment of more nuanced or complex text.

Overall, sentiment analysis is a useful tool for extracting subjective information from text data, but it is important to carefully consider the limitations of the approach and the specific requirements of the task at hand when using it. There are many approaches to performing sentiment analysis, including rule-based systems, machine learning models, and deep learning models.

In this research, the pretrained contextualized language models are used and analyzed with different evaluation matrices. This approach compares and contrasts the utilization of the intermediate representation of the RoBERTa model. The transfer learning and fine tuning effects on the classification mode are explored. Many researches have shown that transformer models such as BERT with proper fine-tuning can play a crucial role in sentiment analysis.

This method is to explore the effectiveness of the intermediate layer of the RoBERTa model and compare the results on the RoBERTa models and fine tuning approaches. Finally, the conclusion can be drawn that generalized pre-trained language models do not require the explicit construction of extraordinarily complicated networks. The outcome of this experiment suggests that using deeper layers may be a more effective strategy to boost accuracy.

7.1 Future Work

With the little dataset, this study is applicable to any classification issue. The transformer models such as RoBERTa with proper fine-tuning can play a crucial role in sentiment analysis by training minimal epochs. The various fine-tuning layers (poolers) are examined in this study to see which works best. A technique for assessing the suggested model's performance in real-world applications is offered. This method seeks to apply it for more occupations after analyzing its possibilities. The sequence labeling task is planned to train using the approach. Making generation on the minimal dataset using fine-tuning advantage is the goal of the research.

Publication

[1] Eaint Thet Hmu Soe, Win Pa Pa, “Utilizing RoBERTa Intermediate Layers and Fine-Tuning for Sentence Classification”, published in the ICCR 2022, Korea

References

[1] Aye Mya Hlaing, Win Pa Pa. MyanBERTa: A Pre-trained Language Model For Myanmar. 2022 International Conference on Communication and Computer Research. ICCR Korea. 2022

[2] Conneau, Alexis, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer and Veselin Stoyanov. "Unsupervised Cross-lingual Representation Learning at Scale." Annual Meeting of the Association for Computational Linguistics, 2019.

[3] Hay Mar Su Aung, Win Pa Pa. Analysis of Word Vector Representation Techniques with Machine-Learning Classifiers for Sentiment Analysis of Public Facebook Page's Comments in Myanmar Text. 18th International Conference on COmputer Application (ICCA 2020), Yangon, Myanmar, 2020

[4] Hongchan Li, Yu Ma, Zishuai Ma, Haodong Zhu*. Weibo Text Sentiment Analysis Based on BERT and Deep Learning. Applied Sciences. 2021; 11(22):10774. 2021

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv preprint arXiv:1810.04805, 2019

[6] M.D. Devika, Sunitha C, Amal Ganesh. Sentiment Analysis: A comparative Study on Different Approaches, Procedia Computer Science, Volume 87, 2016

[7] Prajval Sudhir, Varun Deshakulkarni, Suresh. Comparative study of various approaches, applications and classifiers for sentiment analysis, Global Transitions Proceedings, Volume 2, Issue 2, 2021

[8] Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. arXiv preprint arXiv:1301.3781, 2013

[9] Vaswani, Ashish and Shazeer, Noam and Parmar, Niki and Uszkoreit, Jakob and Jones, Llion and Gomez, Aidan N and Kaiser, Lukasz and Polosukhin, Illia. Attention Is All You Need . arXiv preprint arXiv:1706.03762, 2017

[10] Xin Li1, Lidong Bing2, Wenxuan Zhang1 and Wai Lam1. Exploiting BERT for End-to-End Aspect-based Sentiment Analysis arXiv preprint arXiv:1910.00883, 2019

[11] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, Veselin Stoyanov. RoBERTa: A Robustly Optimized BERT Pretraining Approach, ArXiv abs/1907.11692, 2019

[12] Youwei Song, Jiahai Wang *, Zhiwei Liang, Zhiyue Liu, Tao Jiang. 2020. Utilizing BERT Intermediate Layers for Aspect Based Sentiment Analysis and Natural Language Inference, ArXiv abs/2002.04815, 2020

[13] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, Radu Soricut. ALBERT:A Lite BERT for Self-supervised Learning of Language Representations, International Conference on Learning Representations, 2020

[14] https://www.kdnuggets.com/2021/11/guide-word-embedding-techniques-nlp.html

[15] https://intellipaat.com/blog/what-is-lstm/

[16] https://towardsdatascience.com/bert-explained-state-of-the-art-language-Model-for-nlp-f8b21a9b6270

[17] https://github.com/Rabbit-Converter/Rabbit

[18] https://en.wikipedia.org/wiki/Deep_learning