BRAUDE
College of Engineering, karmiel

Software Engineering Department

ORT Braude College

Capstone Project Phase A – 61998

Transfer Learning for Sentiment Analysis Using BERT Based Supervised Fine-Tuning

23-2-R-9

Dorin Hila Bachar 313603367
Dorin.hila.bachar@e.braude.ac.il

Moshe Moalem 205802945
Moshe.Moalem@e.braude.ac.il

:Supervisors
Prof. Zeev Volkovich
Dr. Renata Avros

Table Of Contents

Abstract

Sentiment analysis, a critical aspect of Natural Language Processing (NLP), has seen significant advancements with the introduction of deep learning models. This paper explores the application of the Bidirectional Encoder Representations from Transformers (BERT) model, a state-of-the-art deep learning model, in sentiment analysis tasks. The study focuses on the process of fine-tuning the pre-trained BERT model, a technique known as transfer learning, which allows the model to apply the knowledge gained from pre-training on a large corpus of text to specific tasks. The paper presents a detailed comparison of BERT with other models like Word2Vec, GloVe, and FastText, highlighting the superior performance of BERT in sentiment analysis tasks. The study also introduces Bangla-BERT, a specific pre-trained BERT model for the Bangla language, which has shown remarkable results in various sentiment analysis tasks. The findings of this study underscore the potential of BERT-based supervised fine-tuning in enhancing the accuracy and efficiency of sentiment analysis.

Index Terms: Deep Learning; Sentiment Analysis; Transformers; Word Embedding; BERT; Bangla-BERT; Binary Classification; Convolution Networks.

1. Introduction

Sentiment analysis, a subfield in the realm of Natural Language Processing (NLP), has been the focus of numerous studies in the past years. This process, which seeks to understand emotions through the body of text, has been approached through a variety of machine learning and deep learning methodologies. The ultimate goal of sentiment analysis is to comprehend an individual's sentiment or emotions, a task that can be accomplished by extracting hidden features from a text, constructing a model to classify these features, and subsequently assessing the model's performance. This technique is used worldwide in a multitude of domains, including the classification of movie reviews, online product reviews, and social media posts.

The scope of sentiment analysis research is not limited to a single language or domain but in fact, it spans various languages and fields, with models capable of distinguishing between positive and negative sentiments in restaurant reviews, amazon products reviews, and even conducting sentiment analysis on Twitter data in different languages such as Portuguese and Arabic.

The advent of pre-trained language models has revolutionized sentiment analysis. Models such as Elmo, GPT, and BERT have been used on large quantities of unlabeled data for the purpose of getting a deeper understanding of language.

Among these models, BERT received significant attention due to its unique bidirectional attention mechanism. However, given that BERT is trained exclusively in English, researchers have taken it upon themselves to develop language-specific BERT models to enhance accuracy.

There are a few examples such as the Arabic BERT model (AraBERT) [1], Dutch (BERTje)[2], and Romanian (RobBERT) [11] models.

In the context of the Bangla language, sentiment analysis has also made significant strides. For instance, a dataset of Bangla book [3] reviews was created and a machine learning approach was applied, yielding promising results. Other studies have also utilized SVM on a dataset of Bangladesh cricket [7], achieving noteworthy accuracy.

The field of sentiment analysis is not limited to the mentioned studies, there is countless other research using deep learning techniques, such as Long Short-Term Memory (LSTM) networks and Convolutional Neural Networks (CNNs).

These have been particularly effective for sentiment analysis due to their ability to capture long-term dependencies in text data and analyze social media text, which often contains informal language, misspellings, abbreviations, and acronyms.

The use of attention mechanisms in deep learning models for sentiment analysis has seen an uprise in popularity in recent years. These mechanisms enable models to concentrate on the most relevant parts of the input when making predictions, a feature that is especially beneficial in sentiment analysis, where the sentiment of a text is often manifested by a few keywords or phrases.

2. Related Work

Different studies have been conducted on sentiment analysis, a process that determines the emotional tone behind words, employing different machine learning and deep learning techniques.

Sentiment analysis is about understanding an individual's sentiment or feelings, which is achieved by extracting features from a text corpus, building a model to classify these features, and then evaluating the model's performance. This approach has found applications in different areas such as classifying movie reviews, online product reviews, and social media posts.

Research on sentiment analysis has been carried out in different languages and domains. For instance, a model developed by Akshay [6] and his team could discern positive and negative sentiments in restaurant reviews with an accuracy of up to 94.5%. Another study [12] that used the SVM classifier to analyze smartphone reviews reported an accuracy of 81.77%. Other researchers have focused on sentiment analysis of Twitter data in Portuguese [13], while a team led by Ombabi developed a sentiment classifier for Arabic [9] that achieved an accuracy of 90.75%, surpassing previous methods.

Using BERT models in different languages for the task of sentiment analysis has huge benefits, for example, the Arabic BERT model (AraBERT) [9], [1] achieved a score of 99.44 in a sentiment analysis experiment, while the Persian (PersBERT)[4], Dutch (BERTje) [2], and Romanian (RobBERT) [11], [8] models scored 88.12, 93.00, and 80.44, respectively. In the context of the Bangla language, a team led by Hossain created a dataset of Bangla book reviews [4], applied a machine learning approach, and found that the Multinomial Naive Bayes (MNB) method had an accuracy of 88%.

Another study using SVM on a dataset of Bangladesh cricket [7] achieved an accuracy of 64.60%. A sentiment classifier proposed by Sarker [5] and his team for Bengali tweets outperformed n-gram and SentiWordnet features by 45%.

3. Background

This section is dedicated to describing the theory, the mathematical foundations, and the history of our model.

3.1 Neural Network

A neural network is a type of machine learning model that simulates the human brain's structure and attempts to teach it in the same way. Neural networks consist of multiple layers of interconnected nodes called neurons that process the input to generate output. The data is received from the Dendrite, the data processed in the Nucleus, and after that through the Axon to other neurons. In the neural network, each node is a neuron, and each edge is a connection between 2 neurons. The goal of a neural network is to learn from data and generalize the learning to make predictions, accurate decisions, problem-solving, and classification. By iteratively adjusting the weights and biases of the network during the training process, the neural network aims to optimize its performance and improve its ability to solve the task.

3.1.1 Input layer

The neural network receives input data, which is usually a set of numerical values that represent some features of the data.

3.1.2 Weights

Each edge in the network has a weight, which determines how strongly it responds to the input data.

3.1.3 Vanishing Gradient

In deep neural networks especially with recurrent connections or many layers, there is a problem called the vanishing gradient that challenges the training process of the model. The problem is that in the process of training the neural network using a gradient-based optimization algorithm such as gradient descent, the gradients of the loss function with respect to the parameters are computed and used to update the weights. The gradients are multiplied as they propagate back through the many layers so that each layer presents its own set of weights. When gradients are multiplied layer by layer, they can become exponentially small. As a result, the early layers receive very small gradients, leading to slow or no learning in those layers.

This can be seen in the graph of the sigmoid function and its derivative. For very large values for the sigmoid function, the derivative takes a very low value.

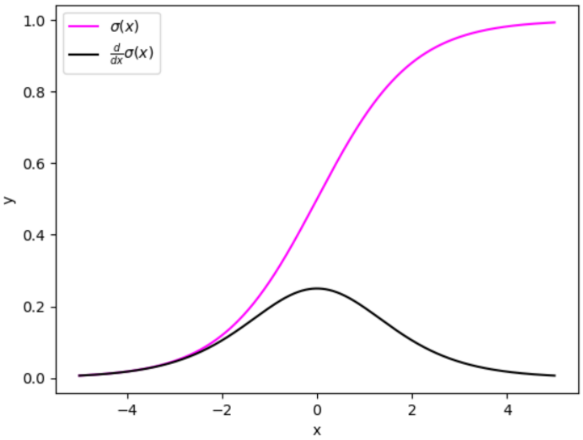


Figure 1, The Sigmoid activation function and it's derivative. [source]

Several techniques have been developed to solve the vanishing gradient problem: activation functions, batch normalization, weight initialization, gradient clipping, and improving models by changing architecture such as the RNN model to the LSTM model.

3.1.3.1 Activation Function

For each edge in a neural network, we will multiply the input and weight of the edge and sum between them. The result will enter the activation function. If the result of the activation function is higher than the threshold, then data will be passed to the next neuron otherwise not.

The activation function can have a significant impact on the vanishing gradient problem. When gradients close to zero, they disappear as they propagate back through the network layers. There are common activation functions that help alleviate the vanishing gradient problem by maintaining a non-zero gradient for positive inputs.

Sigmoid

The sigmoid is the logistic function that maps an input value to a value between 0 and 1. The sigmoid function has an S-shaped curve and is symmetric around its midpoint x=0. When the input value is very large the function will approach the value 1 and for a very low value, the function will approach the value 0. For input values near zero, the output of the sigmoid function is close to 0.5.

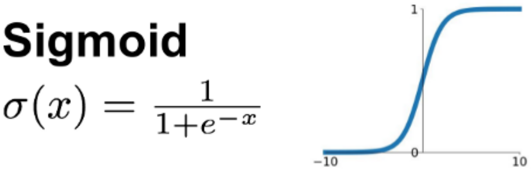


Figure 2, The Sigmoid function. [source]

tanh

A tanh function, also called the hyperbolic tangent function, is a mathematical function that maps a value between -1 and 1. It is symmetric around the origin x=0. When the input value is very large the function will approach the value 1 and for a very low value, the function will approach the value -1. For input values near zero, the output of the tanh function is close to 0.

In some cases, a tanh function is an advantage over a sigmoid function because it is zero-centered, which means that its average output value is close to 0.

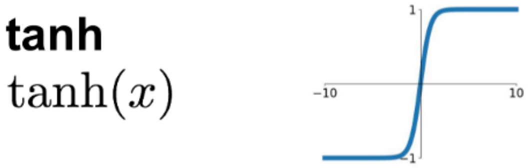


Figure 3, The Hyperbolic Tangent function. [source]

ReLU

The Rectified Linear Unit (ReLU) function returns the maximum of zero, and the input value. If the input is positive, it outputs the value directly, and if it is negative, it outputs 0.

One of the main advantages of the ReLU function allows the model to learn more quickly and effectively by allowing the gradient to flow more easily for positive inputs, as the derivative of ReLU is 1 for positive inputs. This can lead to faster and more stable convergence during training. But the function has a problem called "ReLU dying". The ReLU dying is the condition where neurons become permanently inactive, output zero, due to a large negative bias or a not good initialization it no longer contributes to the learning process. This issue can be addressed by using Leaky ReLU or ELU.

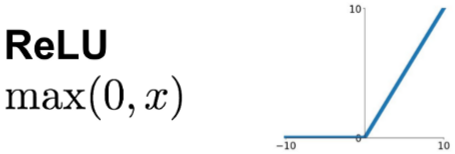


Figure 4, The Rel-U function. [source]

Leaky ReLU

The Leaky ReLU function is a variant of the ReLU activation function that addresses the "dying ReLU" problem. The Leaky ReLU for positive inputs outputs the input itself, but for negative inputs, it returns the input multiplied by the small constant a (typically set to a small value like 0.01 or 0.2). This non-zero output for negative inputs helps prevent the dying ReLU problem.

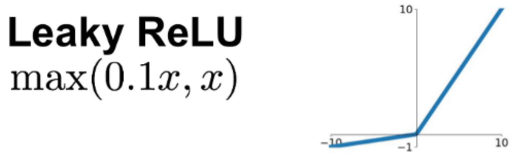


Figure 5, The Leaky Rel-U function. [source]

ELU

ELU (Exponential Linear Unit) is another variant of ReLU activation that addresses the limitations of both ReLU and Leaky ReLU. In addition to the non-zero slope for negative inputs, ELU also incorporates an exponential component, so that there is a smooth transition from negative to positive inputs.

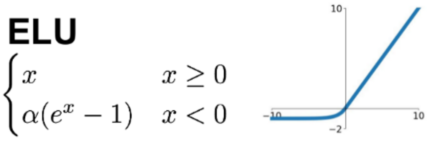


Figure 6, The ELU function. [source]

3.1.3.2 Batch Normalization

This technique normalizes input data by subtracting its mean and dividing it by its standard deviation before it is fed to each layer of a neural network. As a result, the distribution of inputs is stabilized and the internal covariate shift is reduced.

3.1.3.3 Weight Initialization

A controlled initialization of the weights in the network can alleviate the vanishing gradient problem. Initialization of the weights so that gradients are neither too small nor too large.

3.1.3.4 Gradient Clipping

Clipping the gradients to a predefined threshold prevents the gradients during training from becoming too large or too small. As a result, the network learns and updates the weights more efficiently.

3.1.4 Output Layer

The output layer will produce the prediction or output of the network's problem solution after the weights and activation function are applied. The weights and activation function act as a filter, allowing only data that meets certain criteria to pass through. The output layer then produces the desired output of the neural network.

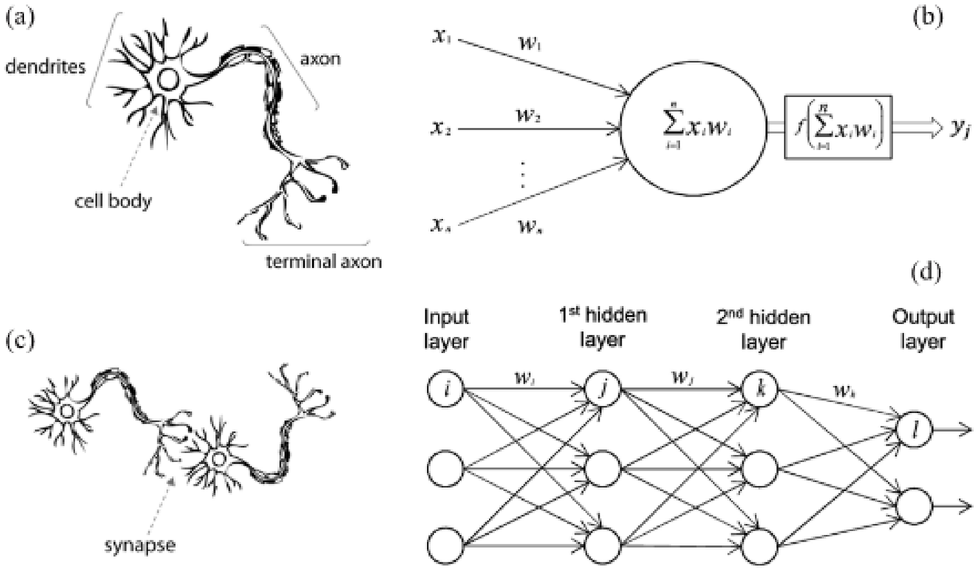


Figure 7, Neural network models simulate the structure of neurons in the human brain. [source]

3.2 CNN- Convolutional Neural Network

CNN is a type of neural network that is particularly effective in data processing such as: extracting features from text or images, object detection, and classifying. CNN uses particular layers: convolutional, max pooling, and fully connected to learn and extract relevant features from the input data.

3.2.1 Convolutional Layers

CNNs use convolutional layers to scan the input data for patterns or features. These layers apply a set of filters also called kernels to the input data, which identify specific features.

The kernel is a small matrix of weights that slides over the input and performs element multiplication with the corresponding elements of the input and sums the results. There are often multiple kernels in a CNN's convolutional layer. Kernels each focus on detecting a different feature, and their outputs form separate feature maps. During the CNN training process, the kernel weights are learned.

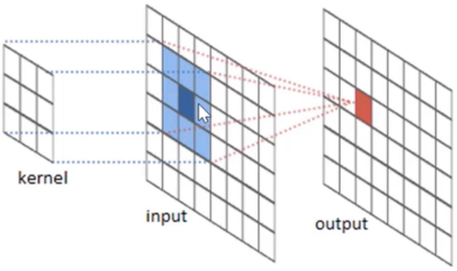
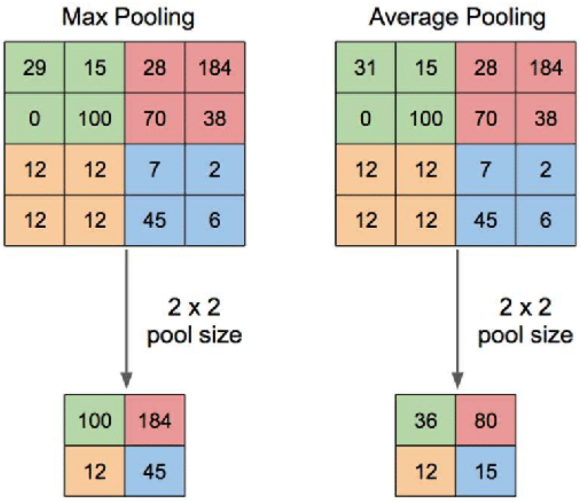


Figure 8, The operation of the convolution layer. [source]

3.2.2 Pooling Layers

Max pooling works by taking the maximum value from each of the sections of the feature map, while average pooling takes the average of the values in each of the sections. This helps to reduce the computational complexity of the model, while still preserving the most important information from input.



Max Pooling

29	15	28	184
0	100	70	38
12	12	7	2
12	12	45	6

100	184
12	45

Average Pooling

31	15	28	184
0	100	70	38
12	12	7	2
12	12	45	6

36	80
12	15

Figure 9, The operation of the polling layers. [source]

3.2.3 Fully Connected Layers

A final classification or regression layer is typically added to the network at the end to perform the final task. They take the high-level features extracted from the previous layers and map them to the desired output format, such as class probabilities for classification tasks. The purpose of training is to find the optimal values of the kernel weights that allow the network to learn and detect meaningful features in the input data that are relevant to the specific task.

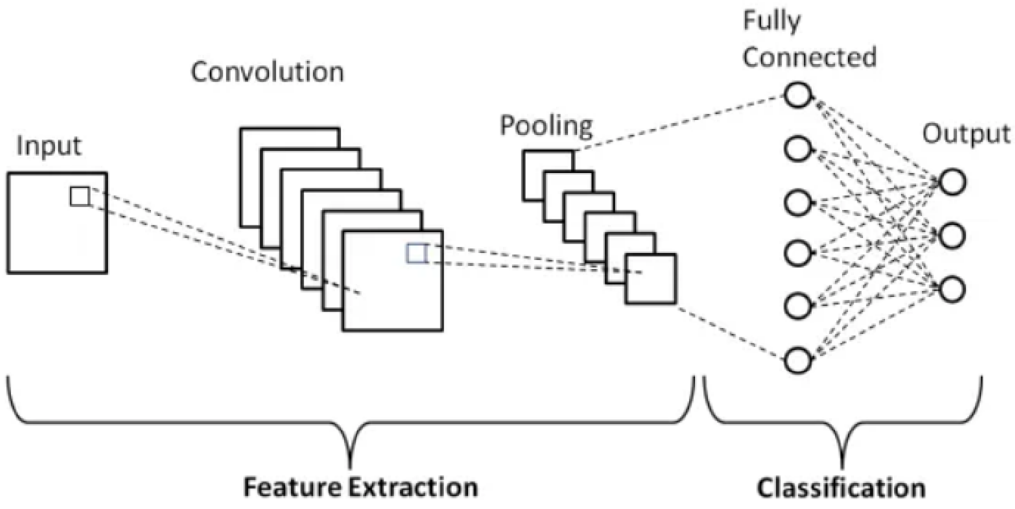


Figure 10, The architecture of CNN. [source]

3.2.4 Training of CNN

Training a CNN involves the process of optimizing the network's parameters such as weights to learn from training data and improve its performance on a specific task. Here are the steps in training a CNN.

Initialization network

This step provides an initial starting point for the network to begin learning. Initialize the weights and biases of the CNN randomly.

Forward Propagation

This step performs forward propagation through the network. In this process, the input data is fed through layers of the network, and convolutions, pooling, activation functions, and other operations are applied to produce predicted outputs.

Loss Function Calculation

This step compares the predicted output with the actual output from the data using a suitable loss function. Loss functions estimate the error between predicted and actual outputs.

Backpropagation

The purpose of the backpropagation process is to calculate the gradient of the loss function in relation to the weights and biases of the network and then update the weights in the opposite direction of the gradient to minimize the loss.

Iteration

Repeat the training steps with multiple iterations or until you reach the desired level of accuracy. Each iteration helps the network adjust its parameters based on the gradients and get closer to the optimal values that minimize the loss.

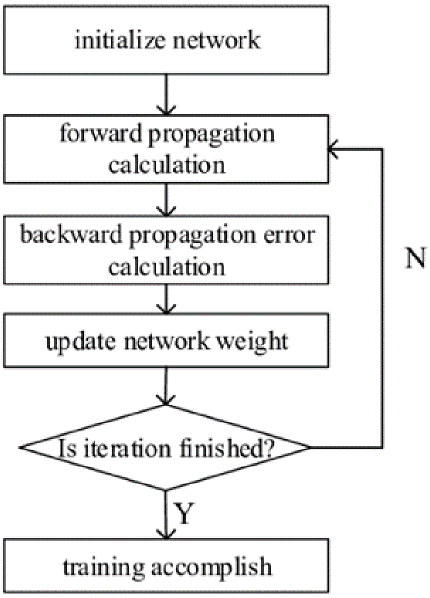


Figure 11, The stages of the training process of the CNN. [source]

3.3 LSTM - Long Short-Term Memory

LSTM is a type of recurrent neural network (RNN) architecture that is specifically designed to capture long-term dependencies in sequential data. LSTMs overcome the vanishing gradient problem (diminishing gradient when backpropagation through time) better than RNNs by mitigating the exponential deterioration of gradients over time. LSTM is widely used for various NLP (natural language processing) tasks, such as translation, sentiment analysis, speech recognition, and more. Its ability to capture long-term dependencies and efficiently process sequential data makes it a powerful tool in the field of deep learning.

3.3.1 Long Short-Term Memory Architecture

The LSTM architecture contains a memory cell and three gates: The input gate, the forget gate, and the output gate. These gates control the flow of information into, out of, and within the memory cell, and allow it to retain or discard information selectively, that is, it performs selection for information.

Memory Cell

The memory cell maintains and updates information over time through linear activation function. It retains relevant information and prevents the vanishing gradient problem.

Input Gate

The input gate decides how much information from the input data should be stored in the memory cell. The way the information is saved is determined by a sigmoid activation function, which produces values between 0 and 1. When the value is 0, there is no need to save the new information, when the value is 1, all the information should be stored.

Forget Gate

The forget gate decides how much information from the previous memory cell state should be forgotten. The way of forgetting the information from the memory cell is determined by a sigmoid activation function. When the value is 0, information should be forgotten, when the value is 1, information should be retained for future use.

Output Gate

The output gate decides how much information from the memory cell state should be outputted to the layer, considering the previous and current states. This is the task of extracting useful information from the current cell state.

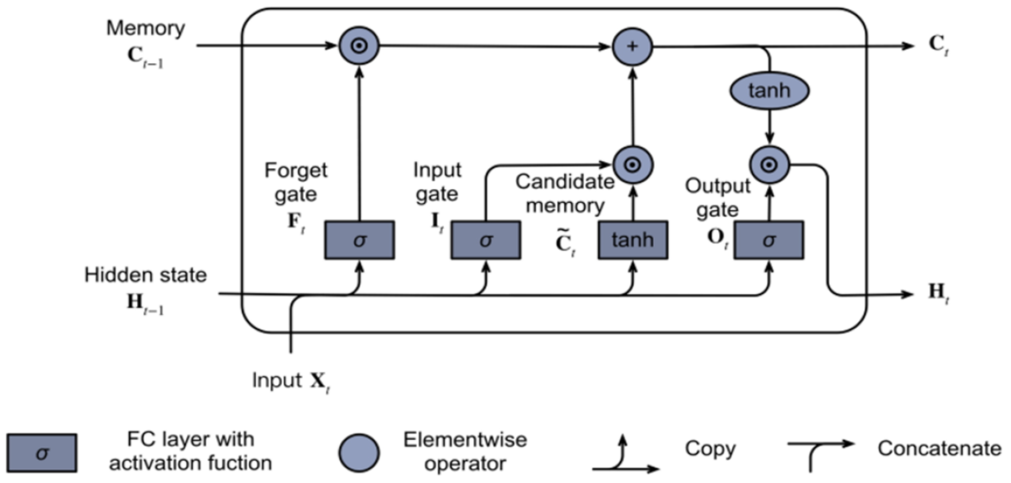


Figure 12, The memory cell of the LSTM layer. [source]

3.4 Bi-LTSM

An extension of the standard LSTM network that processes sequences in both forward and backward directions which means that it can capture richer contextual information and have improved performance on sequence-based tasks.

The input process goes through 2 separate LTSM layers, one for forward processing and one for backward processing, with contextual information being captured from the past and future elements in the sequence.

Later the outputs of both LSTM layers are combined, typically by concatenation or addition to one unified representation.

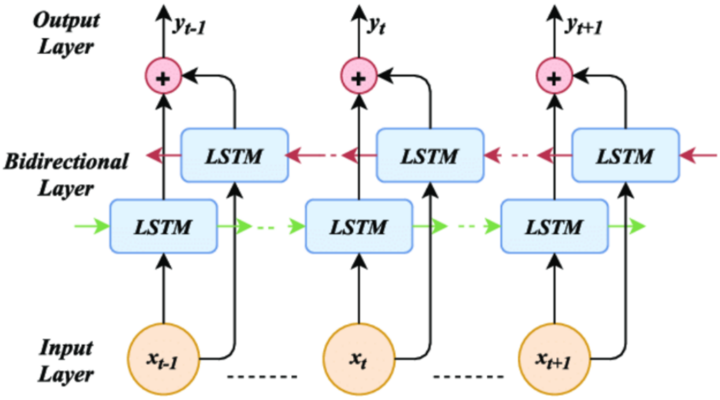


Figure 13, The architecture of Bi-LSTM. [source]

3.5 Word Embedding

It is a crucial aspect of Natural Language Processing (NLP) that involves the representation of words or phrases in a numerical format mostly as a high-dimensional vector space. This technique allows algorithms to capture the semantic and syntactic relationships between words, and so enables more effective processing of text data.

Traditional methods of text representation such as Bag-of-Words (BoW) [14] represent words as discrete symbols, while those methods are simple and effective for certain tasks, they are failing to capture the semantic relationships between words.

After the words are represented as a vector there is a way to find similarities in the words, similar words will be positioned close to each other in space, while unrelated words are positioned far apart, this is what allows algorithms to understand the semantic similarity between words. There are several popular methods for generating word embeddings, including Word2Vec, Global Vectors for Word Representation (GloVe), and FastText. These methods use different algorithms to learn the vector representations of words based on their context in the text.

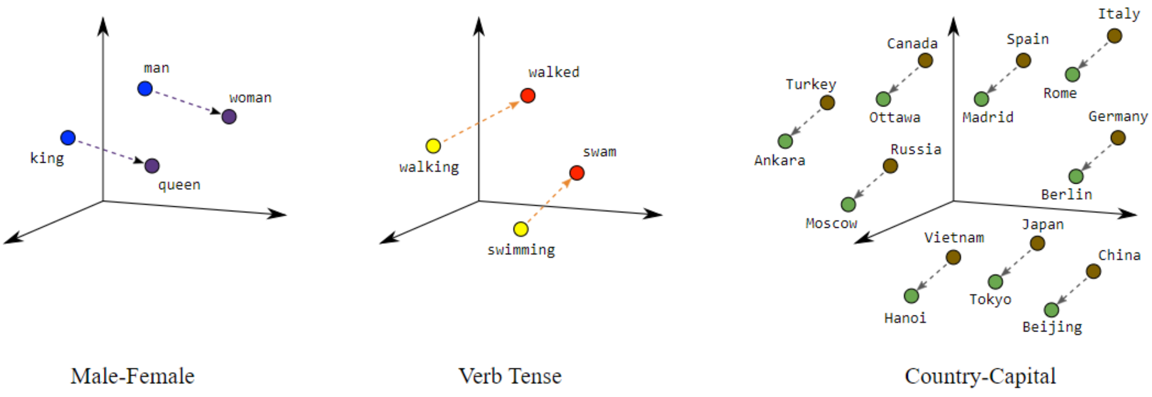


Figure 14, " The king is to the queen as man is to woman encoded in the vector space as well as the verb Tense and Country, and their capitals are encoded in low dimensional space preserving the semantic relationships."[17].[source]

3.6 CNN Bi-LSTM

The CNN-BiLSTM model is a hybrid neural network architecture that combines the strengths of Convolutional Neural Networks (CNNs) and Bidirectional Long Short-Term Memory (BiLSTM) networks. This combination allows the model to effectively process sequential data with complex patterns and dependencies.

The CNN layers are used to extract a sequence of higher-level features from the input data, those features are then fed into the BiLSTM layers, which process the feature sequence in both directions to capture complex patterns and dependencies.

The input data is convolved with a set of filters in the CNN layers to produce a sequence of feature maps. These feature maps are then passed through a non-linear activation function, such as the Rectified Linear Unit (ReLU), and pooled to reduce their dimensionality.

The result sequence of pooled feature maps is then fed into the BiLSTM layers, the forward and backward LSTM layers process the feature sequence in opposite directions and produce two sets of hidden states, these hidden states are then concatenated to form the final output sequence which can be used for later tasks, for example in a classification task, the final output can be passed through a fully connected layer and a softmax activation function to produce a probability distribution over the classes.

3.7 Transformers

The Transformer model, introduced in the paper "Attention is All You Need" by Vaswani et al, is a type of deep learning model that has revolutionized the field of Natural Language Processing (NLP). Unlike traditional recurrent neural network (RNN) models, which process sequence data in a linear manner, the Transformer model processes the entire sequence simultaneously, making it highly parallelizable and efficient for large-scale NLP tasks.

3.7.1 Transformers Architecture

Positional Encoding is important in the transformer architecture, as mentioned the transformer work is parallel, to keep the order and the connection between the input words the model uses positional encoding, which injects information about the relative or absolute position of the words in the sequence into the model, the positional encodings are added to the input embeddings before they are fed into the encoder or decoder.

Encoder is the First major component of the Transformer model, it is responsible for processing the input sequence and producing a sequence of continuous representations that capture the contextual information of each word in the sequence. The encoder consists of a stack of identical layers, each containing two sub-layers: a multi-head self-attention mechanism and a position-wise fully connected feed-forward network. The input for the first layer is the input sequence after the positional encoding, the rest layers input is the output of the previous layer. Each sub-layer also includes a residual connection and a layer normalization operation.

Self-Attention also known as scaled dot-product attention, is the core component of the transformer model, It allows the model to weigh the relevance of each word in the sequence for the computation of the representation of a given word, in other words, it allows the model to focus on the most relevant parts of the input sequence when generating the output sequence. The self-attention mechanism computes a weighted sum of input values (values) based on their relevance to each query. The relevance is determined by computing the dot product of the query with all keys, followed by a softmax operation. The queries, keys, and values are all derived from the input sequence by applying linear transformations.

Multi-Head Self-Attention Mechanism applied multiple times in parallel (that is why it is called "multi-head") with different learned linear transformations of the input sequence. This is what allows the model to capture different types of relationships between words.

Position-Wise Fully Connected Feed-Forward Network applies a linear transformation followed by a non-linear activation function (ReLU) and another linear transformation to each word in the sequence independently, this allows the model to capture complex patterns in the data.

Decoder is the second major component of the transformer model; it is responsible for generating the output sequence based on the representations produced by the encoder. The first sublayer in each decoder layer is a **masked multi-head self-attention** mechanism, which is a similar concept to the self-attention mechanism in the encoder, but with an added "mask" that prevents it from attending to future words in the sequence.

Later, the decoder is just like the encoder, it consists of a stack of identical layers, however, each layer in the decoder contains an additional sub-layer that performs multi-head attention over the output of the encoder stack.

This ensures that the prediction for each word only depends on the previous words. This allows the decoder to focus on the relevant parts of the input sequence when generating each word in the output sequence.

Applications of transformers are diverse and include machine translation, text summarization, sentiment analysis, and question answering. Additionally, transformers are used as the foundation for many state-of-the-art NLP models, including BERT (Bidirectional Encoder Representations from Transformers), and GPT (Generative Pretrained Transformer).

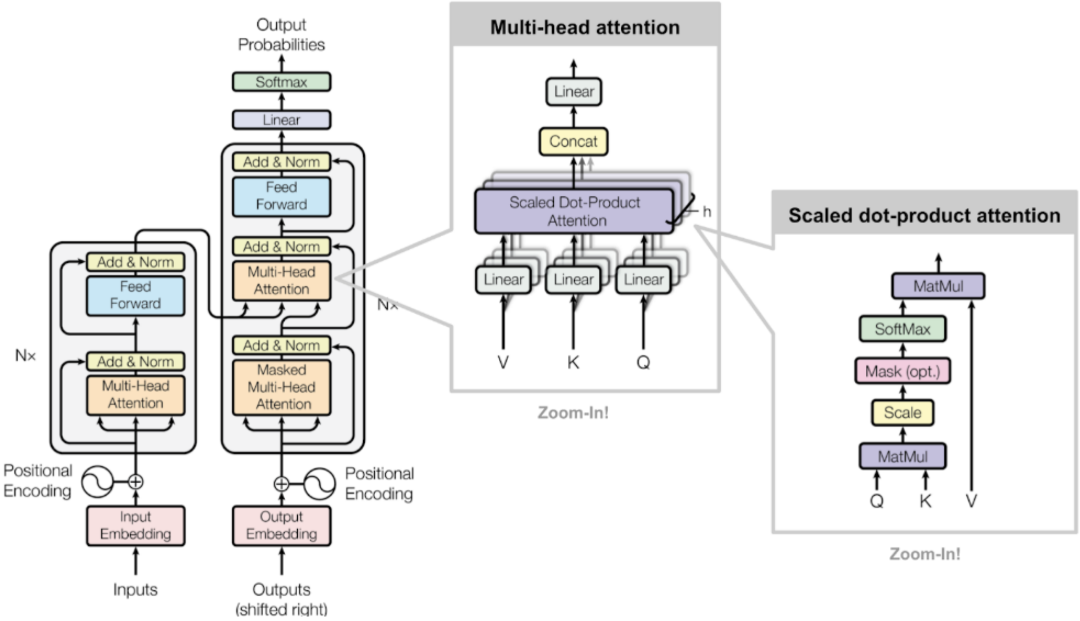


Figure 15, The Transformer - model architecture. [source]

3.8 BERT

BERT, which stands for Bidirectional Encoder Representations from Transformers, is a groundbreaking model in the field of Natural Language Processing (NLP) that has been introduced by researchers at Google in 2018. Ever since then, BERT has significantly improved the state-of-the-art performance on a wide range of NLP tasks. BERT is based solely on the encoder part in the Transformer model. The uniqueness of BERT is its bidirectional training approach, while traditional language models are either trained left-to-right or right-to-left, BERT breaks from this convention and is trained to read in both directions. This bidirectional understanding allows BERT to grasp the context of a word based on all of its surroundings (left and right of the word) and therefore have a better understanding of the logic and connection between the words in the text.

3.8.1 Training of BERT

BERT's training process is divided into two stages: pre-training and fine-tuning.

3.8.1.1 Pre-Training

In the pre-training stage BERT is trained on an unsupervised task of predicting words in a sentence, a process known as Masked Language Model (MLM), and a task of predicting whether a sentence follows another sentence, known as Next Sentence Prediction (NSP). This pre-training is performed on a large corpus of text, allowing BERT to learn a rich understanding of language, and providing a good initialization, enabling the model to learn the task with a relatively small amount of data.

3.8.1.2 Fine-Tuning

In the fine-tuning stage, BERT is adapted to a specific NLP task which involves continuing the training process on task-specific data. This fine-tuning process allows BERT to be adapted to a wide range of tasks, such as text classification, named entity recognition, and question answering

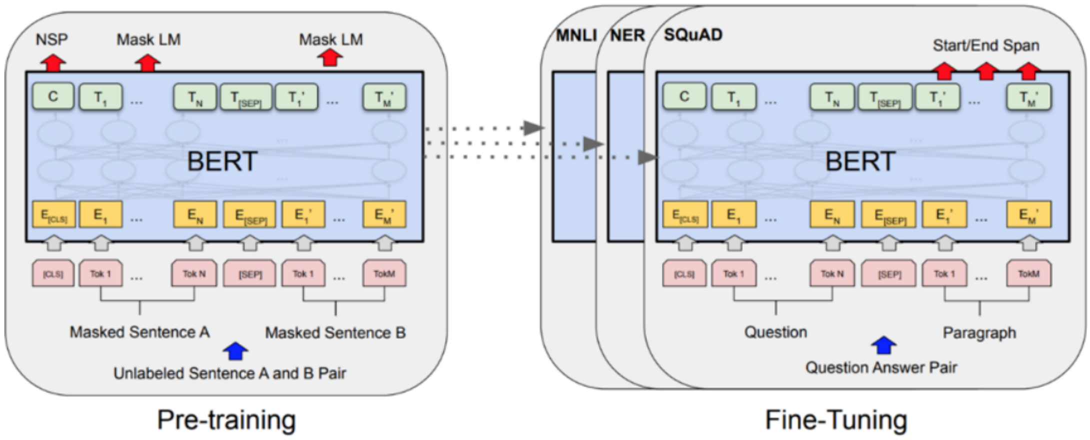


Figure 16, During fine-tuning, all parameters are fine-tuned. [CLS] is a special symbol added in front of every input example, and [SEP] is a special separator token (e.g. separating questions/answers). [source]

4. The Model - Bangla-BERT

The Bangla-BERT model is a fine-tuned adaptation of the original BERT model which was built specifically for sentiment analysis for the Bangla language. This section provides a detailed examination of the elements that together build the model: Encoder, CNN, and Bi-LTSM.

After reading this section you will understand these elements, their roles, and how they contribute to the overall functionality of the Bangla-BERT model.

4.1 Workflow Of Sentiment Analysis

In this section, we will dive into the workflow process followed to gather and prepare the data for analysis. This systematic approach ensures the reliability and quality of the dataset used in this study.

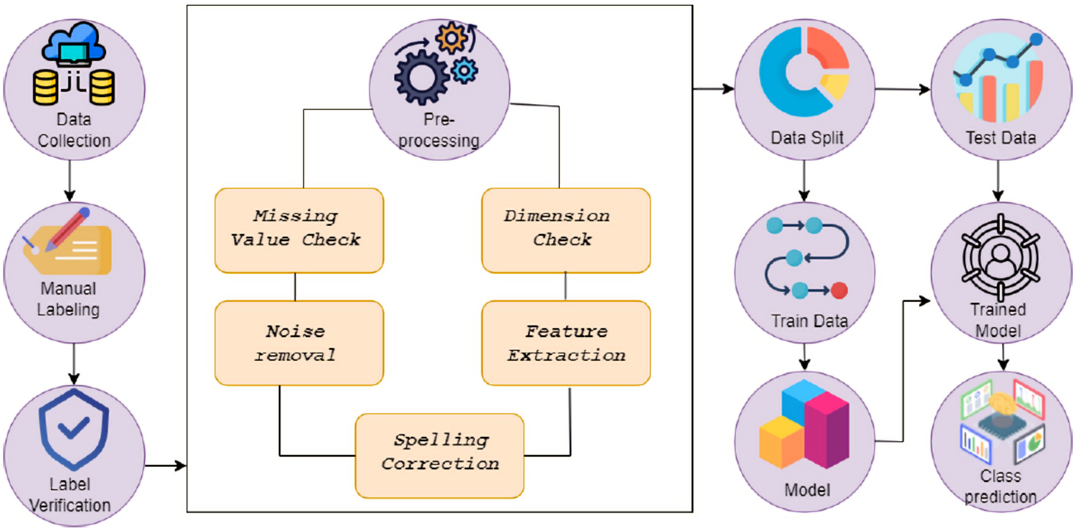


Figure 17, Whole workflow of sentiment analysis.[source]

4.1.1 Part One - Data Source, Data Collecting And Labeling

Data Source

The first step in the workflow process involves identifying suitable data sources. The researchers carefully select various sources, including online social media platforms, review websites and publicly available datasets. This diverse range of sources allows for a comprehensive collection of data, capturing different perspectives and opinions related to the research topic.

Data Collection and Labeling

Once the data sources are identified, the researchers employ appropriate techniques to collect the necessary data. This may involve web scraping, API integration, or other methods to gather textual content, user reviews, ratings, and other relevant data points. Additionally, during the data collection phase, the researchers assign labels to the data based on specific characteristics or categories pertinent to their research objective. Labeling plays a crucial role in supervised learning and model training.

4.2 Second part - Pre-Processing

Data Preprocessing

Data preprocessing is a critical step in the workflow process to ensure data quality and suitability for analysis. The researchers perform several important preprocessing steps, including missing value check, noise removal, spelling correction, and feature extraction.

Missing Value Check

The researchers carefully examine the collected data for missing values and employ appropriate techniques to handle them. This ensures that the dataset used for analysis is complete and reliable.

Noise Removal

To enhance the quality of the data, the researchers apply noise removal techniques. They identify and eliminate outliers, irrelevant variables, or any other sources of noise that may affect the accuracy of the analysis.

Spelling Correction

Given that the data consists of textual content, the researchers address spelling errors to maintain consistency and accuracy. They utilize spelling correction techniques to identify and rectify misspelled words, ensuring the integrity of the textual data.

Feature Extraction

Feature extraction is a vital step in data preprocessing. The researchers extract meaningful features from the data using techniques such as bag-of-words and TF-IDF. These features serve as crucial inputs for subsequent analysis and model development.

4.3 The Bengla-BERT Model

In this section we will explain the architecture and the models that build the Bengla-BERT model.

Representative Mechanism of Bangla-BERT

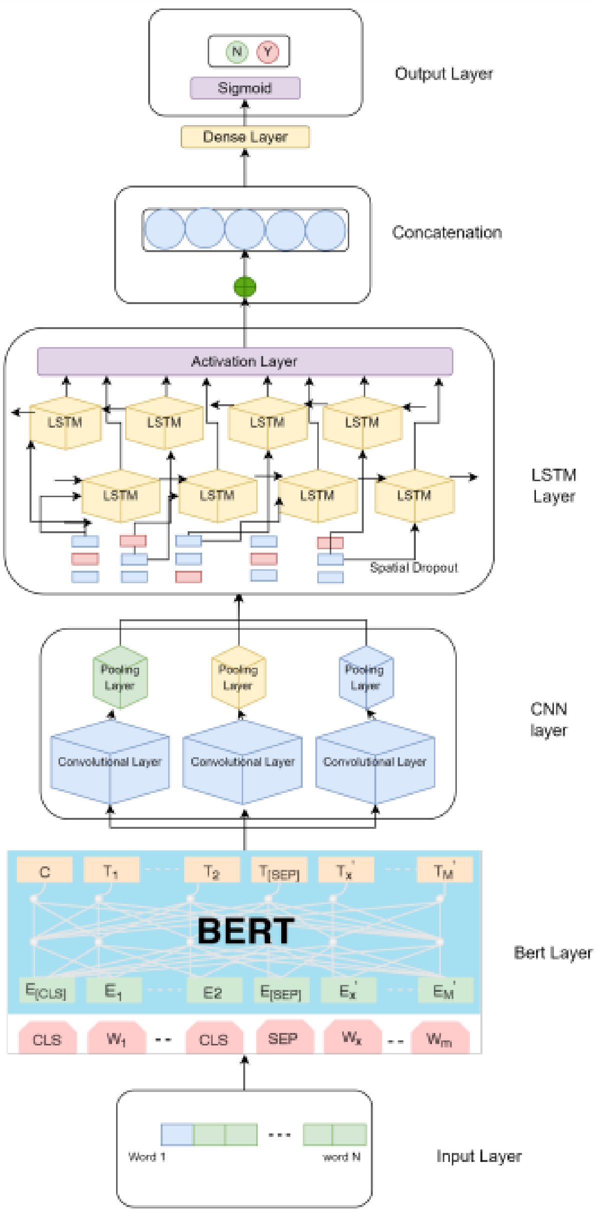


Figure 18, During fine-tuning, all parameters are fine-tuned. [CLS] is a special symbol added in front of every input example, and [SEP] is a special separator token (e.g. separating questions/answers). [source]

Input and Word Embedding

The first stage of the process involves the tokenization of the input sequence. The Bangla-BERT model uses the WordPiece tokenization strategy, meaning the input is split into words and each word gets tokenized. WordPiece tokenization strategy is particularly effective for languages like Bangla which have a rich morphological structure and a large number of unique words. The tokenization process ensures that the model can handle a wide range of words and phrases even when those words may not have been present in the training data. Once the input sequence is tokenized, the tokens are converted into vectors using a learned embedding so that the model could process in the Encoder layer.

BERT Stage and Transformer Encoder

The embedded tokens were fed into the Transformer encoder, passing through the self-attention mechanism that allows the model to weigh the importance of each word in the context of the others, capturing the intricate dependencies between words in a sentence. The feed-forward neural network further processes the attention-weighted representations, refining the output vectors.

CNN Layer

The output of the BERT stage is then passed through a Convolutional Neural Network (CNN) layer. CNN is particularly effective at identifying local and global patterns within the data, such as the arrangement of words in a sentence or the overall sentiment of a text. Through the CNN layer, the Bangla-Bert model captures the newly learned patterns and uses them to improve its understanding of the input text.

Bi-LSTM Layer

The output of the CNN layer is then passed through a Bidirectional Long Short-Term Memory (Bi-LSTM) layer. Bi-LSTM is highly effective for learning long-term dependencies in sequence data because of its ability to process the data in both directions. Word meaning can be dependent on the words that come before or after it, and capturing the true meaning of the word makes the model highly relevant in the field of language understanding. After the Bi-LTSM the model captures these long-term dependencies, further enhancing its understanding of the text.

Concatenation, Dense Layer, and Sigmoid Layer

After the Bi-LSTM layer, the model concatenates the output vectors and passes them through a dense layer. The dense layer is a fully connected neural network layer that can learn non-linear relationships between the input and output. This allows the model to capture complex patterns in the data that may not be apparent from the individual components. The output of the dense layer is then passed through a sigmoid activation function. The sigmoid function maps the output of the model to a value between 0 and 1, allowing it to be interpreted as a probability. This is particularly useful for binary classification tasks, such as sentiment analysis, where the model needs to predict whether a text is positive or negative.

Output

The final output of the Bangla-BERT model is a single value between 0 and 1 that represents the predicted sentiment of the input text, yes or no.

5. Expected Achievements

In our project, we aim to develop a robust binary classification system that can accurately determine the authorship of a given text, particularly focusing on identifying whether the text was written by Shakespeare or not. To accomplish this, we will leverage state-of-the-art technologies and techniques such as fine-tuning the BERT module and applying binary classification algorithms. By fine-tuning the BERT module on a dataset of Shakespeare's writings, we will train the model to capture the unique linguistic patterns and writing style associated with Shakespeare's works. This process will enable the model to discern the distinguishing characteristics that differentiate Shakespeare's writings from those of other authors. Once the BERT module is fine-tuned, we will proceed to feed the algorithm different texts and determine whether Shakespeare wrote the text or not. We anticipate providing a reliable and effective approach to determining the authorship of texts.

6. Research Process

6.1 The Process

Our project is divided into two main parts: Part A focuses on conducting thorough research on the assigned model, while Part B involves implementing and fine-tuning the model to accomplish the desired task of author identification.

During Part A, we commenced by immersing ourselves in an in-depth study of the provided paper [10]. Additionally, we summarized the topic at hand and extensively explored supplementary resources. To comprehend the workings of our BERT model, we delved into areas such as transforms, CNN, LSTM, word embedding, and prior models to better understand the advantages of our chosen module. Subsequently, we engaged in productive discussions with our mentors, presenting them with a concise summary and engaging in comprehensive conversations about the project's stages and our desired goals. This collaboration allowed us to outline a well-structured work plan to guide our book-writing process.

With the work plan in place, we embarked on the process of organizing and executing our writing tasks. Throughout the semester, we regularly shared sections of the book with our mentors, seeking their valuable feedback on each stage of our progress. We also posed targeted questions to gain a deeper understanding of the model and ensure our implementation aligned with our objectives.

To prepare for Part B of the project, we diligently considered the implementation of the algorithm required to solve the author identification task, drawing from our acquired knowledge of the model. This involved creating diagrams to guide our code design and developing a graphical user interface (GUI) to visualize the anticipated final product.

Upon completing the book, we shifted our focus to preparing a comprehensive presentation that encompassed our profound understanding of the model and the goals we aimed to achieve.

6.2 Use Case Diagram

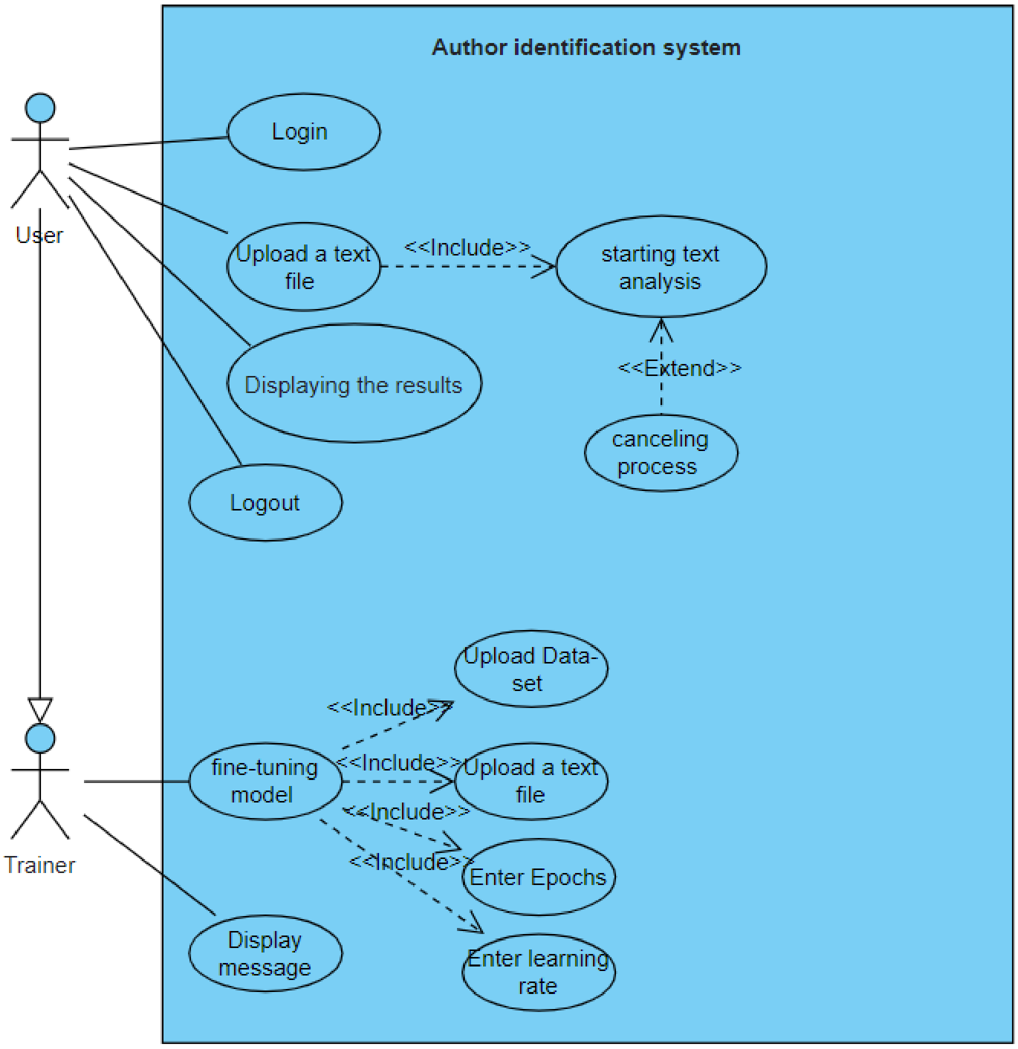


Figure 19, Use case diagram.

6.3 Activity Diagram

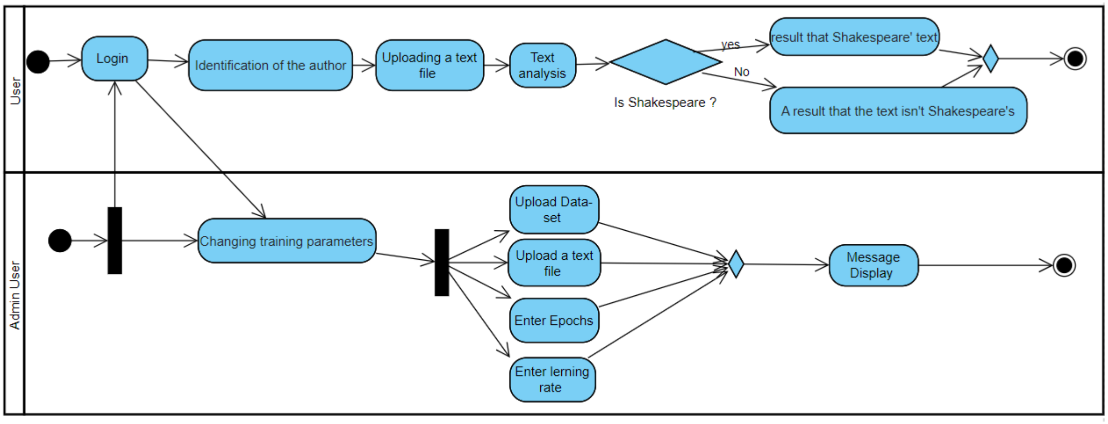


Figure 20, Activity Diagram.

6.4 User Interface (GUI Window)

Our system can be used by 2 different users: using users and admin users who train the model. That's the reason for creating a sign-in screen.

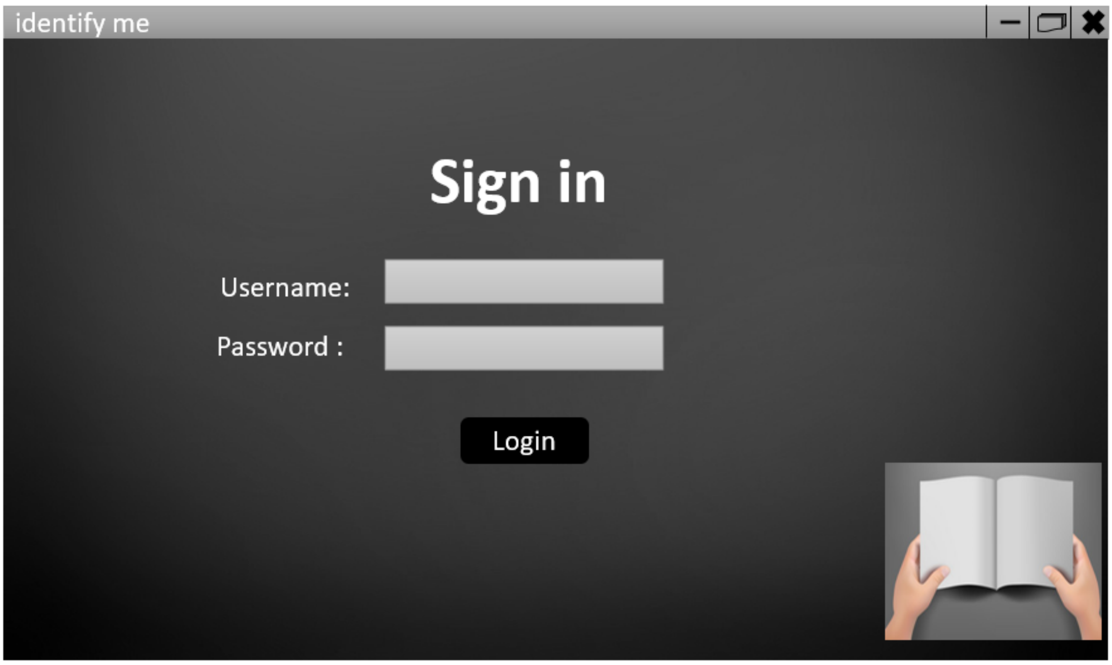


Figure 21, Login window.

Homepage of Admin type of user, who can train the model and choose settings for the training. Admin users can also use the model for text classification.

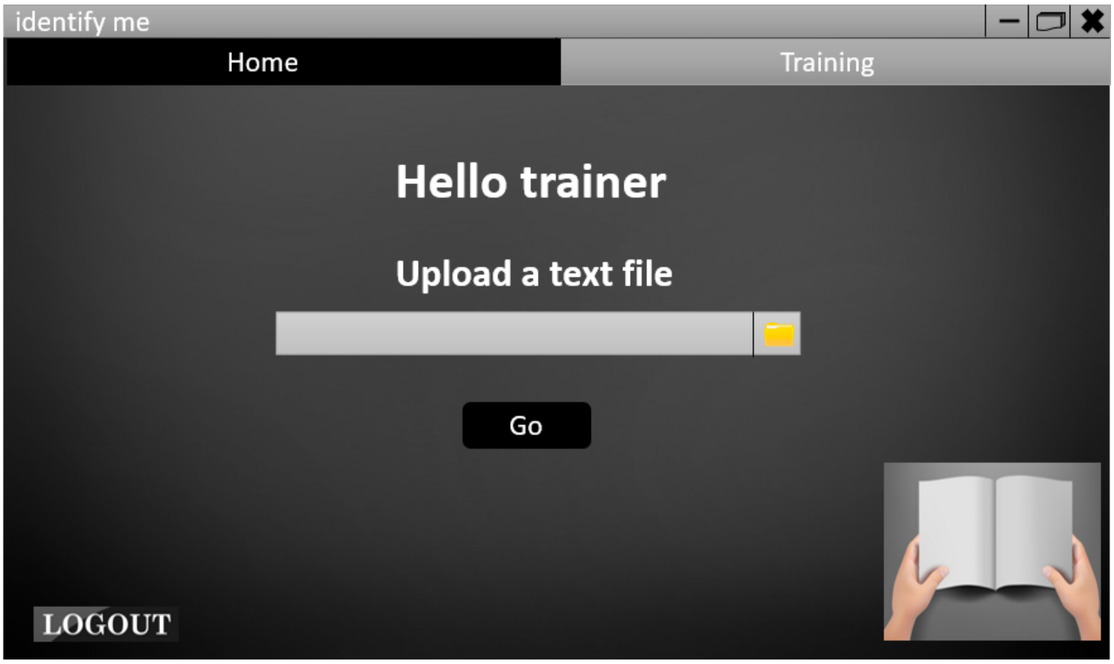


Figure 22, Admin home screen.

Homepage of a normal type of user, in this homepage the user can upload a text and then press the "Go" button which starts the process of text analysis.

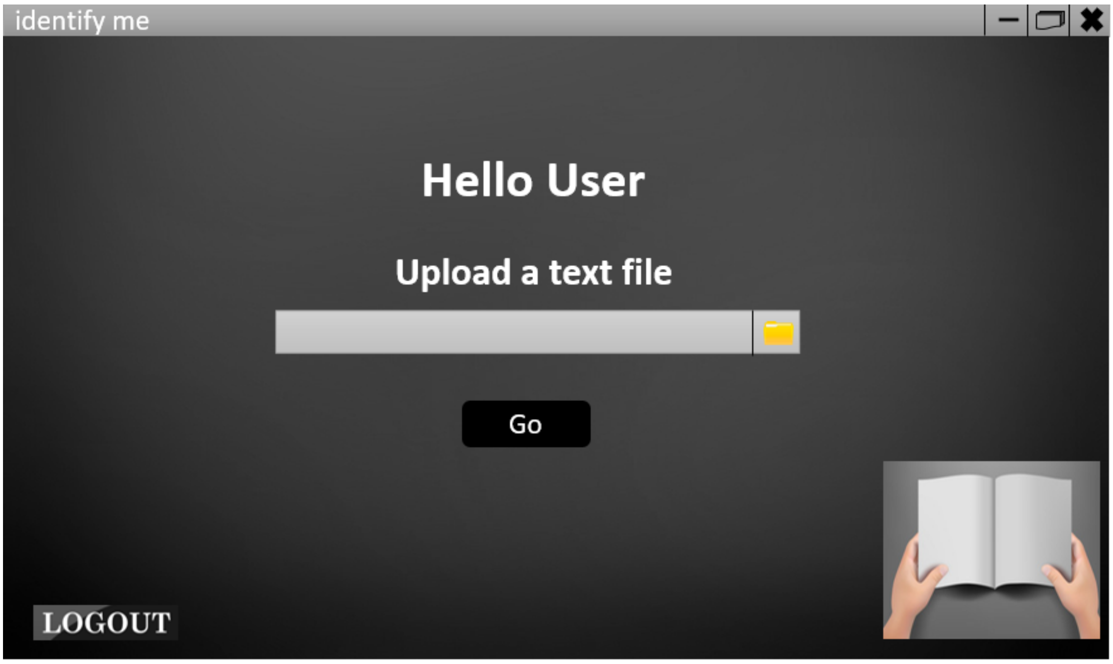


Figure 23, User home screen.

After the user pressed the "Go" button the system is processing the data, and there is a cancel button for stopping the process.

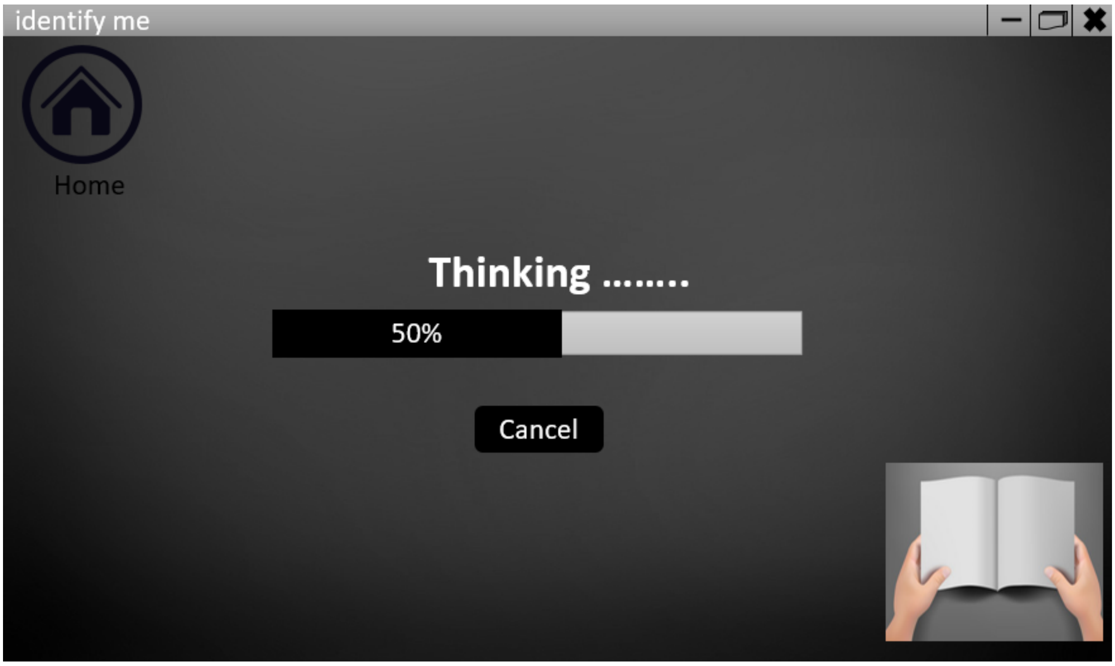
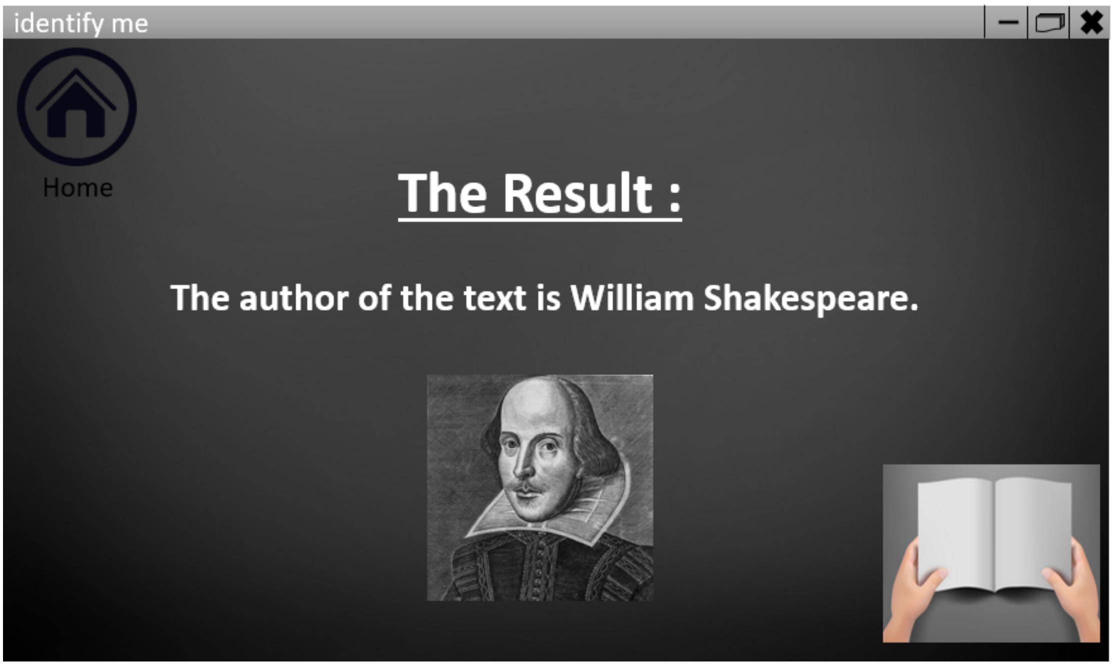


Figure 24, Text process screen.

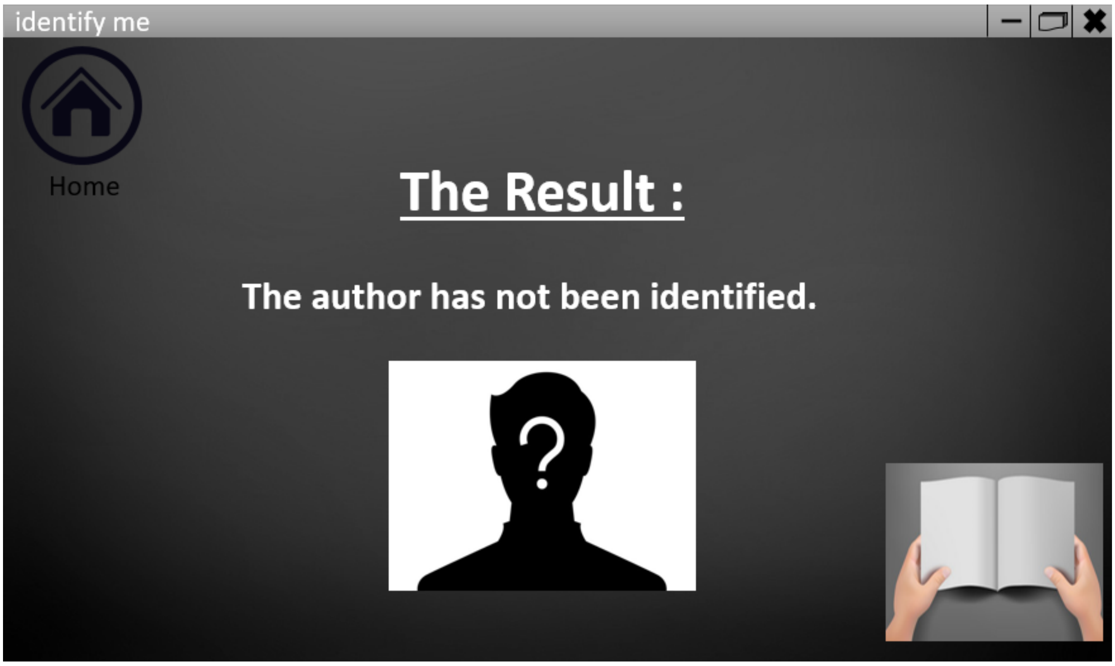
Result in a Screen for a text that the algorithm determined belonged to Shakespeare.



The author of the text is William Shakespeare.

Figure 25, Result screen for Shakespeare texts.

Result in a Screen for a text that the algorithm determined doesn't belong to Shakespeare.

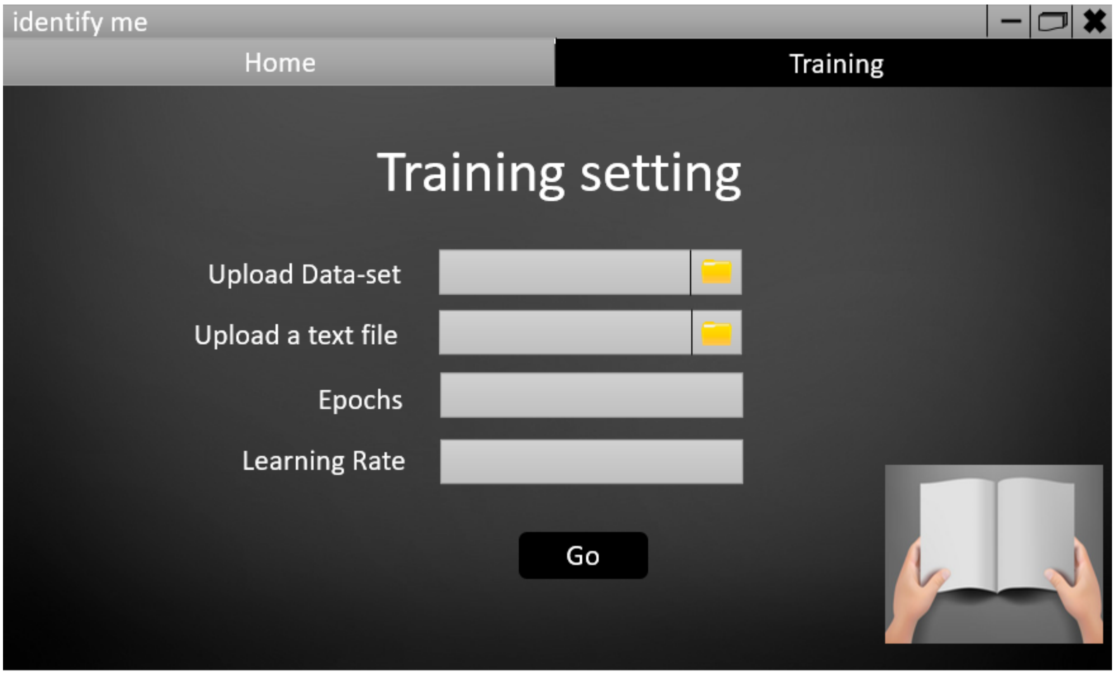


The Result :

The author has not been identified.

Figure 26, Result screen for none Shakespeare texts.

Training screen for Admin-type users, in this screen the training process setting is settled. The settings that can be decided are the number of epochs, the learning rate, uploading a dataset, and a specific text file of the labeled data.



Training setting

Upload Data-set

Upload a text file

Epochs

Learning Rate

Figure 27, Training settings screen.

7. Verification Plan

Test Number	Current Screen	Test Description	Expected Result
1	Login window	Press Login after entering the correct username and password.	Moving to the correct home screen for that user type.
2	User home screen	Upload a text file.	The file loaded successfully.
3	User home screen	Click on the Go button.	Moving into the processing window.
4	User home screen	Pressing the Logout button.	Successfully log out of the system.
5	Result screen	Text file content was written by Shakespeare.	Shakespeare's screen appears.
6	Admin Training Tab	Uploading Data Set.	The file loaded successfully.
7	Admin Training Tab	Pressing the Go button without filling one of the fields.	The error message appeared “Please fill all the fields before running the program”.
8	Admin Training Tab	Pressing the Go button after filling in all the fields.	Training message appeared.

8. Summary

In today's world, natural language processing faces significant challenges in capturing the contextual nuances and complexities of the text. However, there is a solution in the form of BERT. By leveraging transformer-based architectures and self-attention mechanisms, BERT has revolutionized the field by enabling a deeper understanding of the semantic meaning and sentiment expression in the text.

Throughout our project, we showcased the exceptional capabilities of BERT in sentiment analysis and binary classification. By harnessing its contextual understanding of the text, BERT empowers businesses and organizations to extract invaluable insights from user-generated content, customer feedback, and social media sentiments. The accurate prediction of sentiment and precise classification of data into binary categories greatly enhance decision-making and comprehension of textual information.

The integration of BERT has not only taken sentiment analysis and binary classification to new heights, but it has also made a profound impact on the broader field of natural language processing, and in our opinion, it is really amazing.

9. References

[1] Antoun, W., Baly, F., & Hajj, H. (2020). Arabert: Transformer-based model for arabic language understanding. arXiv preprint arXiv:2003.00104.

[2] De Vries, W., van Cranenburgh, A., Bisazza, A., Caselli, T., van Noord, G., & Nissim, M. (2019). Bertje: A dutch bert model. arXiv preprint arXiv:1912.09582.

[3] Hossain, E., Sharif, O., & Moshiul Hoque, M. (2021). Sentiment polarity detection on bengali book reviews using multinomial naive bayes. In Progress in Advanced Computing and Intelligent Engineering: Proceedings of ICACIE 2020 (pp. 281-292). Singapore: Springer Singapore.

[4]Farahani, M., Gharachorloo, M., Farahani, M., & Manthouri, M. (2021). Parsbert: Transformer-based model for persian language understanding. Neural Processing Letters, 53, 3831-3847.

[5] Kowsher, M., Tahabilder, A., Sanjid, M. Z. I., Prottasha, N. J., & Sarker, M. M. H. (2020, August). Knowledge-base optimization to reduce the response time of bangla chatbot. In 2020 Joint 9th International Conference on Informatics, Electronics & Vision (ICIEV) and 2020 4th International Conference on Imaging, Vision & Pattern Recognition (icIVPR) (pp. 1-6). IEEE.

[6]Krishna, A., Akhilesh, V., Aich, A., & Hegde, C. (2019). Sentiment analysis of restaurant reviews using machine learning techniques. In Emerging Research in Electronics, Computer Science and Technology: Proceedings of International Conference, ICERECT 2018 (pp. 687-696). Springer Singapore.

[7]Mahtab, S. A., Islam, N., & Rahaman, M. M. (2018, September). Sentiment analysis on bangladesh cricket with support vector machine. In 2018 international conference on Bangla speech and language processing (ICBSLP) (pp. 1-4). IEEE.

[8] Masala, M., Ruseti, S., & Dascalu, M. (2020, December). Robert–a romanian bert model. In Proceedings of the 28th International Conference on Computational Linguistics (pp. 6626-6637).

[9] Ombabi, A. H., Ouarda, W., & Alimi, A. M. (2020). Deep learning CNN–LSTM framework for Arabic sentiment analysis using textual information shared in social networks. Social Network Analysis and Mining, 10, 1-13.

[10]Prottasha, N. J., Sami, A. A., Kowsher, M., Murad, S. A., Bairagi, A. K., Masud, M., & Baz, M. (2022). Transfer learning for sentiment analysis using BERT based supervised fine-tuning. Sensors, 22(11), 4157.

[11] Romanian (RobBERT) - https://pieter.ai/robbert/

[12] Singla, Z., Randhawa, S., & Jain, S. (2017, June). Sentiment analysis of customer product reviews using machine learning. In 2017 international conference on intelligent computing and control (I2C2) (pp. 1-5). IEEE.

[13]Souza, M., & Vieira, R. (2012). Sentiment analysis on twitter data for portuguese language. In Computational Processing of the Portuguese Language: 10th International Conference, PROPOR 2012, Coimbra, Portugal, April 17-20, 2012. Proceedings 10 (pp. 241-247). Springer Berlin Heidelberg.

[14] Zhang, Y., Jin, R., & Zhou, Z. H. (2010). Understanding bag-of-words model: a statistical framework. International journal of machine learning and cybernetics, 1, 43-52.

[15]https://machinelearningjourney.com/index.php/2020/08/07/vanishing-and-exploding-gradients/

[16]https://medium.com/@shrutijadon/survey-on-activation-functions-for-deep-learning-9689331ba092

[17] https://towardsdatascience.com/word-embeddings-for-nlp-5b72991e01d4