TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

THAPATHALI CAMPUS

Minor Project Mid-Term Report

On

Fake News Detection using BERT Model

Submitted By:

Abhishek Chaudhary (THA076BCT003)

Adhip Bhattarai (THA076BCT004)

Kshitiz Poudel (THA076BCT018)

Nandan Singh (THA076BCT020)

Submitted To:

Department of Electronics and Computer Engineering

Thapathali Campus

Kathmandu, Nepal

February 2023

TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

THAPATHALI CAMPUS

Minor Project Mid-Term Report

On

Fake News Detection using BERT Model

Submitted By:

Abhishek Chaudhary	(THA076BCT003)
Adhip Bhattarai	(THA076BCT004)
Kshitiz Poudel	(THA076BCT018)
Nandan Singh	(THA076BCT020)

Submitted To:

Department of Electronics and Computer Engineering

Thapathali Campus

Kathmandu, Nepal

Under the Supervision of

Mr. Kshetraphal Bohara

February 2023

ACKNOWLEDGEMENT

We would like to express our thanks and gratitude towards the Institute of Engineering, Tribhuvan University for motivating us in doing this project by including a minor project in the course of Bachelor in Computer Engineering.

Also, we would like to thank the Department of Electronics and Computer Engineering for providing the opportunity and guidance for their valuable guidelines in the course of this project.

In addition to this, we would like to offer our sincere gratitude to our supervisor Mr. Kshetraphal Bohara for his valuable guidance, feedback, and resources that were critical to the implementation of the project.

Abhishek Chaudhary (THA076BCT003)

Adhip Bhattarai (THA076BCT004)

Kshitiz Poudel (THA076BCT018)

Nandan Singh (THA076BCT020)

ABSTRACT

The objective of the project was to classify text inputs containing inaccurate information using a pre-trained BERT model. Google's BERT model was a state-of-the-art natural language processing tool, pre-trained on vast amounts of text data and applied to both similarity checks and fake news detection. Due to the scarcity of credible fake news detectors for the Nepali language, a web-based graphical user interface application was established to detect fake news in Nepali text news. The procedure involved searching for the provided title in Google, scraping the resulting titles and URLs, and performing a similarity test between the provided title and the scraped titles. The news source was examined for reliability if the similarity score exceeded the established threshold value of 0.7 for cosine similarity and 0.3 for Jaccard similarity. The news was deemed authentic if it was from a reliable source. If the similarity score was below the threshold value or the source was unreliable, the BERT model was utilized to classify the news as genuine or fake. The BERT model was trained using a pre-processed, tokenized Nepali text news dataset, which was then fine-tuned using hyperparameter tuning. The model's performance was evaluated using accuracy, precision, recall, and F1 Score metrics to determine the authenticity of Nepali text-based news.

Keywords: BERT, Fake, news, online, website

Table of Contents

List of Figures

List of Tables

List of Abbreviations

ADAM:	Adaptive Moment Estimation
BERT:	Bidirectional Encoder Representations from Transformers
CSS:	Cascading Style Sheets
GLUE:	General Language Understanding Evaluation
GUI:	Graphical User Interface
HTML:	Hypertext Markup Language
LSTM:	Long short-term memory
TPU:	Tensor Processing Unit
NLP:	Natural Language Processing
NLTK:	Natural Language Toolkit
RNN:	Recurrent Neural Network
SVM:	Supported Vector Machine
URL:	Uniform Resource Locator

1. INTRODUCTION

In today's world majority of the people rely on the news from social media platforms instead of traditional platforms. This variation is because it costs more time and money to rely on news from traditional platforms rather than social media. People propagate fake news on social media for the sole purpose of self or political gain. So, it is safe to say that despite the benefits provided by social media in our daily life news provided by social media is less reliable than traditional ones like newspapers.

Since the internet and smart devices have made it easier to assess the news on social media platforms, the spreading of fake news through these platforms has also increased. Hence, it is safe to assume that the news on social media has a large influence on society and some people can take advantage of this fact. There are websites and social media platforms which publish fake news, propaganda, and misinformation pretending to be real news. Sometimes, these platforms and websites also publish news that is half true and half false. Most of the time, this fake news aims to affect the general public opinion on various matters. Politicians also circulate fake news to change public opinion so that they can win the election. Some fake news is created just to trigger people's distrust and make them confused. It also casts hate on people's minds. Even Forbes magazine has published an article providing a list of fake news websites[1].

Fake news is not a new phenomenon. People often fall victim to fake news because it divides people into two or more groups with each group claiming that their opinion towards a certain agenda is legitimate and correct. Also, since people are not the real-time witness of an incident, they blindly trust the source which provides them the information about the incident. In addition to this, often people's emotions are targeted which reduces their rational thinking power.

There are various types of fake news:

- Clickbait
- Sponsored content
- Fabricated journalism

A typical characteristic of fake news is that it challenges people's belief and try to affect their opinion. They are too good to be true and have grammatical mistakes. They are not published by a reliable source either. They are often the news without facts or misleading or false facts. They are often reported by unreliable people. They may even contain a picture made using editing software to look like a real picture. They are often grammatically incorrect.

1.1. Motivation

During the time of the election in Nepal during Mangsir month, there were many false claims which were spread to gain political influence. As the election is crucial for democracy, we started to feel that this misinformation must be stopped at all costs. Even online Khabar published an article regarding this fake news in the election[2].

1.2. Problem Definition

Since false news may spread disinformation, damage reputations, and even inspire violence, the issue of fake news in the Nepali language needs to be addressed. Due to fake news circulation, information has become difficult to verify because of the popularity of social media, which makes it simpler for false information to propagate quickly and makes it challenging for the general people to confirm the veracity of the news. Many individuals in Nepal may lack the knowledge and resources necessary to evaluate news sources critically and spot fake news, leaving them open to the spread of misleading information.

Given a piece of news, the problem is to determine whether it is real or fake. A real news article presents accurate and verifiable information, while a fake news article presents false or misleading information. The goal is to develop a model that can accurately classify a given news article as real or fake based on its content. The model should be able to handle a variety of different writing styles and language structures and should be able to effectively capture the meaning and context of words and phrases in the news article. The model should also be able to handle long-range dependencies, as the context of a statement may be relevant to determining its veracity.

1.3. Project Objectives

The following were the objectives of our project: -

- To classify /distinguish real and fake news from Nepali text using a pre-trained BERT model

- To implement the system into use on the web and make it broadly accessible.

1.4. Scope/ Applications of Project

In this age of the internet and social media, fake news spreading is a real issue. And there are not so many fake news identifiers for Nepali news. This fake news detection web application will efficiently detect whether Nepali text-based news is true/false.

For this project, we used a pre-trained BERT model that will further be trained on Nepali text-based real and fake news.

For training our model, we have translated English Fake news to Nepali using Google-translate. so, our model might be inefficient to predict fake news. Also, our model is trained on a few thousand datasets only so accuracy may become the issue.

The application of this project is as follows:

- Media and Journalism: Media and journalism: maintaining the credibility and integrity of journalism through spotting false information in news stories, social media, and other sources.

- Political campaign: identifying and countering political disinformation and propaganda spread during elections and campaigns.

- Business: identifying false information in marketing, advertising, and financial reporting to uphold accountability and transparency.

2. LITERATURE REVIEW

In research conducted by Leo Breiman and Adele Cutler[3], they published an article about Random Forest for the first time in 2001 in a research paper named "Random Forests". According to them, random forest is an ensemble learning method for classification and regression that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.

Long Short-Term Memory (LSTM) is a type of Recurrent Neural Network (RNN) that can process sequential data and make use of long-term dependencies. It was specifically designed to overcome the vanishing and exploding gradient problems that can occur when training traditional RNNs, and it has been successful in a wide range of applications including language translation, image captioning, and speech recognition. An LSTM network is composed of "cells" that contain information from the past and pass it on to the present and future. These cells are controlled by three "gates" that decide what information to store, what information to throw away, and what information to pass on. This allows the LSTM to selectively remember and forget information, making it well-suited for tasks that require the retention of long-term dependencies.

BERT[4]stands for Bidirectional Encoder Representations from Transformers is a transformer-based architecture for natural language processing tasks such as language translation, question answering, and language modeling. It was developed by researchers at Google and has been shown to outperform many previous models on a wide range of tasks. The key innovation of BERT is that it is a "bidirectional" model, meaning that it takes into account the context both to the left and the right of a given word. This is in contrast to previous models, which typically only considered the context to the left of a word. This bidirectional approach allows BERT to better understand the meaning of a word in the context of the entire sentence, rather than just based on the words that come before it. BERT is trained on large amounts of data and can learn the structure of the language in an unsupervised manner.

2.1. Previous works

In research performed by Kelly Stahl, it uses Network Analysis, Linguistic Cues, Fact-checking, Naïve Bayes Classifier, SVM, and Semantic Analysis. They used both Naïve Bayes Classifier and SVM. They found the accuracy of the combined methods was more than the accuracy of the individual methods. They found that the biggest drawback of the Naïve Bayes classifier was that it deems all features of a document, or whichever textual format is used, to be independent even though most of the time that is not the situation[5].

Jiawei et al. Conducted a study based on actual textual information, authorship, and article-subject relationship. They introduced the FAKE DETECTOR framework which combines representation feature learning and credibility label interface to compose a deep diffusive network model. They were able to achieve an accuracy of 0.63[6].

Also, Gowthami. K et. al used SVM and Random Forest to identify fake news. At first, they collected the dataset and then pre-processed the data. They applied SVM and Random Forest algorithm and they compared the results and accuracy[7].

In research conducted by Yesugade, et al. they collected scrap data and transformed the data into the format. With the help of the NLTK toolkit, they removed the stop words. Then they performed word embedding and the word index of the tokenized dataset was generated. In addition to this, they compared RNN-LSTM and sigmoid validation. Here, they used 100 neurons in each layer with each layer using the sigmoid activation function. They classified the news as fake news – 0, and real news – 1[8].

Uma et al. used combinations of different algorithms like static search, dynamic search, and URL search in the classification of fake News. The static search included different machine learning algorithms Like Naive Bayes, Random Forest, and Logistic Regression. Dynamic search asked the users to enter specific keywords in a news and produced a percentage probability of truthfulness of an article by comparing articles with similar keywords on the internet.URL search method accepts a specific website domain and calculates the authenticity of the website compared to the databases of the websites like LIAR, BuzzFeed, and BS Detector. They were able to get up to 80 percent

accuracy with the Logistic regression model and 92.73 percent accuracy with passive aggressive classifier[9].

In research conducted by Sastrawan et. al, about the Detection of fake news using deep learning CNN–RNN-based methods they conducted the research in 2 phases i.e., the training phase and the testing phase. The first phase begins by retrieving the training data from the database. The data cleansing is performed on the data. Then they performed a data augmentation process to the cleaned data to balance the data between the classes. This augmented data is then processed and transformed into word vectors. These word vectors are used to train the deep learning model. This model is stored in the database for the next phase. In the second phase, the trained model is evaluated. At first testing, data is pre-processed and the previously stored model is taken from a database. Now, the pre-processed data and saved model are used to predict the pre-processed test data and the results are displayed[10].

In research conducted by Farokhian et. al, at first, they extracted the headlines of the news and fed it to the BERT network. On the other hand, using the Max-Worth algorithm, the most appropriate news text span is selected from the rest of the news text. This text span is again fed to another BERT network. The output from both BERT networks was joined together and fed to the dropout layer. Then the output from the dropout layer was fed to the dense layer which classified the news as real or fake[11].

In research conducted by Dam et. al, they detected click-bait in the Nepali language using SVM and random forest algorithm. At first, they took the news title and news body from the dataset and performed pre-processing on the dataset. They made a collection of unique words from corresponding news titles and news-body called vocabulary. They used TDFIDF to represent the words in the form of vectors. They calculated the cosine angle between two vectors to measure how similar they were. They used SVM and random forest algorithms to detect clickbait. They obtained an accuracy of 95.03% using SVM and 94.93% using the Random Forest algorithm[12].

3. REQUIREMENT ANALYSIS

3.1. Project Requirements

3.1.1. Hardware Requirements

The hardware requirements for using BERT for fake news detection will depend on several factors, such as the size of the dataset, the complexity of the model, and the number of training and inference iterations. However, some general guidelines are:

1. A powerful CPU: A high-end CPU with multiple cores is recommended for efficient data pre-processing, model training, and evaluation.

2. A GPU: A GPU is highly recommended for training and inference as BERT models are computationally heavy. A CUDA-enabled GPU such as NVIDIA with at least 4GB of memory is recommended.

3. Memory: A minimum of 8 GB of RAM is recommended for data pre-processing, model training, and evaluation. More memory may be required depending on the size of the dataset.

4. Storage: A high-capacity storage device such as an SSD is recommended to store the dataset, pre-trained models, and trained models.

5. Network: A high-speed internet connection may be needed to download the pre-trained BERT models and to update the libraries and packages.

3.1.2. Software Requirements

The software requirements for using BERT for fake news detection in Python include:

1. Python: Python is a well-liked programming language with a sizable community and includes a ton of NLP and machine learning packages, making BERT implementation simpler.

2. TensorFlow: For certain pre-processing and assessment activities, TensorFlow is necessary.

3. Pandas: Pandas are used to clean and manipulate data. Matplotlib, Seaborn, or other visualization libraries: These libraries can be used to create visualizations of the dataset and the model's performance.

4. CUDA and cu-DNN: CUDA and cu-DNN must be installed to use a GPU. For generic processing on GPUs, NVIDIA has created the parallel computing platform and programming language known as CUDA. Deep neural network primitives are available in the cu-DNN GPU-accelerated library.

5. HTML, CSS, Javascript, Flask: To construct a website we would use this software.

3.2. Feasibility Analysis

1. Data availability:

For our BERT model which helps to understand the context of the given text, we scraped real news data from various sources like e-Kantipur, Ghorkhapatra, Nagarik, etc., and for fake news, we translated the news from Kaggle in the English language into Nepali language using Google translate. Besides this, we had a few fake news datasets scraped from clickbait sites. For source checking the user provides the data

2. Financial Feasibility:

All the libraries we need for the completion of this project are open-source. So, there is no significant cost for the completion of our project. Also, the hardware requirements are minimal, so we do not have to buy any additional hardware equipment for the completion of our project.

3. Technical Feasibility:

Currently, we have a laptop with the following specifications:

i. Processor: 11th Gen Intel® Core™ i7-11800H @ 2.30 GHz(16 CPUs) ~ 2.3 GHz

ii. RAM: 16 GB

iii. Storage: 512 GB SSD

iv. Graphics: NVIDIA GEFORCE RTX 3050 Ti (4 GB memory size)

v. Network: We have a fast and stable internet connection on our laptops.

For software requirements, we have installed all the necessary packages required for the completion of our project.

4. Operational Feasibility:

Since the project will have an easy-to-use GUI, any person without technical knowledge related to machine learning can access it. Our system will predict the truthfulness of news with a machine learning model and using source checking, but all the details will be hidden from the end user. The end user will just have to provide the news title and text. Therefore, almost anyone can operate the system. So, we can say that our system is operationally feasible.

4. SYSTEM ARCHITECTURE AND METHODOLOGY

Our system classifies news based mainly on the context of the news and by checking similar news on other sources on the web. This will ensure that our prediction will be very reliable. For a semantic understanding of news articles, we have used the BERT multilingual pre-trained model. BERT (Bidirectional Encoder Representations from Transformers) as its name suggests is an architecture using only the encoder part of transformers. BERT consists of a stack of encoder layers. where each encoder layer consists of multi-head attention and a feed-forward neural network.

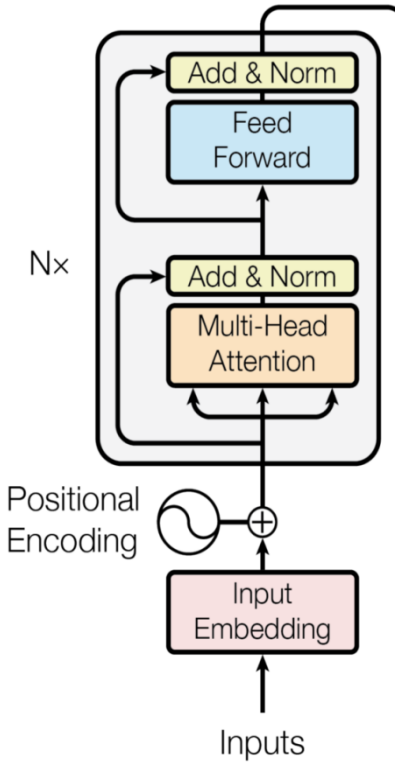


Figure 4-1. Encoder architecture

Such a model has been proven to be very good for NLP tasks because it allows bidirectional context understanding. This means that any word in input text is not only analyzed concerning words before it but also after it. This allows BERT to capture the meaning and context of words in a sentence more effectively. On the other hand, the whole input can be passed at once instead of sequence to sequence. This helps in better utilization of GPU which is designed for parallel processing. Also, BERT can handle long-range dependencies and attend to specific parts of the input, which are important for understanding the context of a statement and determining whether it is true or false.

The training of BERT is for understanding the semantics between input text and predicted output i.e., real or fake. Since the BERT model is already pre-trained in the Nepali language, we can say that it understands the relationship between different Nepali words, and how the combination of them can generate a certain meaning. The main purpose of training the architecture on real and fake news datasets is to let the model understand certain features of fake news and real news. With the semantic understanding of the Nepali language, the BERT model will generate vectors for given news text. Our model aims to capture the underlying patterns between these vectors of real and fake news that will distinguish them.

Only the context or semantics of a news article is not enough to fully classify the news. For example, some news about events may be true today but not after a particular amount of time. The news of someone being prime minister may have been true before but the same person cannot be true today. The semantics of such news does not change but the truthfulness may change. To overcome this problem, we will compare the given input news article with similar news articles scraped from different Nepali news sources. The algorithm for this method looks like this:

Step 1: Input the news title and description from the user.

Step 2: With the given title, google search for similar news/articles.

Step 3: Scrape the news title and URLs obtained from step 2.

Step 4: Compute the similarity between the input news title and the list of news titles obtained from step 2, using cosine similarity.

Step 5: Create a set ‘S’, that contains the trusted Nepali newspaper URL

Step 6: Classify the news as true and Fake.

Step 6.1: True if the similarity is greater than the threshold value and the news is from a trusted newspaper

Step 6.2: If 6.1 fails, then run the BERT model on the description of the input and check if the news is true based on the context

4.1. Block Diagram

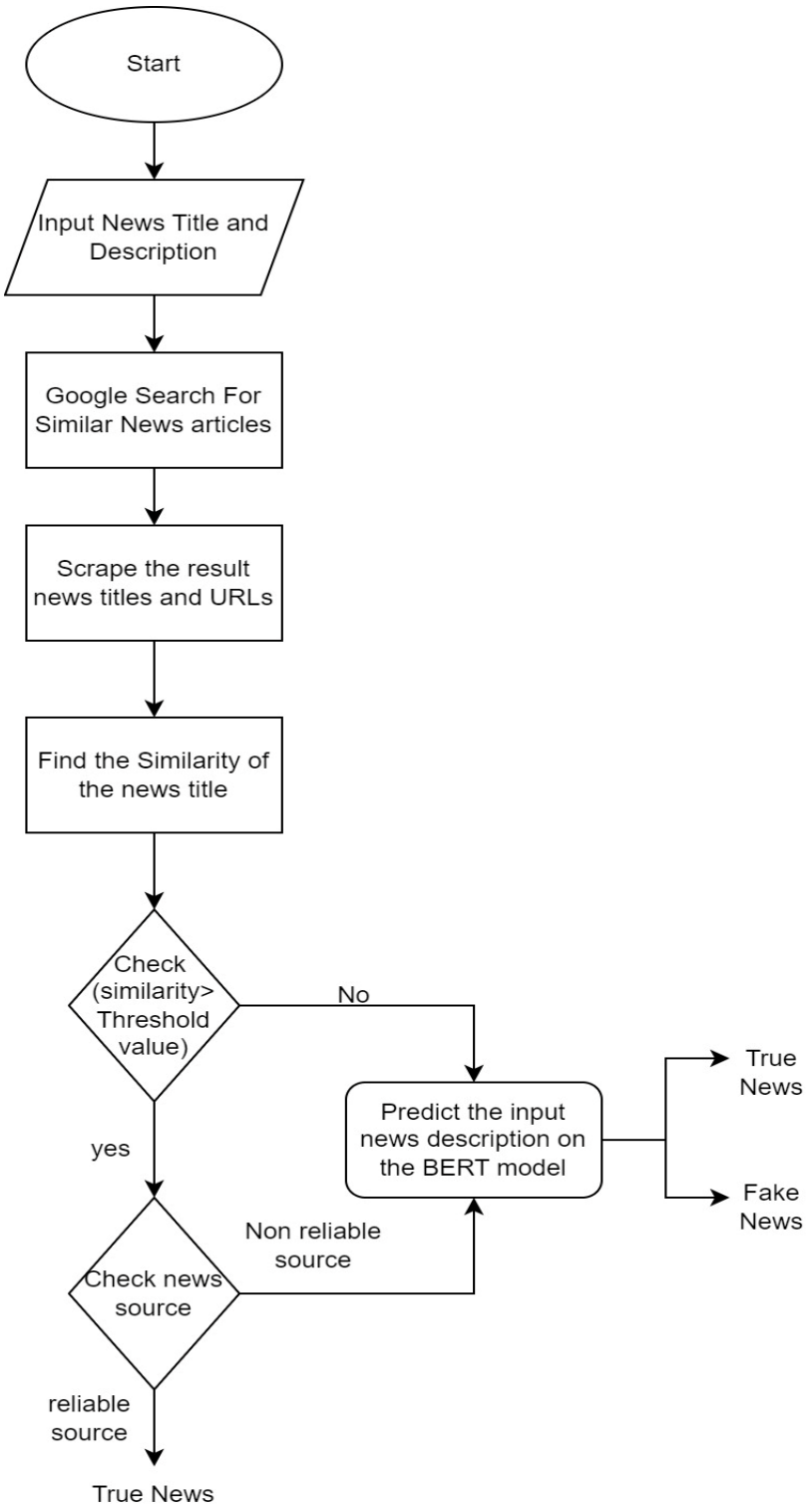


Figure 4-2. Proposed methodology

4.2. Description of the Working Principle:

This Fake news detection has two features to detect news as true/fake. Firstly, we use source checking of the news by web scrapping method.

1. Web Scrapping Method:

This method is straightforward and it doesn't require any training or testing data.

Input news titles and descriptions are taken from the user. With the given title, we import Google search in python and scrape all the related news articles.

Then we perform similarity checking for the user-given title and title obtained from a search using the BERT model and/or tf-idf. For this, we use 2 metrics for checking similarity; cosine similarity and Jaccard similarity.

- Cosine similarity:

This metric gives a value between -1 and 1 for BERT embedding vectors and values of between 0 to 1 for vectors generates using tf-idf value of 1 denotes that the vectors are exactly similar. A value of -1 shows that they are exactly the opposite. More the value, the more the similarity.

$$similarity = \cos(\theta) = \frac{A . B}{|A||B|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}} \dots \dots \dots \dots \dots (1)$$

- Jaccard similarity:

The Jaccard similarity checks if two texts are similar by directly comparing the words. So, it doesn't need the text to be converted into vector form. The value ranges from 0 to 1. 0 meaning no similarity and 1 meaning the same words used.

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots .. (2)$$

Where,

J= Jaccard distance

A= text1

B= text 2

If the cosine similarity is greater than 0.7 and the Jaccard similarity is at least 0.3 then we can say that the two texts are similar. After checking the similarity of the news titles, we check if the news is from reliable sources such as eKantipur, gorkhapatraonline, Setopati, etc. If the news is similar and from reliable sources then we called as true. Otherwise, we implement our BERT model to test the given text as real or fake.

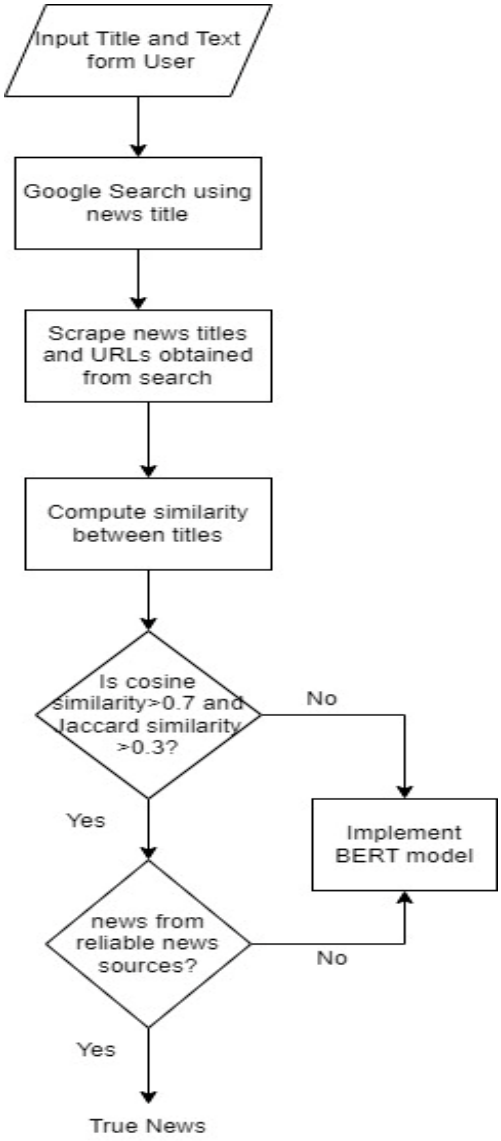


Figure 4-3. Flowchart for Web Scrapping Method

2. BERT Model:

Sometimes, we couldn't find social media news on google and the news might be from local news sites, not from reliable sources such as eKantipur, setopati, ratopati, etc. But this news can be true so for that we used the BERT model that classifies the news as fake/real based on the context of the news.

The working detail of training of Bert model with extra classification dense and dropout layers to classify news based on the semantics of the Nepali language is described below:

- Data collection and Pre-processing: fake news which we have collected from various sources are labeled as 0 or 1 i.e., Real news-0 and fake news-1. Each example either real or fake contains a title and description of a single Nepali news in Nepali text. We combined the title and description into a single column. The title and description are separated with SEP tokens as with any other sentence. Since the Nepali text is not suitable as input directly to the model. we have to modify or preprocess it. For pre-processing, we performed the following steps:

    - At first, we checked if any value is null in the dataset. If the null value was present, we removed that entire row from the dataset.

    - Then we deleted other unnecessary columns like date, author, etc. from the dataset.

    - After this we removed punctuations like ':', '?', '&', etc. from our dataset. We performed this process because we have a limited number of tokens available for giving input to the model. So we need to remove those characters from the text that does not have a significant meaning which will efficiently use the available number of tokens.

- In addition to this, we also performed the removal of stop words like ‘उनले’, ‘का’, ‘छ’, etc. from our dataset. This process is also performed for the same reason.

- Then we performed stemming on our dataset and generated stemmed text for input data.

- Test-train split: Here, we split the dataset into train data and test data in the ratio 8:2. Then again split the training data again into training data and validating data.

- Tokenization: First, we convert all the words into their corresponding tokens using the Bert preprocess. Every statement starts with CLS Token and ends with a SEP token. All unknown tokens are replaced by UNK tokens.

- Define model architecture: Here, we defined our model architecture which consists of BERT and additional classification layers.

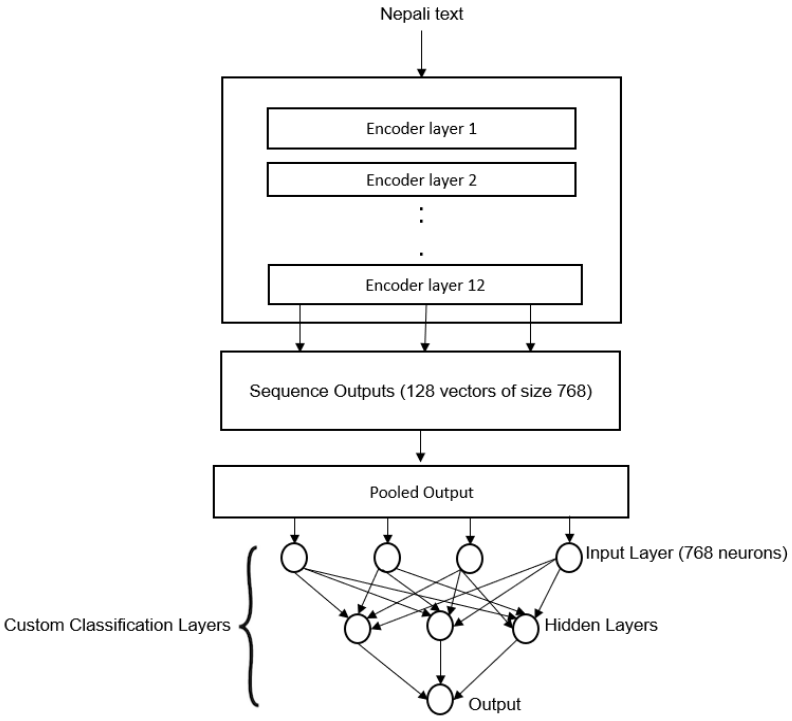


Figure 4-4. Model Architecture

We have used binary cross-entropy to calculate the cost as we only have two classes. The cost function is given as:

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \,.\, log\big(p(y_i)\big) + (1 - y_i)\,.\, log\,(1 - p(y_i)) \ldots\ldots (3)$$

Here, y is the actual output, and p(y) is the probability of y predicted by our system. Since the fake news class is labeled as 1 and real news is labeled as 0, if the actual model output is 1, the second term will be canceled and cost calculated only by the first term i.e., $y_i \,.\, log\left(p(y_i)\right)$ which becomes $log\left(p(y_i)\right)$ since y_i is 1. Now, if the model's prediction is closer to 0, the term $log\left(p(y_i)\right)$ becomes very high which signifies that the closer the model's prediction to actual output lesser the cost. It works similarly if the actual output is 0. Also, for the optimizer, we used the Adam optimizer. The optimization function is given as:

$$w_{t+1} = w_t - \alpha_t m_t \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (4)$$

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1)\left[\frac{\delta L}{\delta w_t}\right] \ldots\ldots\ldots\ldots\ldots\ldots\ldots (5)$$

Where,

m_t = aggregate of gradients at time t [current] (initially, $m_t = 0$)

m_{t-1} = aggregate of gradients at time t-1 [previous]

W_t = weights at time t

W_{t+1} = weights at time t+1

α_t = learning rate at time t

∂L = derivative of Loss Function

∂W_t = derivative of weights at time t

β = Moving average parameter

In the pre-trained BERT multilingual model, the data is passed through the embedding layer. It converts each word in the input to a vector known as ‘word embedding’ that captures the meaning and context of the word. These word embeddings are combined to form a fixed-length vector representation called ‘sentence embedding’ that captures the meaning and context of the sentence. These embeddings are passed to an encoder that understands the meaning and context of the text. The encoder is a stack of transformer layers that use a self-attention mechanism to weigh the importance of each word in the input text and a feed-forward neural network to extract complex features from the input text. The self-attention mechanism allows the model to understand the relationships between words and their context, which is important for accurately detecting fake news. The feed-forward neural network then processes the representation created by the self-attention mechanism, allowing the model to extract more complex features from the input text. Then we added dense and dropout classification layers to the BERT model.

The input to the neural network for classification is the output vector from the BERT. BERT outputs 768-sized vectors corresponding to the number of tokens. We only need the pooled output which is the final vector from the last encoder layer. Therefore, there are 768 neurons in the input layer. The hidden layers are added with dropouts in some layers.

Dropout regularization is performed by randomly dropping 10% of the neurons in the hidden layers. This technique helps to prevent over-fitting because the final output cannot be heavily affected by any single neuron to make every prediction close to the actual output.

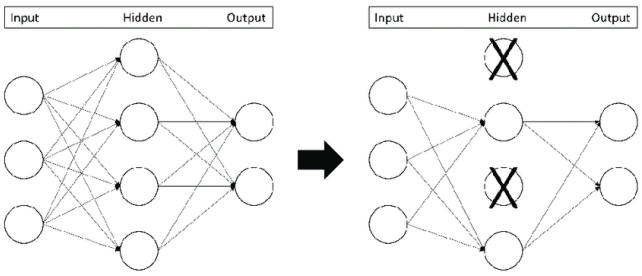


Figure 4-5. Dropout Regulation

- We then trained our model in training data and user testing data to test our model. For training, we imported “bert-multilingual-uncased” as a pre-trained BERT model in the Nepali language. We also evaluated performance matrices and confusion matrices.

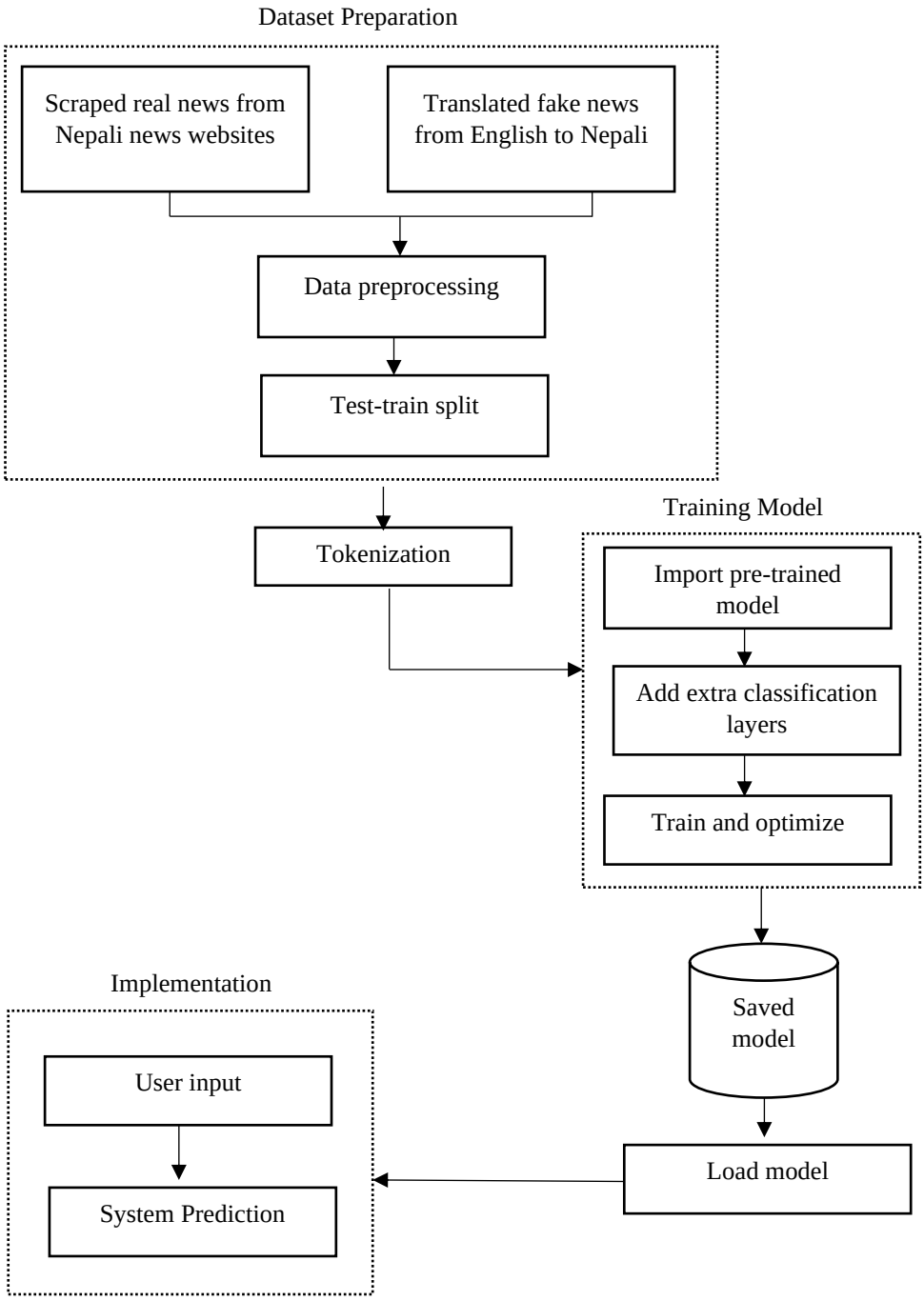


Figure 4-6. Fake news detection using BERT

5. IMPLEMENTATION DETAILS

5.1. Web scrapping Method:

This method is used for source verification of social media news. We search the news title on Google and find the similarity of user input news with the results from searching. If we find similar news, then we check if the news is from reliable sources. If both conditions are true then we classify the news as real news.

If the news is not from reliable sources, we cannot say the news is not real. So, we used user Text input and run on a trained BERT model which classifies the news based on its context.

5.2. BERT Modeling:

5.2.1. Data Collection

For the implementation of this project, we scraped about 3432 real news from various websites like Gorkhapatra, onlinekhabar, ekantipur, etc. For the fake news dataset, we translated about 2889 English fake news data to Nepali using google translate. English fake news dataset was downloaded from Kaggle.

5.2.2. Data preprocessing

The dataset required for our project is Nepali news in text form. A single news article consists of a title and description. Other data like data, sources are not so relevant for our model since our model is trained to understand the overall context of the text. We evaluated our model on both stemmed and non-stemmed pre-processed Nepali text. Surprisingly, the accuracy of non-stemmed pre-processed data was more than that of stemmed pre-processed data.

5.2.3. Tokenization and embedding

The model is trained on the description and title of a news article. We used bert_multi_cased_preprocess/3 to generate input mask(for attention), input type–ids, and input word-ids for each text paragraph.

- Input type-ids: It shows the relationship between 2 sentences in a text. It is more useful for next-sentence prediction.
- Input word-ids: They are the integer representation of words. There is no corresponding mapping of word to integer and the maximum size of input – word-ids are 128 regardless of the size of the text. The first token is always CLS (101) and the last token is always SEP(102)
- Input masks: It tells which token to focus on. If the word-id or token is greater than 0, it is set to 1, else set to 0. Its size is also 128 for a given text.

Once the text is tokenized, embedding is done. bert_multi_cased_L-12_H-768-A-12/4 is used from the tensorflow library is used for this task. This generates vectors that encode the context of tokens. As a result, Encoder output, sequence output, and pooled output are generated.

- Encoder output: Each encoder layer generates a 768-sized vector for each token. There are 12 encoder layers and 128 tokens for each text input. So, one encoder generates 128 vectors for a given text of any size.
- Sequence output: The final encoder layer output is the same as the sequence output. For 128 tokens, there are 128 vectors of size 768.
- Pooled output: It is a vector of size 768 for the entire text. It encodes the context of all 128 token vectors into a single vector. Pooled output is what we will use since the context of the entire text as a whole is required for us instead of token-level context. Also, it is computationally efficient for us.

5.2.4. Model Training

The Nepali text is the input for Bert layers which encode text into vectors. Then the vectors are given as input to the classification layer. The classification layers consist of a few hidden layers and finally a single neuron layer for output prediction.

Dropout regularization has been performed by dropping some percent of neurons randomly in each layer. This has been done to prevent over-fitting.

Once the model architecture is ready, the model is trained using the tensorflow library using binary cross-entropy as a loss function and ADAM optimizer. Different value of

hyper-parameters like learning rate, number of epochs, number of hidden layers, percentage of dropout neurons, and activation function was tried out for optimization purposes.

The optimized hyperparameters for fine-tuning this BERT model are:

- Learning rate = 1e-3
- Epoch = 30
- Batch size = 32

6. RESULTS AND ANALYSIS

In this study, we sought to ascertain how real and fake data related to one another. We evaluated the ratio of these two types of data. We then plotted a graph between different parameters between pre-processed stemmed and non-stemmed data. At first, we plotted the loss versus the number of epochs for both stemmed and non-stemmed data on a graph to show the results. For both stemmed and non-stemmed data, we also developed a confusion matrix to further assess the correctness of the results. Finally, to determine whether the method was more successful in separating authentic data from fraudulent data, we compared the evaluation metrics of the stemmed and non-stemmed data.

6.1. Fake and real data ratio

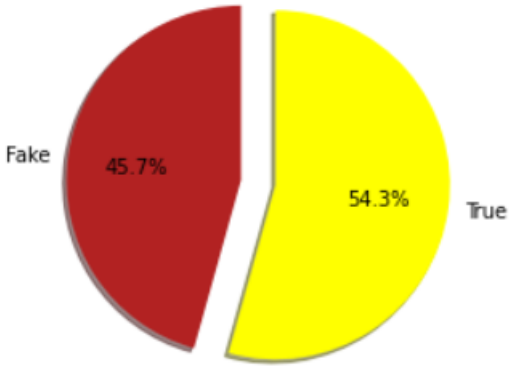


Figure 6-1. Fake and real data percentage

6.2. Loss vs Epoch

a) Stemmed Data

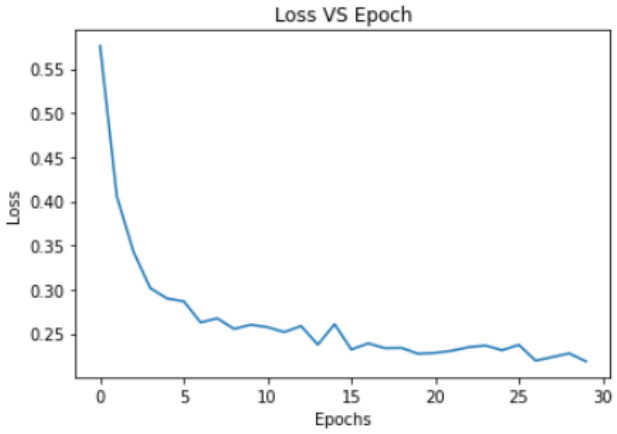


Figure 6-2. Loss vs Epoch graph for Stemmed data

b) Non-stemmed Data

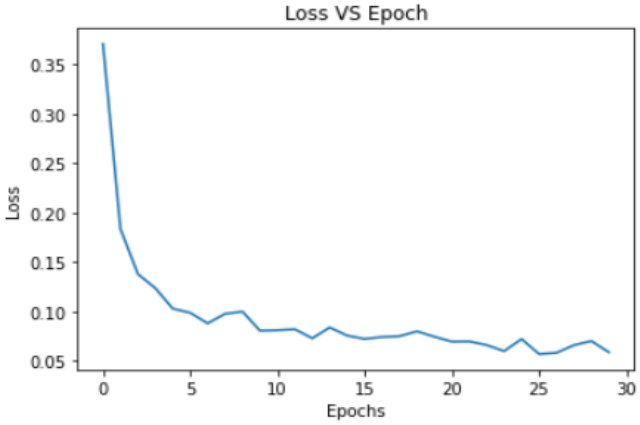


Figure 6-3. Loss vs Epoch for non-stemmed data

6.3. Confusion Matrix

a) Stemmed Data

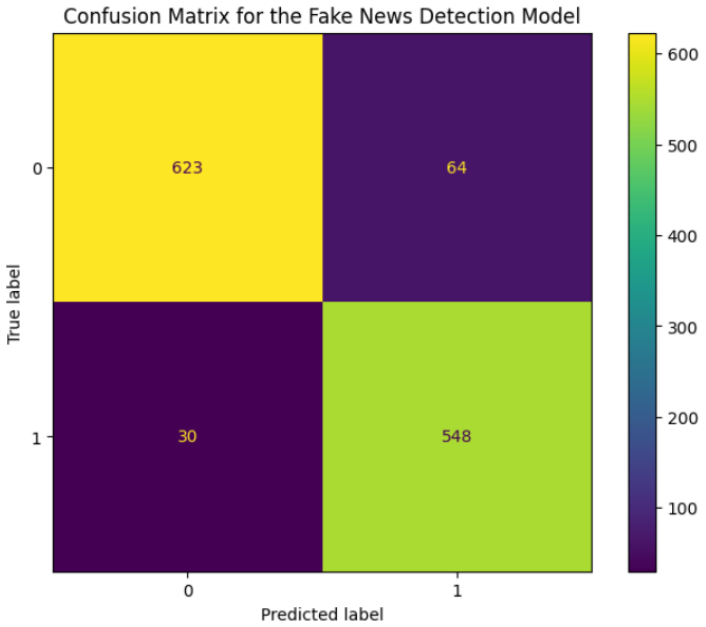


Figure 6-4. Confusion matrix for stemmed data

b) Non-stemmed Data

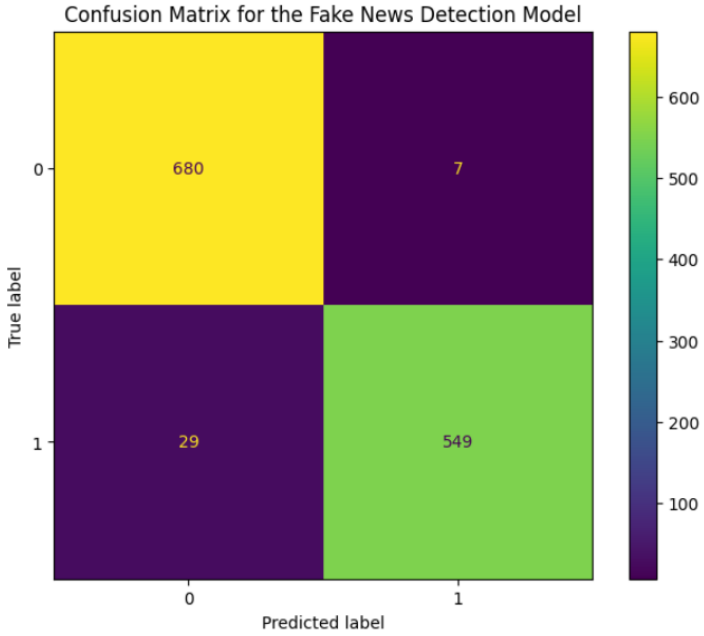


Figure 6-5. Confusion Matrix for non-stemmed data

6.4. Performance Evaluation

a) Stemmed Data

	precision	recall	f1-score	support
0	0.95	0.91	0.93	687
1	0.90	0.95	0.92	578
accuracy			0.93	1265
macro avg	0.92	0.93	0.93	1265
weighted avg	0.93	0.93	0.93	1265

Figure 6-6. Evaluation Matrices of Stemmed Data

b) Non-Stemmed Data

	precision	recall	f1-score	support
0	0.96	0.99	0.97	687
1	0.99	0.95	0.97	578
accuracy			0.97	1265
macro avg	0.97	0.97	0.97	1265
weighted avg	0.97	0.97	0.97	1265

Figure 6-7. Evaluation Matrices of non-stemmed data

7. REMAINING TASKS

- Currently, we have developed a BERT model trained on a fewer dataset. So, we have to yet again train the model on a large dataset.

- The progress of the web implementation project is not complete. So, we have to implement this system on a website.

8. APPENDICES

APPENDIX A: Solve GLUE tasks using BERT on TPU

[13]BERT may be used for numerous natural language processing issues. The GLUE benchmark teaches how to optimize BERT for a variety of tasks:

- Is the phrase grammatically correct? — CoLA (Corpus of Linguistic Acceptability).

- Predicting the emotion of a statement is the goal of SST-2 (Stanford Sentiment Treebank).

- See if two phrases are semantically identical using the MRPC (Microsoft Research Paraphrase Corpus).

- Determine whether a pair of questions are semantically identical using QQP (Quora Question Pairs2).

- Given a premise sentence and a hypothesis sentence, the aim is to determine whether the premise implies the hypothesis (entailment), contradicts the hypothesis (contradiction), or neither of the two (MNLI, or Multi-Genre Natural Language Inference) (neutral).

- Question-response Natural Language Inference (QNLI): It is your responsibility to ascertain whether the context sentence responds to the query.

- RTE stands for Recognizing Textual Entailment, which checks if a phrase implies a particular hypothesis or not.

- The goal of WNLI (Winograd Natural Language Inference) is to determine if the original statement and the sentence with the pronoun substitution are implied by one another.

The various operations performed are:

- Loading models from TensorFlow Hub:

  Here, we can select the BERT model from TensorFlow Hub that we want to fine-tune. There are several different BERT models to pick from.

  - The original BERT authors released BERT-Base, Uncased, and seven additional models with training weights.

  - Small BERTs let you experiment with trade-offs between speed, size, and quality since they have the same basic design but feature fewer and/or smaller Transformer pieces.

  - ALBERT: A Lite BERT in four different sizes that share parameters between layers to decrease the model size (but not calculation time).

  - Eight models that are all BERT-based but offer a selection of pre-training domains to better match the objective job are referred to as BERT Experts.

  - Electra gets pre-trained as a discriminator in a setup that mimics a Generative Adversarial Network (GAN) and has the same architecture as BERT (in three different sizes).

  - The Transformer architecture's core features two upgrades thanks to BERT with Talking-Heads Attention and Gated GELU [base, big].

- After this we have to choose a BERT model to fine-tune

- Preprocess the text

  The preprocessing model is employed on the Classify text using BERT Colab and is directly integrated with the BERT encoder.

This article shows how to use Datasetmap to do preprocessing as part of the input pipeline for training, and how to integrate that preprocessing into the model before it is exported for inference. In this approach, despite the TPU's need for numeric inputs, training and inference may be performed with only raw text inputs.

Regardless of TPU needs, asynchronous preprocessing in an input pipeline can improve performance.

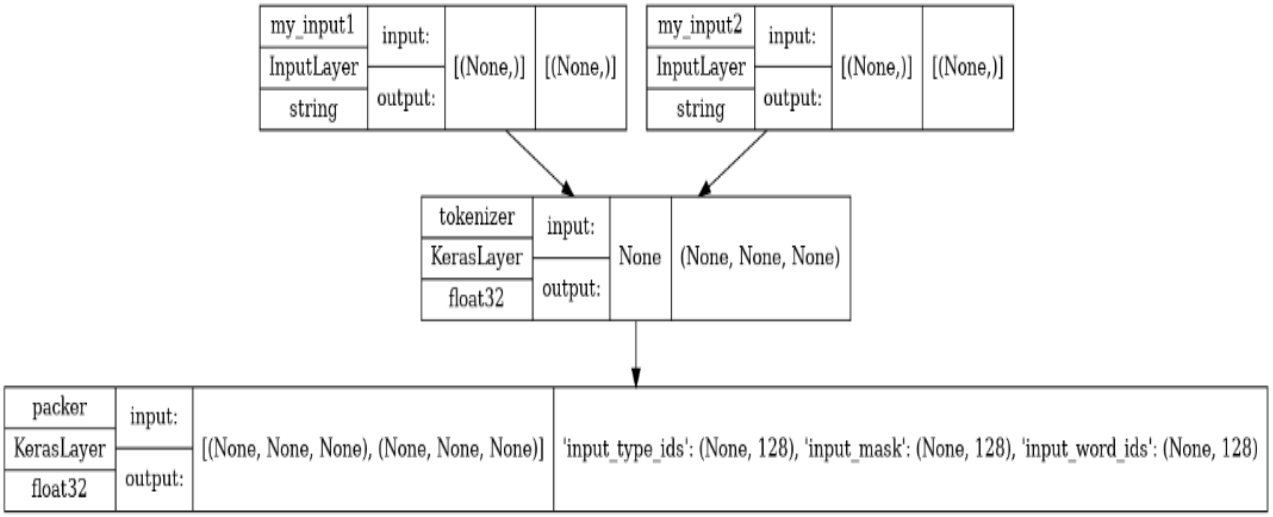


Figure 8-1. Test Process Model of BERT

- Define the Model

  The preprocessed inputs will now be sent through the BERT encoder, a linear classifier will be placed on the top (or another arrangement of layers), and dropout regularization will be used to construct our model for categorizing sentences or phrase pairs.

- Select a task from GLUE.

  We'll apply TensorFlow DataSet from the GLUE benchmark suite.

  Because the separate TPU worker host cannot access the local filesystem of the collab runtime, Colab allows us to download these tiny datasets to the local filesystem. The code below reads them totally into memory.

To enable the TPU worker to read data from larger datasets, we'll need to make our own Google Cloud Storage bucket.

The dataset also chooses the suitable loss function for training as well as the issue type (classification or regression).

- Train the model.

  Finally, we can fully train the model on the selected dataset.

  - Distribution

    Within the framework of the TPU distribution method, we will develop and construct our primary Keras model and distribute training onto it.

  - Optimizer

    The optimizer setup from BERT pre-training (as in Classify text using BERT) is followed by optimizer fine-tuning: The AdamW optimizer is used, preceded with a linear warm-up phase over the first 10% of training steps (num warmup steps), with a linear decline of a notional starting learning rate. The initial learning rate is lower for fine-tuning in line with the BERT paper (best of 5e-5, 3e-5, 2e-5).

- Export for inference:

  The preprocessing component and the newly developed, improved BERT can both be included in the final model that we produce.

  Preprocessing must be integrated into the model at inference time (because there is no longer a separate input queue for training data that does it). Preprocessing is more than simply computation; it also requires the attachment of its resources (the vocabulary table) to the Keras Model that is stored for export. What will be preserved is this vast assemblage.

The model may be saved to Colab, where you can download it later to use it again.

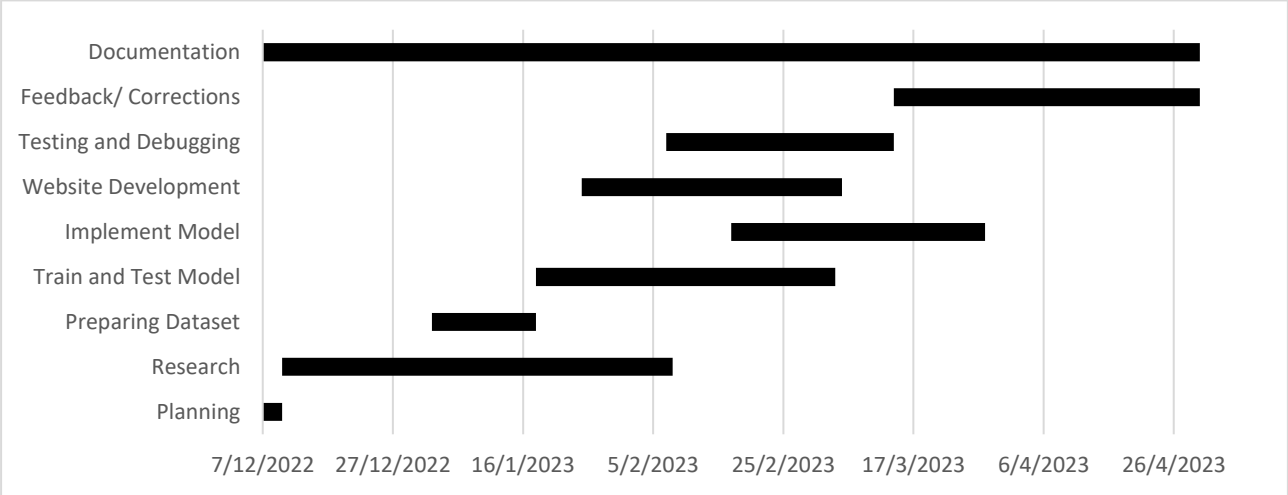
- Testing the Model:

  Testing the output of the exported model is the last step.

  Reload the model and test it using some inputs from the test split of the dataset to make some comparisons.

APPENDIX B: Gantt-Chart

Table 8-1. Gantt Chart



APPENDIX C: Project Cost

Since all of the libraries used for the implementation of this project are open source there is no such cost required for the implementation of this project.

APPENDIX D: Code Snippets

```
for i in list(true_news):
  true_news[i]=true_news[i].str.replace('|', '')
  true_news[i]=true_news[i].str.replace('?', '')
  true_news[i]=true_news[i].str.replace(':', '')
  true_news[i]=true_news[i].str.replace(';', '')
  true_news[i]=true_news[i].str.replace("'", '')
  true_news[i]=true_news[i].str.replace('"', '')
  true_news[i]=true_news[i].str.replace(',', '')
  true_news[i]=true_news[i].str.replace('.', '')
  true_news[i]=true_news[i].str.replace('(', '')
  true_news[i]=true_news[i].str.replace(')', '')
  true_news[i]=true_news[i].str.replace('\n', '')
  true_news[i]=true_news[i].str.replace('&', '')
  true_news[i]=true_news[i].str.replace('—', '')
  true_news[i]=true_news[i].str.replace('‘', '')
  true_news[i]=true_news[i].str.replace('’', '')
  true_news[i]=true_news[i].str.replace('-', '')
  true_news[i]=true_news[i].str.replace('“', '')
  true_news[i]=true_news[i].str.replace('”', '')
  true_news[i]=true_news[i].str.replace('!', '')
```

Figure 8-2. Punctuation removal

```
from keras.optimizers import Adam
optimizer = Adam(learning_rate=1e-3)

METRICS = [
      tf.keras.metrics.BinaryAccuracy(name='accuracy'),
      tf.keras.metrics.Precision(name='precision'),
      tf.keras.metrics.Recall(name='recall')
]

model.compile(optimizer=optimizer, loss='binary_crossentropy', metrics=METRICS)
```

Figure 8-3. Optimizer and loss function

References

[1] C. Elliott, "Forbes," 21 February 2019. [Online]. Available: https://www.forbes.com/sites/christopherelliott/2019/02/21/these-are-the-real-fake-news-sites/?sh=2554c7c23c3e.

[2] D. Adhikari, "Onlinekhabar," 13 December 2022. [Online]. Available: https://english.onlinekhabar.com/misinformation-elections-nepal.html.

[3] A. C. Leo Breiman, *Random Forests*, 2001.

[4] M.-W. C. K. L. K. T. Jacob Devlin, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," Google AI Language, 2019.

[5] K. Stahl, "Fake news detection in social media," 2018.

[6] B. D. P. S. Y. Jiawei Zhang, "FAKEDETECTOR: Effective Fake News Detection," 2019.

[7] S. R. M. K. S. Gowthami. K, "Identification of Fake News through SVM and Random Forest," 2020.

[8] S. K. S. P. R. V. S. P. Tejaswini Yesugade, "Fake News Detection using LSTM," International Research Journal of Engineering and Technology (IRJET), 2021.

[9] S. S. S. M. P. Uma Sharma, "Fake News Detection using Machine Learning Algorithms," International Journal of Engineering Research & Technology (IJERT), 2021.

[10] I. B. D. M. S. A. I. Kadek Sastrawan, "Detection of fake news using deep learning CNN–RNN based methods," 2021.

[11] V. R. H. V. Mahmood Farokhian, "Fake news detection using parallel BERT deep neural networks," 2022.

[12] S. P. P. T. B. T. Shiva Ram Dam, "Detecting Clickbaits on Nepali News using SVM and RF," 2021.

[13] "Tensorflow," [Online]. Available: https://www.tensorflow.org/text/tutorials/bert_glue.