Fake News Detection Using Transfer Learning (English to Telugu)

A Final Project Mid Semester Report

submitted by

P. THARUN SAI (CED19I014)

in partial fulfilment of requirements
for the award of the dual degree of

BACHELOR OF TECHNOLOGY AND MASTER OF TECHNOLOGY

Department of Computer Science and Engineering
INDIAN INSTITUTE OF INFORMATION TECHNOLOGY, DESIGN AND MANUFACTURING, KANCHEEPURAM

MAY 2024

DECLARATION OF ORIGINALITY

I, **P. Tharun Sai**, with Roll No: **CED19I014** hereby declare that the material presented in the Project Report titled **Fake News Detection Using Transfer Learning (English to Telugu)** represents original work carried out by me in the **Department of Computer Science and Engineering** at the Indian Institute of Information Technology, Design and Manufacturing, Kancheepuram.
With my signature, I certify that:

- I have not manipulated any of the data or results.
- I have not committed any plagiarism of intellectual property. I have clearly indicated and referenced the contributions of others.
- I have explicitly acknowledged all collaborative research and discussions.
- I have understood that any false claim will result in severe disciplinary action.
- I have understood that the work may be screened for any form of academic misconduct.

P. Tharun Sai

Place: Chennai

Date: 11.05.2024

CERTIFICATE

This is to certify that the report titled **Fake News Detection Using Transfer Learning (English to Telugu)**, submitted by **P. Tharun Sai (CED19I014)**, to the Indian Institute of Information Technology, Design and Manufacturing Kancheepuram, in partial fulfilment of requirements for the award of the dual degree of **BACHELOR OF TECHNOLOGY AND MASTER OF TECHNOLOGY** is a bonafide record of the work done by him/her under my supervision. The contents of this report, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Dr. Sivaselvan B
Project Internal Guide
Professor
Department of Computer Science and Engineering
IIITDM Kancheepuram, Chennai - 600 127

Place: Chennai

Date:

ACKNOWLEDGEMENTS

I am deeply indebted to Dr. Sivaselvan B, Mercy Faustina J Madam and Santosh Kumar U Sir, and am grateful to them for giving me the autonomy to work and experiment with ideas. I would like to take this opportunity to express my deep gratitude to them, not only for their academic guidance but also for their personal interest in the project and their constant support, coupled with confident and motivating conversations, which proved to be very fruitful and helped to instill confidence and trust in me. It is only thanks to their helpfulness that the present work has reached the stage it has.

Finally, I am grateful to our institution and to our colleagues whose constant encouragement has helped me renew my mind, refocus my attention and energy, and accomplish this work.

P. Tharun Sai

ABSTRACT

Fake news became a major problem in today's digital world. It poses a threat to public discourse, trust, and the integrity of information. In response to this challenge, this project explores the use of transfer learning techniques to detect Fake News. Using pre-trained neural networks, we aim to improve the accuracy and efficiency of Fake News classification.

This project contributes to ongoing efforts to develop effective solutions for Fake News detection and provides insights into the role of transfer learning in addressing this critical societal problem. The news that we are going to handle is in the language Telugu.

KEYWORDS: Public discourse; Integrity; Information; Transfer Learning.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

Introduction

1.1 Fake News

THE VOICE OF REASON SINCE THE BEGINNING
TODAY IS THE DAY
The Daily Times
YOU CAN RELAX
FAKE NEWS
DON'T READ ALL ABOUT IT. STOP. USE REASON.
IT'S ALL SENSATIONALISM. CHECK YOUR SOURCES AND DO YOUR HOMEWORK BEFORE YOU BELIEVE OR REACT.

Figure 1.1: Headlines

In today's world of the Internet, information is a valuable commodity in every field and plays a fundamental role in decision-making, problem-solving, research, and communication. Some areas where information is indispensable are science and research, medicine and healthcare, technology and computers, business and finance, journalism and media, marketing and advertising.

This false or misleading information, presented as if it were true, often intending to deceive, manipulate, or influence people's beliefs or actions, is called fake news.

It takes many forms, including fabricated stories, manipulated images and videos, and disinformation spread through social media, websites, or traditional media channels.

From the fig2 we can see the statistics of the source of news in India. The major source is online social media and with the increasing population that is being attracted to the internet, there is a huge problem here. If fake news is spread like wildfire this

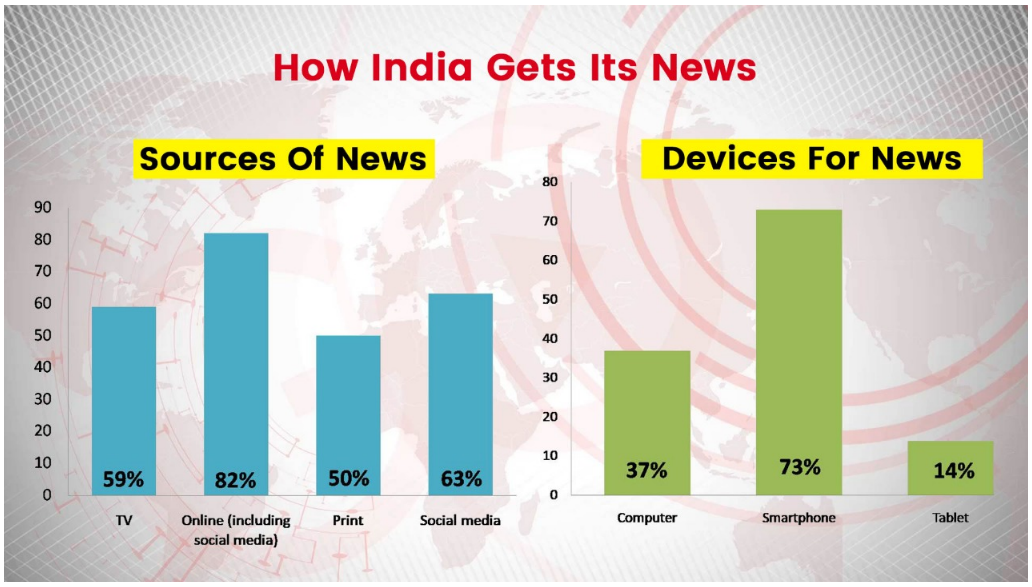


Figure 1.2: Sources of news

may cause unrest in public opinion which may sometimes lead to violence resulting in riots, and division of the public into groups that are threatening each other.

1.2 Importance of Fake News Detection

The importance of detecting and combating fake news cannot be overstated. Fake news can have far-reaching consequences, including:

1. Misinformation: People can make wrong decisions based on false information, whether intentionally or unintentionally.

2. Erosion of trust: The widespread dissemination of fake news erodes trust in reliable news sources and institutions and undermines the public's ability to distinguish between credible and untrustworthy information.

3. Polarization: it reinforces already existing prejudices in society and leads to a division of society or institutions into different groups that separate people from each other.

4. Manipulation: it can be used as a weapon by malicious people to influence public opinion. This can be seen in local politics, elections, and geopolitical events.

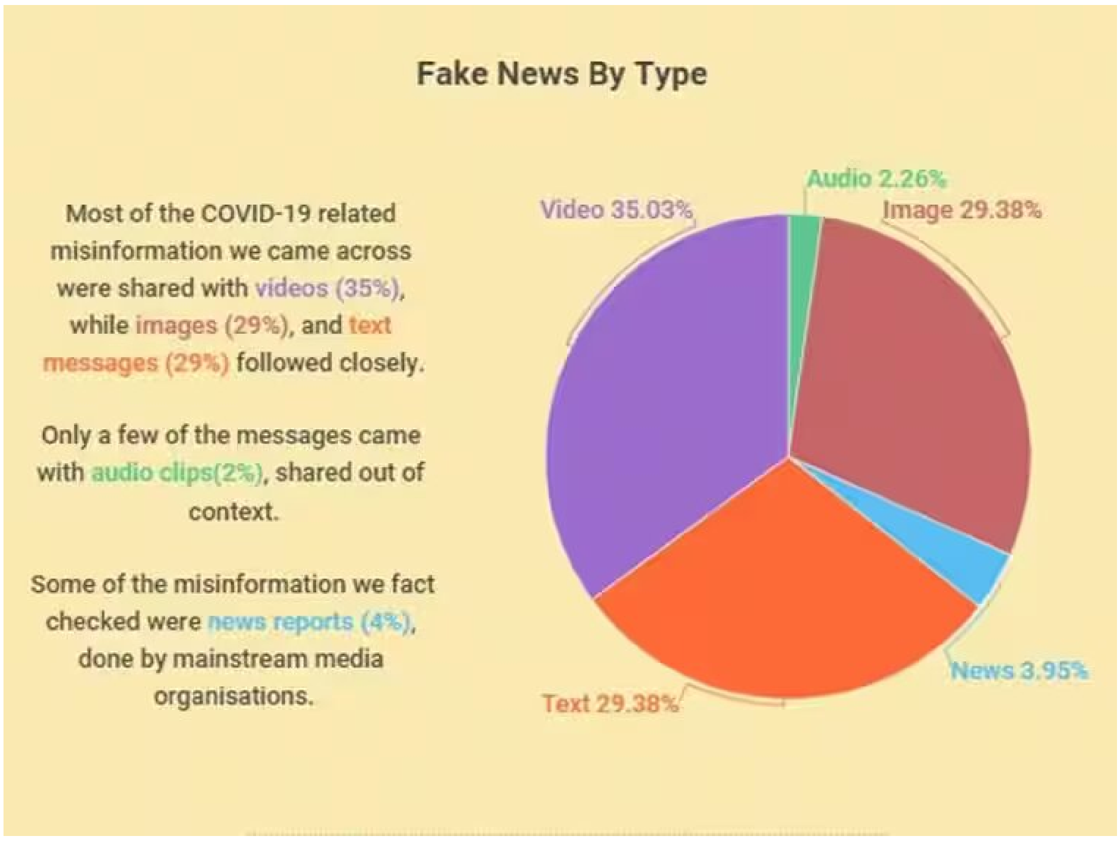


Figure 1.3: Fake news about COVID-19

1.3 Transfer Learning

Transfer learning is an ML technique in which a learning model is trained for one task and is adapted or fine-tuned for a related but different task. Rather than building a model from scratch, transfer learning uses the knowledge gained in solving one problem to improve performance on another.[2] This approach is particularly useful when there is a limited amount of labeled data available for the target task, as it allows you to benefit from existing knowledge and data.[3]

This is what we will use in this project to detect fake news.

CHAPTER 2

Objective

2.1 Objective

Addressing these cases of fake news requires a multifaceted approach, including media literacy education, fact-checking initiatives, improved algorithms for social media platforms, and responsible reporting by mainstream media outlets. It is essential for individuals to critically evaluate information sources and be cautious about sharing unverified or sensationalized news.

The project's main objective is

- To create an interface capable of distinguishing whether a given news is fake or not using Transfer Learning.
- The application should run with minimal computing power and generate insights on-premise.
- This project focuses on Language Telugu.

CHAPTER 3

Methodology

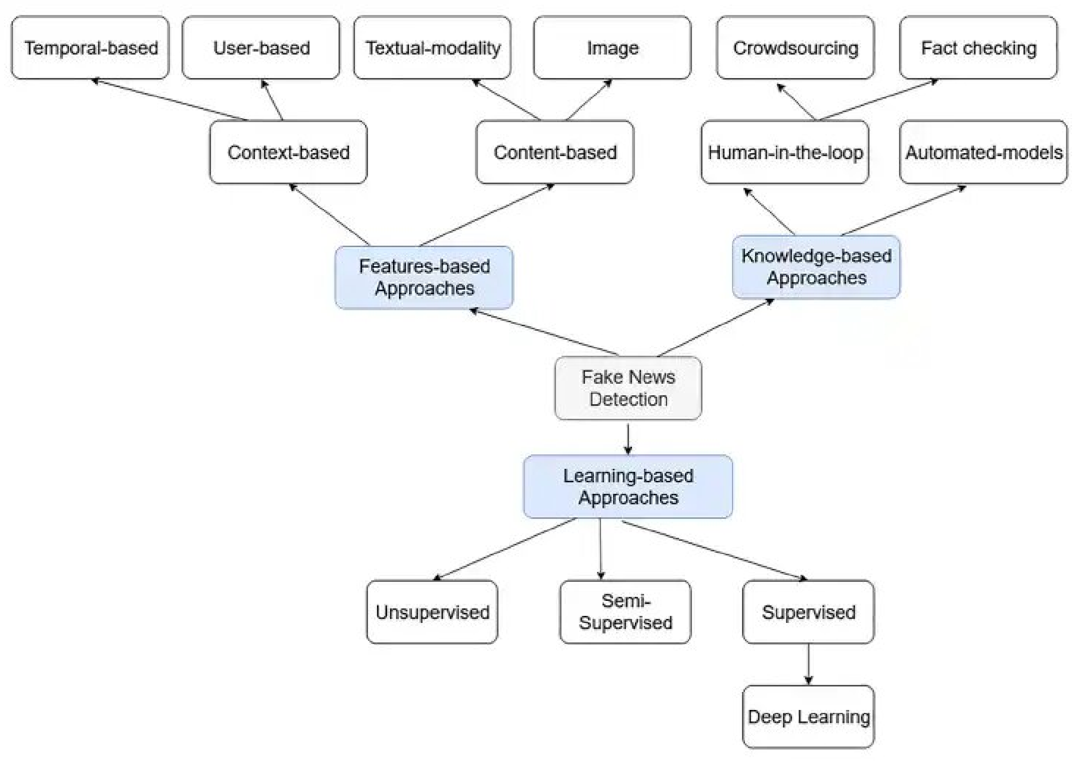


Fig. 2 Approaches for fake news detection

Figure 3.1: Approaches for fake news detection [1]

3.1 Transfer Learning

Fake news detection using transfer learning is a powerful approach. Since our project is in language Telugu learning models we chose to use GPT-3 and Google Bard which can already understand Telugu and are already trained in that language.

[1]

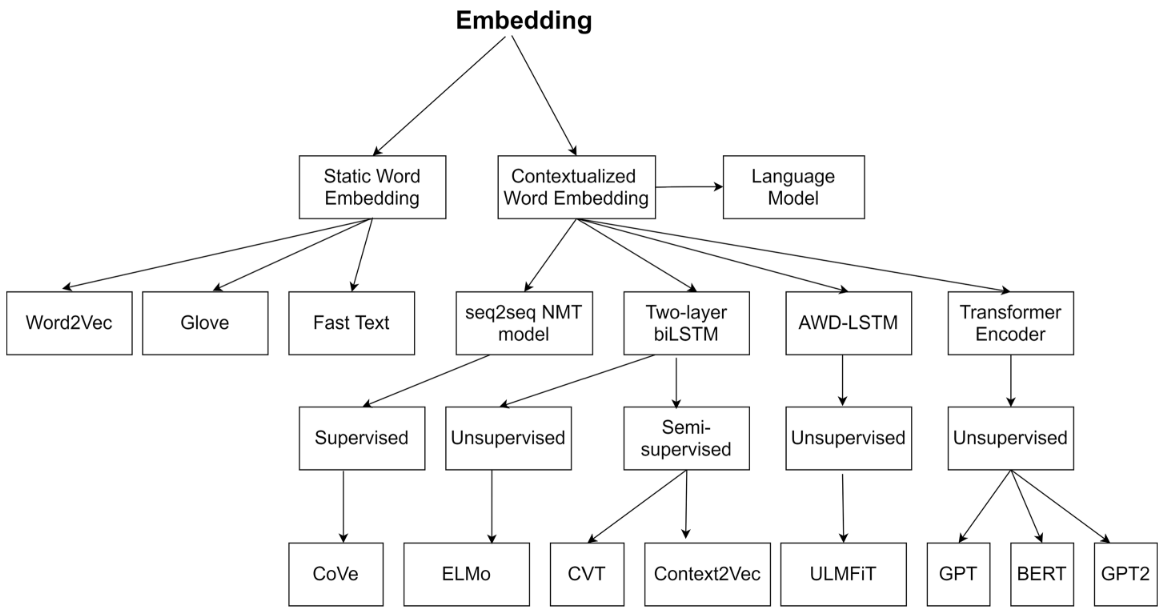


Figure 3.2: An Overview of existing word-embedding models

3.2 Data Collection

- Databases that are required for training the models are being collected by gathering data from newspapers, articles, and a few online news.
- Divide your data set into training, validation, and test sets.
- The quality and size of the dataset that we use determine the success of our model. We also need to think of how to overcome the challenge of unbalanced datasets.

3.3 Choose a Pre-trained Language Model:

- Choose a pre-trained language model designed for NLP tasks. State-of-the-art models such as BERT (Bidirectional Encoder Representations from Transformers), GPT-3, or RoBERTA are excellent choices. These models have been trained with large text corpora and have shown impressive performance on various NLP tasks.

3.4 Training the Models

- Perform fine-tuning of the selected pre-trained language model on your labeled message record. During fine-tuning, adapt the model to your specific fake news detection task.
- Define the task as a binary classification (real or fake news) and train the model to predict the correct label.

- Evaluate the fine-tuned model using standard classification metrics such as accuracy, precision, recall, F1 score, and confusion matrix on the test set.
- Experiment with hyperparameters such as learning rate, batch size, and number of training epochs to optimize model performance.
- Depending on the output probabilities of the model, you can set a threshold to classify news articles as fake or real. For example, if the predicted probability of being fake is above 0.5, classify it as Fake News.

3.5 Model Evaluation:

- Divide your data set into training, validation, and test sets.
- Evaluate the fine-tuned model using standard classification metrics such as accuracy, precision, recall, F1 score, and confusion matrix on the test set.

3.6 Hyperparameter Tuning:

- Experiment with hyperparameters such as learning rate, batch size, and number of training epochs to optimize model performance.

3.7 Deployment

- Learn how to develop web applications and create.
- Deploy the trained model as part of a web application or API where users can enter news articles for classification.
- Continuously monitor the performance of the model and update it as needed to adapt to evolving fake news tactics.

3.8 Continuous Monitoring:

- Continuously monitor the performance of the model and update it as needed to adapt to evolving fake news tactics.

3.9 Web App Development

- **Define Your Purpose and Goals:** Understand the purpose and target audience of the application.
- **Choose Technology Stack:** Select a suitable technology stack, programming language, and web framework.
- **Design the User Interface (UI):** Create a simple and attractive user interface such that it attracts people who mistakenly visit.
- **Backend Development:** Write backend code that handles logic, requests, user authentication, and interaction with the database.
- **Database Integration:** Select a DBMS accordingly and connect it to the database and then all to the backend.
- **User Authentication:** Give the user a username and password to keep his data separate and safe.
- **Testing:** Test run the application to check for any errors and take care of it.
- **Deployment:** Release the web app online.
- **Maintainence:** Check for bugs and add updates regularly.

3.10 Databases

For the above process to run smoothly we need a proper dataset for our use. Getting a Telugu dataset on our specific topic might be hard. So we have to know how to create a dataset for use.

3.11 Data Structure for Database

- Determine the information that we wanted to extract from newspapers. This includes headlines, context, author name, publication company, dates, etc.
- Organize the database like a spreadsheet format initially for the ease of data entry.

3.12 Data Entry

- Manually go through all newspapers and extract relevant information according to our defined structure.

- Using a spreadsheet, each row represents a news article with columns for different attributes like headline, context, author, date, etc.

3.13 Digitization

- As the training dataset increases in size we will be scanning the newspapers and then use text extractors to build databases. As Training a learning model requires very large databases. For more information use citation.[4]
- In this the articles are scanned giving us images. From these images, we are going to use tools that have OCR capabilities(Optical Character Recognition). For more information use citation.[5]

3.14 Data Cleaning

- Clean the extracted data from unnecessary characters, noise, and inconsistencies according to the format of the database.

CHAPTER 4

Work Done

4.1 Learning Model

Many factors need to be considered if we want to have an optimal learning model for this task. Factors like the nature of the data, complexity, computational resources available, and our requirements in performance.[6][7]

After finding suitable learning models we then eliminate learning models by experimenting and evaluating them multiple times based on performance metrics like accuracy, precision, recall, and interpretability of the data.[8]

4.2 Bert Learning Model

We decided on using BERT(Bidirectional Encoder Representations from Transformers) as it provides several advantages.[9]

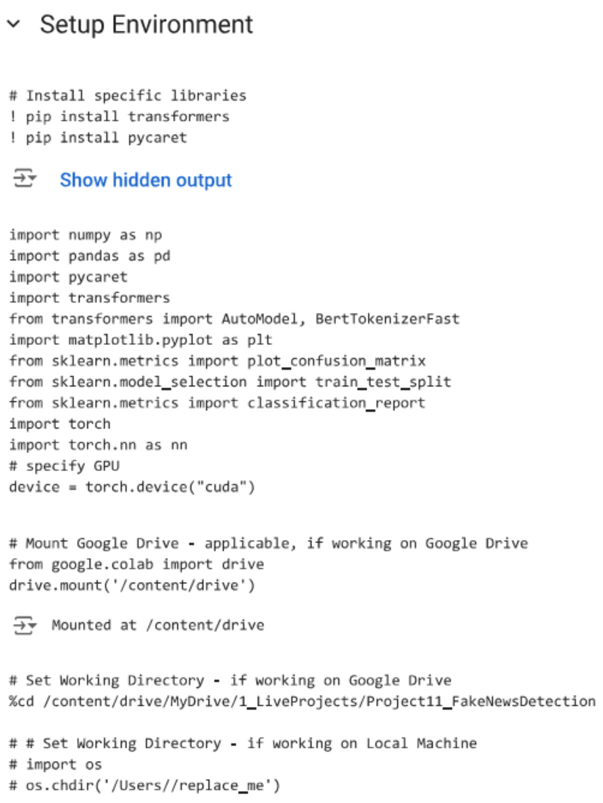
- BERT has already been trained in multiple languages including Telugu.
- As it has been trained on large contextual data this helps it in recognising patterns and relationships between words in a sentence.
- Due to its massive fine-tunning capabilities.
- BERT achieves state-of-the-art results on many linguistic tasks, including text classification. Its ability to capture complex explanations makes it a popular choice for many NLP applications, including fake news detection.
- And most important of all is that it is open source and free to use for wide range of applications.

4.3 Coding for Fake News Detection

We need to have access to the GPU environment for running the codes. This is available in the Google Colab for free use.[10]

- Firstly we need to set up the environment. Start by installing transformers which help us to import learning models.
- Then we import essential libraries like numpy, pandas, sklearn, matplotlib, and others.
- Then we mount our drive where all our project-related data is stored. Then set up a directory for use.
- Now loading the datasets and making changes if needed as per the requirements like labeling the data True or False. And converting the string data to numerical for the machine to understand.
- Checking the data whether it is balanced or not between True and False by plotting a piechart.
- Then we split our data into ratios of Training data, Validation, and Test data like [70:15:15].
- Now it's time for Fine-tuning and we load the BERT model based on our requirements. Wether to use BERTbase or BERTlarge. But for our requirement, BERTbase is enough.
- Then it is time for training the data where we will only be using the Article Title as input data. Using all the context requires much more system resources than what is provided to us by Google Colab and it takes a long time.
- Even though we are limiting ourselves only to the Title we will still be getting a 90 percent accuracy.
- Now we Tokenize our Training and Validation sets. Then we convert integer sequences to Tensors. Then we create dataloaders for training and validation sets.
- Now we freeze our pre-trained model weights as to not lose already trained data.
- We are going to use PyTorch for defining, training, and evaluating our Neural Network models.
- Then we define our Hyperparameters and use AdamW as our optimizer. Then we define our loss function and keep the epoch value 2 as each epoch might take about 20 minutes.
- Then define training and evaluation functions and run them.
- Finally we can no fine-tune our BERT mode for fake news detection.
- Then after fine-tuning we can check our model performance. Then we can check how our model Prediction works..

4.4 Sample codes



Setup Environment

```
# Install specific libraries
! pip install transformers
! pip install pycaret
```

Show hidden output

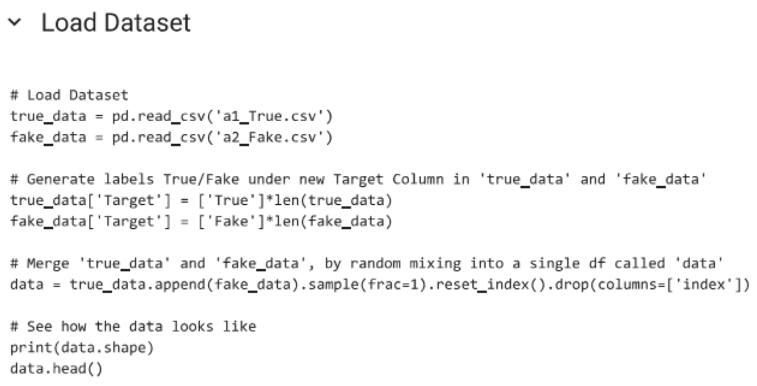
```
import numpy as np
import pandas as pd
import pycaret
import transformers
from transformers import AutoModel, BertTokenizerFast
import matplotlib.pyplot as plt
from sklearn.metrics import plot_confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
import torch
import torch.nn as nn
# specify GPU
device = torch.device("cuda")
```

```
# Mount Google Drive - applicable, if working on Google Drive
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
# Set Working Directory - if working on Google Drive
%cd /content/drive/MyDrive/1_LiveProjects/Project11_FakeNewsDetection

# # Set Working Directory - if working on Local Machine
# import os
# os.chdir('/Users//replace_me')
```

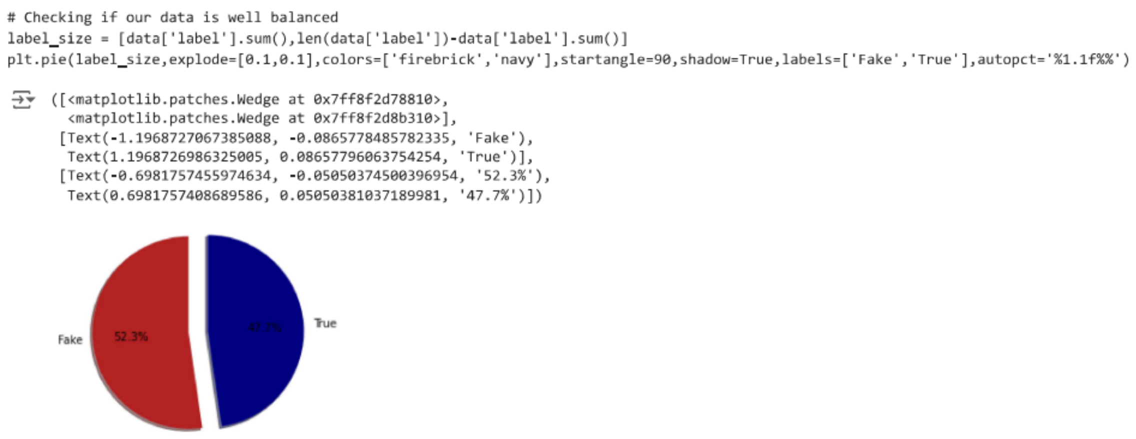
Figure 4.1: Set Environment



Load Dataset

```
# Load Dataset
true_data = pd.read_csv('a1_True.csv')
fake_data = pd.read_csv('a2_Fake.csv')

# Generate labels True/Fake under new Target Column in 'true_data' and 'fake_data'
true_data['Target'] = ['True']*len(true_data)
fake_data['Target'] = ['Fake']*len(fake_data)

# Merge 'true_data' and 'fake_data', by random mixing into a single df called 'data'
data = true_data.append(fake_data).sample(frac=1).reset_index().drop(columns=['index'])

# See how the data looks like
print(data.shape)
data.head()
```

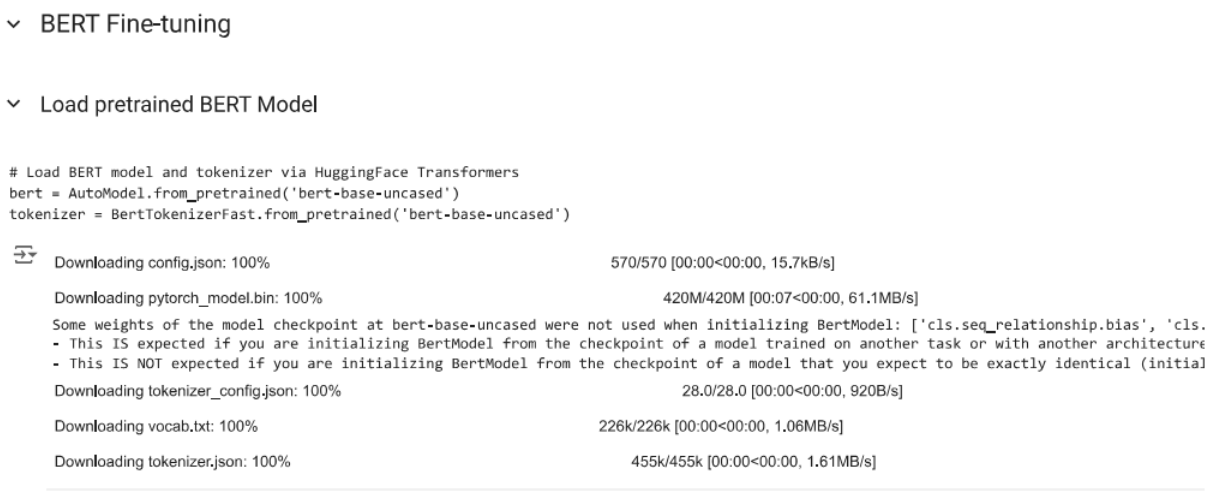
Figure 4.2: Load dataset



```
# Checking if our data is well balanced
label_size = [data['label'].sum(),len(data['label'])-data['label'].sum()]
plt.pie(label_size,explode=[0.1,0.1],colors=['firebrick','navy'],startangle=90,shadow=True,labels=['Fake','True'],autopct='%1.1f%%')
```

```
([<matplotlib.patches.Wedge at 0x7ff8f2d78810>,
  <matplotlib.patches.Wedge at 0x7ff8f2d8b310>],
 [Text(-1.1968727067385088, -0.0865778485782335, 'Fake'),
  Text(1.1968726986325005, 0.08657796063754254, 'True')],
 [Text(-0.6981757455974634, -0.05050374500396954, '52.3%'),
  Text(0.6981757408689586, 0.05050381037189981, '47.7%')])
```

Figure 4.3: Data Balanced



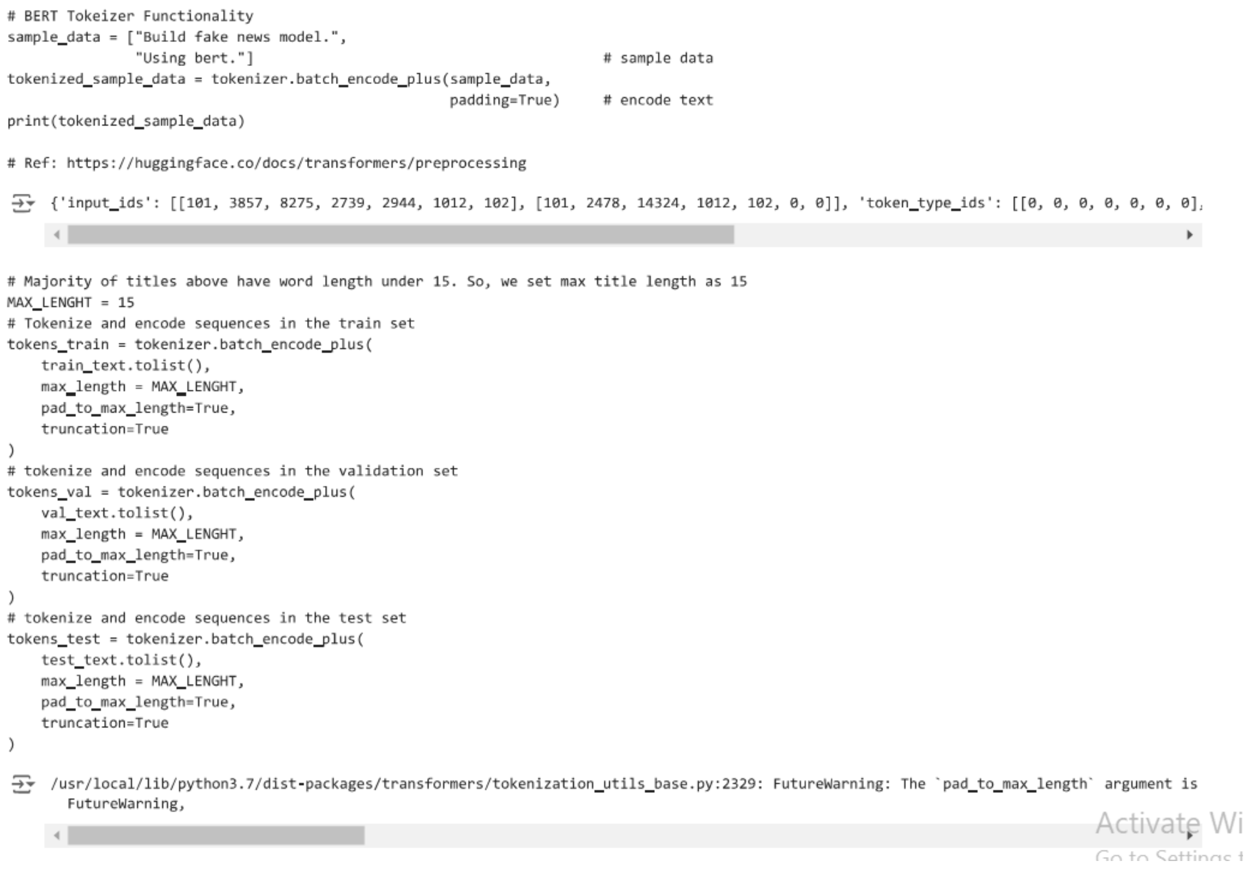
BERT Fine-tuning

Load pretrained BERT Model

```
# Load BERT model and tokenizer via HuggingFace Transformers
bert = AutoModel.from_pretrained('bert-base-uncased')
tokenizer = BertTokenizerFast.from_pretrained('bert-base-uncased')
```

```
Downloading config.json: 100%                                   570/570 [00:00<00:00, 15.7kB/s]
Downloading pytorch_model.bin: 100%                             420M/420M [00:07<00:00, 61.1MB/s]
Some weights of the model checkpoint at bert-base-uncased were not used when initializing BertModel: ['cls.seq_relationship.bias', 'cls.
- This IS expected if you are initializing BertModel from the checkpoint of a model trained on another task or with another architecture
- This IS NOT expected if you are initializing BertModel from the checkpoint of a model that you expect to be exactly identical (initial
Downloading tokenizer_config.json: 100%                         28.0/28.0 [00:00<00:00, 920B/s]
Downloading vocab.txt: 100%                                     226k/226k [00:00<00:00, 1.06MB/s]
Downloading tokenizer.json: 100%                                455k/455k [00:00<00:00, 1.61MB/s]
```

Figure 4.4: Load BERT model



```
# BERT Tokeizer Functionality
sample_data = ["Build fake news model.",
               "Using bert."]                                  # sample data
tokenized_sample_data = tokenizer.batch_encode_plus(sample_data,
                                                    padding=True)     # encode text
print(tokenized_sample_data)

# Ref: https://huggingface.co/docs/transformers/preprocessing
```

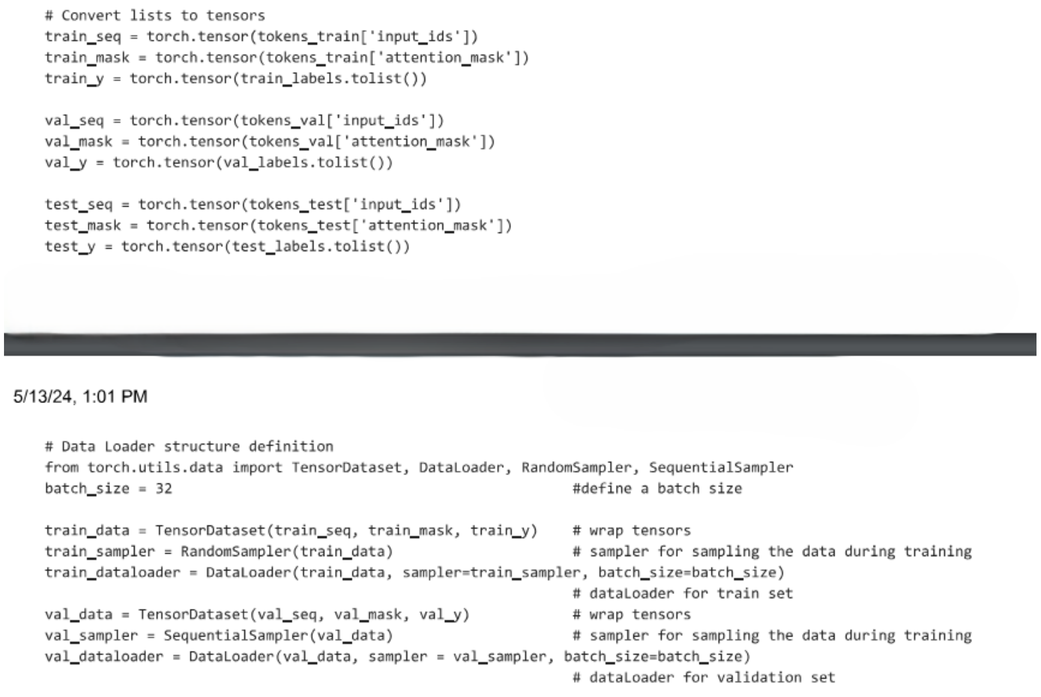
```
{'input_ids': [[101, 3857, 8275, 2739, 2944, 1012, 102], [101, 2478, 14324, 1012, 102, 0, 0]], 'token_type_ids': [[0, 0, 0, 0, 0, 0, 0],
```

```
# Majority of titles above have word length under 15. So, we set max title length as 15
MAX_LENGHT = 15
# Tokenize and encode sequences in the train set
tokens_train = tokenizer.batch_encode_plus(
    train_text.tolist(),
    max_length = MAX_LENGHT,
    pad_to_max_length=True,
    truncation=True
)
# tokenize and encode sequences in the validation set
tokens_val = tokenizer.batch_encode_plus(
    val_text.tolist(),
    max_length = MAX_LENGHT,
    pad_to_max_length=True,
    truncation=True
)
# tokenize and encode sequences in the test set
tokens_test = tokenizer.batch_encode_plus(
    test_text.tolist(),
    max_length = MAX_LENGHT,
    pad_to_max_length=True,
    truncation=True
)
```

```
/usr/local/lib/python3.7/dist-packages/transformers/tokenization_utils_base.py:2329: FutureWarning: The `pad_to_max_length` argument is
  FutureWarning,
```

Activate Wi
Go to Settings t

Figure 4.5: Tokenization

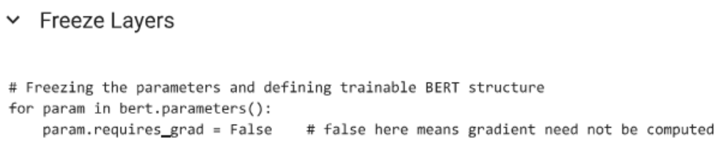


```
# Convert lists to tensors
train_seq = torch.tensor(tokens_train['input_ids'])
train_mask = torch.tensor(tokens_train['attention_mask'])
train_y = torch.tensor(train_labels.tolist())

val_seq = torch.tensor(tokens_val['input_ids'])
val_mask = torch.tensor(tokens_val['attention_mask'])
val_y = torch.tensor(val_labels.tolist())

test_seq = torch.tensor(tokens_test['input_ids'])
test_mask = torch.tensor(tokens_test['attention_mask'])
test_y = torch.tensor(test_labels.tolist())
```

5/13/24, 1:01 PM

```
# Data Loader structure definition
from torch.utils.data import TensorDataset, DataLoader, RandomSampler, SequentialSampler
batch_size = 32                                             #define a batch size

train_data = TensorDataset(train_seq, train_mask, train_y)    # wrap tensors
train_sampler = RandomSampler(train_data)                   # sampler for sampling the data during training
train_dataloader = DataLoader(train_data, sampler=train_sampler, batch_size=batch_size)
                                                            # dataLoader for train set
val_data = TensorDataset(val_seq, val_mask, val_y)          # wrap tensors
val_sampler = SequentialSampler(val_data)                   # sampler for sampling the data during training
val_dataloader = DataLoader(val_data, sampler = val_sampler, batch_size=batch_size)
                                                            # dataLoader for validation set
```

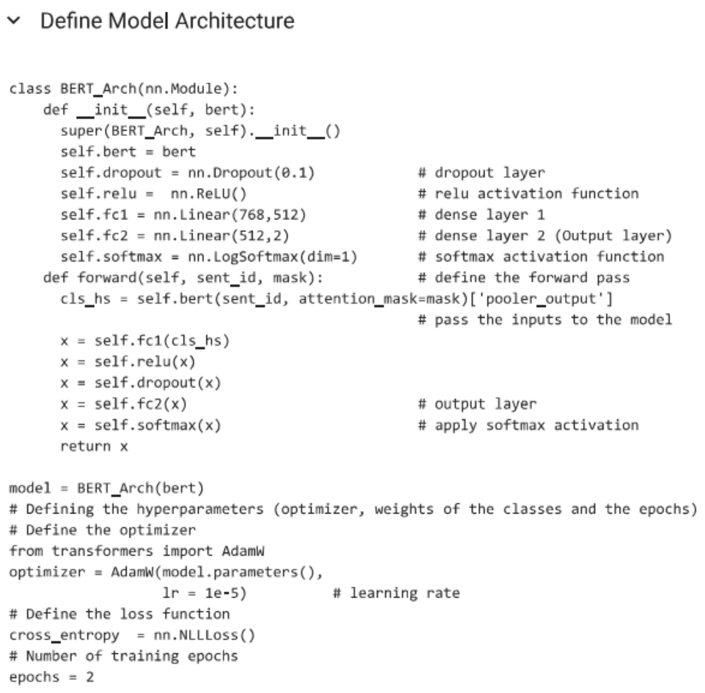
Figure 4.6: Tensors



Freeze Layers

```
# Freezing the parameters and defining trainable BERT structure
for param in bert.parameters():
    param.requires_grad = False    # false here means gradient need not be computed
```

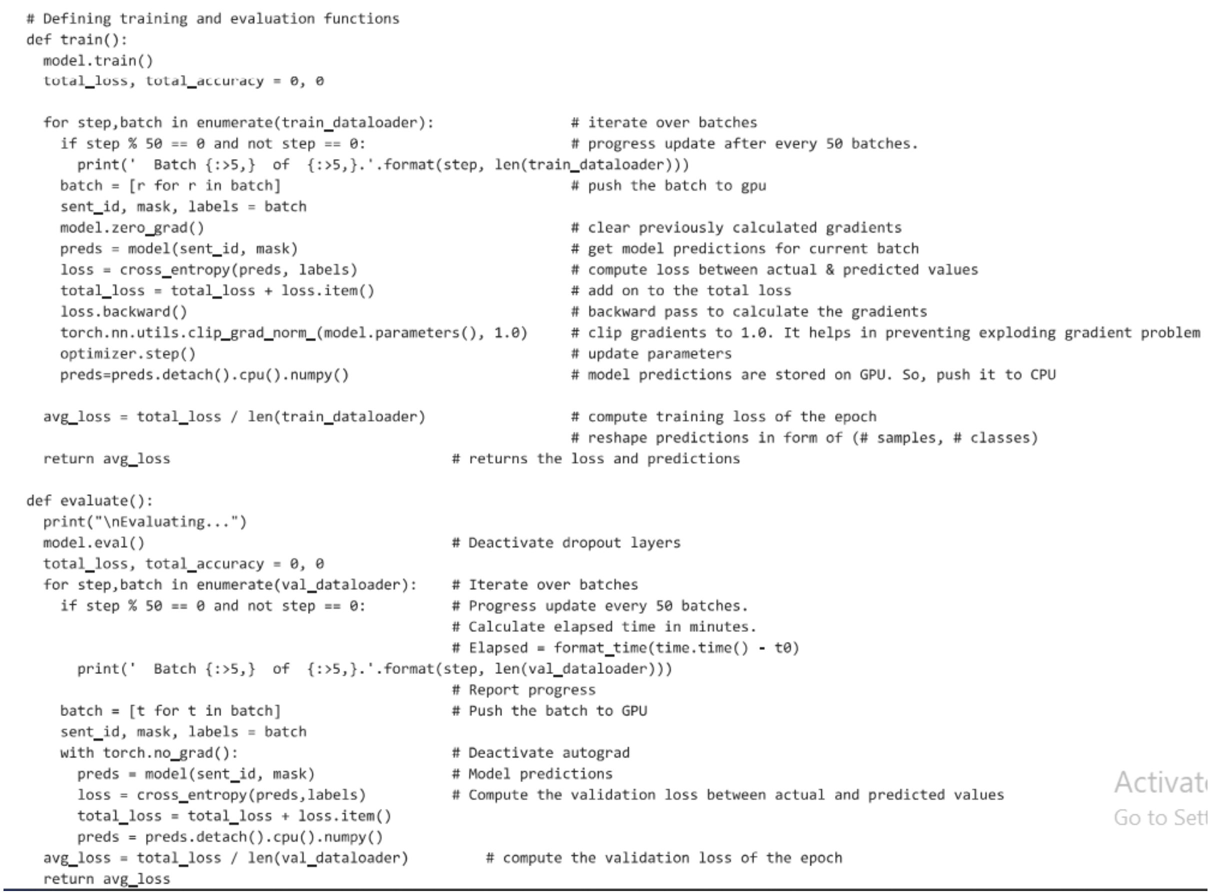
Figure 4.7: Freeze Layers



Define Model Architecture

```
class BERT_Arch(nn.Module):
    def __init__(self, bert):
      super(BERT_Arch, self).__init__()
      self.bert = bert
      self.dropout = nn.Dropout(0.1)            # dropout layer
      self.relu =  nn.ReLU()                    # relu activation function
      self.fc1 = nn.Linear(768,512)             # dense layer 1
      self.fc2 = nn.Linear(512,2)               # dense layer 2 (Output layer)
      self.softmax = nn.LogSoftmax(dim=1)       # softmax activation function
    def forward(self, sent_id, mask):           # define the forward pass
      cls_hs = self.bert(sent_id, attention_mask=mask)['pooler_output']
                                                # pass the inputs to the model
      x = self.fc1(cls_hs)
      x = self.relu(x)
      x = self.dropout(x)
      x = self.fc2(x)                           # output layer
      x = self.softmax(x)                       # apply softmax activation
      return x

model = BERT_Arch(bert)
# Defining the hyperparameters (optimizer, weights of the classes and the epochs)
# Define the optimizer
from transformers import AdamW
optimizer = AdamW(model.parameters(),
                  lr = 1e-5)          # learning rate
# Define the loss function
cross_entropy  = nn.NLLLoss()
# Number of training epochs
epochs = 2
```

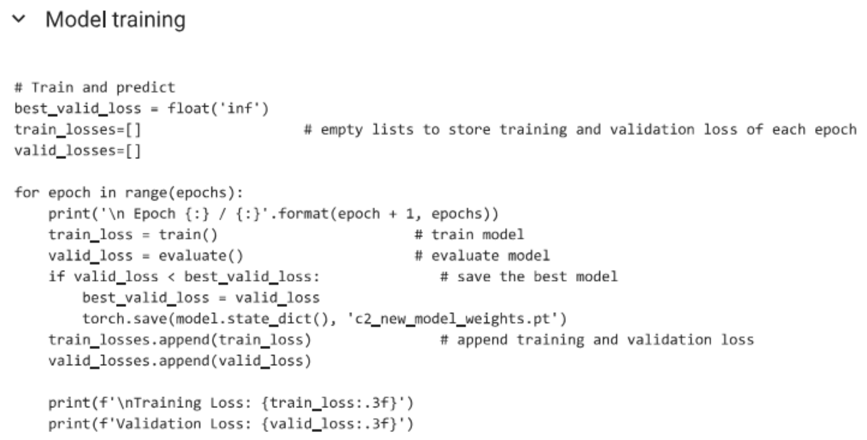
Figure 4.8: Model Architecture



```
# Defining training and evaluation functions
def train():
  model.train()
  total_loss, total_accuracy = 0, 0

  for step,batch in enumerate(train_dataloader):                # iterate over batches
    if step % 50 == 0 and not step == 0:                        # progress update after every 50 batches.
      print('  Batch {:>5,}  of  {:>5,}.'.format(step, len(train_dataloader)))
    batch = [r for r in batch]                                  # push the batch to gpu
    sent_id, mask, labels = batch
    model.zero_grad()                                           # clear previously calculated gradients
    preds = model(sent_id, mask)                                # get model predictions for current batch
    loss = cross_entropy(preds, labels)                         # compute loss between actual & predicted values
    total_loss = total_loss + loss.item()                       # add on to the total loss
    loss.backward()                                             # backward pass to calculate the gradients
    torch.nn.utils.clip_grad_norm_(model.parameters(), 1.0)     # clip gradients to 1.0. It helps in preventing exploding gradient problem
    optimizer.step()                                            # update parameters
    preds=preds.detach().cpu().numpy()                          # model predictions are stored on GPU. So, push it to CPU

  avg_loss = total_loss / len(train_dataloader)                 # compute training loss of the epoch
                                                                # reshape predictions in form of (# samples, # classes)
  return avg_loss                                 # returns the loss and predictions

def evaluate():
  print("\nEvaluating...")
  model.eval()                                    # Deactivate dropout layers
  total_loss, total_accuracy = 0, 0
  for step,batch in enumerate(val_dataloader):    # Iterate over batches
    if step % 50 == 0 and not step == 0:          # Progress update every 50 batches.
                                                  # Calculate elapsed time in minutes.
                                                  # Elapsed = format_time(time.time() - t0)
      print('  Batch {:>5,}  of  {:>5,}.'.format(step, len(val_dataloader)))
                                                  # Report progress
    batch = [t for t in batch]                    # Push the batch to GPU
    sent_id, mask, labels = batch
    with torch.no_grad():                         # Deactivate autograd
      preds = model(sent_id, mask)                # Model predictions
      loss = cross_entropy(preds,labels)          # Compute the validation loss between actual and predicted values
      total_loss = total_loss + loss.item()
      preds = preds.detach().cpu().numpy()
  avg_loss = total_loss / len(val_dataloader)         # compute the validation loss of the epoch
  return avg_loss
```

Activat
Go to Sett

Figure 4.9: Train& Eval Functtions



Model training

```
# Train and predict
best_valid_loss = float('inf')
train_losses=[]                   # empty lists to store training and validation loss of each epoch
valid_losses=[]

for epoch in range(epochs):
    print('\n Epoch {:} / {:}'.format(epoch + 1, epochs))
    train_loss = train()                       # train model
    valid_loss = evaluate()                    # evaluate model
    if valid_loss < best_valid_loss:              # save the best model
        best_valid_loss = valid_loss
        torch.save(model.state_dict(), 'c2_new_model_weights.pt')
    train_losses.append(train_loss)               # append training and validation loss
    valid_losses.append(valid_loss)

    print(f'\nTraining Loss: {train_loss:.3f}')
    print(f'Validation Loss: {valid_loss:.3f}')
```

Figure 4.10: Model Training

CHAPTER 5

CONCLUSION AND FUTURE SCOPE

5.1 Conclusion

We wanted to build an application that can detect whether a given news is true or false with better accuracy and efficiency with minimal requirements. I hope we can continue this project and contribute to this important part of the future digital world.

5.2 Future Scope

The future scope for the project is to launch the application with good efficiency and correctness. If the results were quite good then I would like to explore other languages so that it can be used across the nation and even the world.

REFERENCES

[1] R. K. Kaliyar, A. Goswami, and P. Narang, “Fakebert: Fake news detection in social media with a bert-based deep learning approach,” *Multimedia tools and applications*, vol. 80, no. 8, pp. 11 765–11 788, 2021.

[2] A. Hosna, E. Merry, J. Gyalmo, Z. Alom, Z. Aung, and M. A. Azim, “Transfer learning: a friendly introduction,” *Journal of Big Data*, vol. 9, no. 1, p. 102, 2022.

[3] L. Torrey and J. Shavlik, “Transfer learning,” in *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*. IGI global, 2010, pp. 242–264.

[4] C. Finegan-Dollak, J. K. Kummerfeld, L. Zhang, K. Ramanathan, S. Sadasivam, R. Zhang, and D. Radev, “Improving text-to-sql evaluation methodology,” *arXiv preprint arXiv:1806.09029*, 2018.

[5] C. Patel, A. Patel, and D. Patel, “Optical character recognition by open source ocr tool tesseract: A case study,” *International Journal of Computer Applications*, vol. 55, no. 10, pp. 50–56, 2012.

[6] L. Floridi and M. Chiriatti, “Gpt-3: Its nature, scope, limits, and consequences,” *Minds and Machines*, vol. 30, pp. 681–694, 2020.

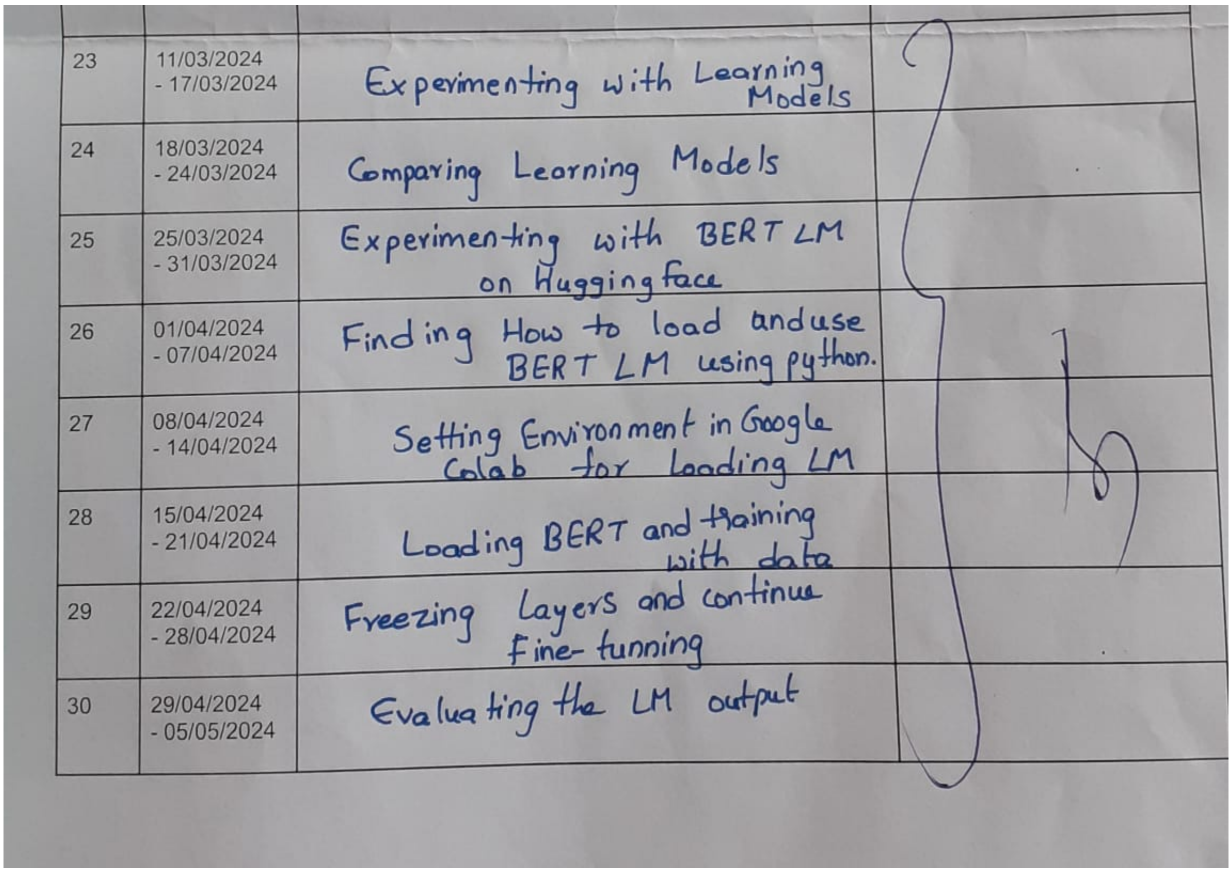
[7] R. Dale, “Gpt-3: What’s it good for?” *Natural Language Engineering*, vol. 27, no. 1, pp. 113–118, 2021.

[8] S. González-Carvajal and E. C. Garrido-Merchán, “Comparing bert against traditional machine learning text classification,” *arXiv preprint arXiv:2005.13012*, 2020.

[9] A. R. Abas, I. Elhenawy, M. Zidan, and M. Othman, “Bert-cnn: A deep learning model for detecting emotions from text.” *Computers, Materials & Continua*, vol. 71, no. 2, 2022.

[10] R. Vuduc, A. Chandramowlishwaran, J. Choi, M. Guney, and A. Shringarpure, “On the limits of gpu acceleration,” in *Proceedings of the 2nd USENIX conference on Hot topics in parallelism*, vol. 13, no. 0, 2010.

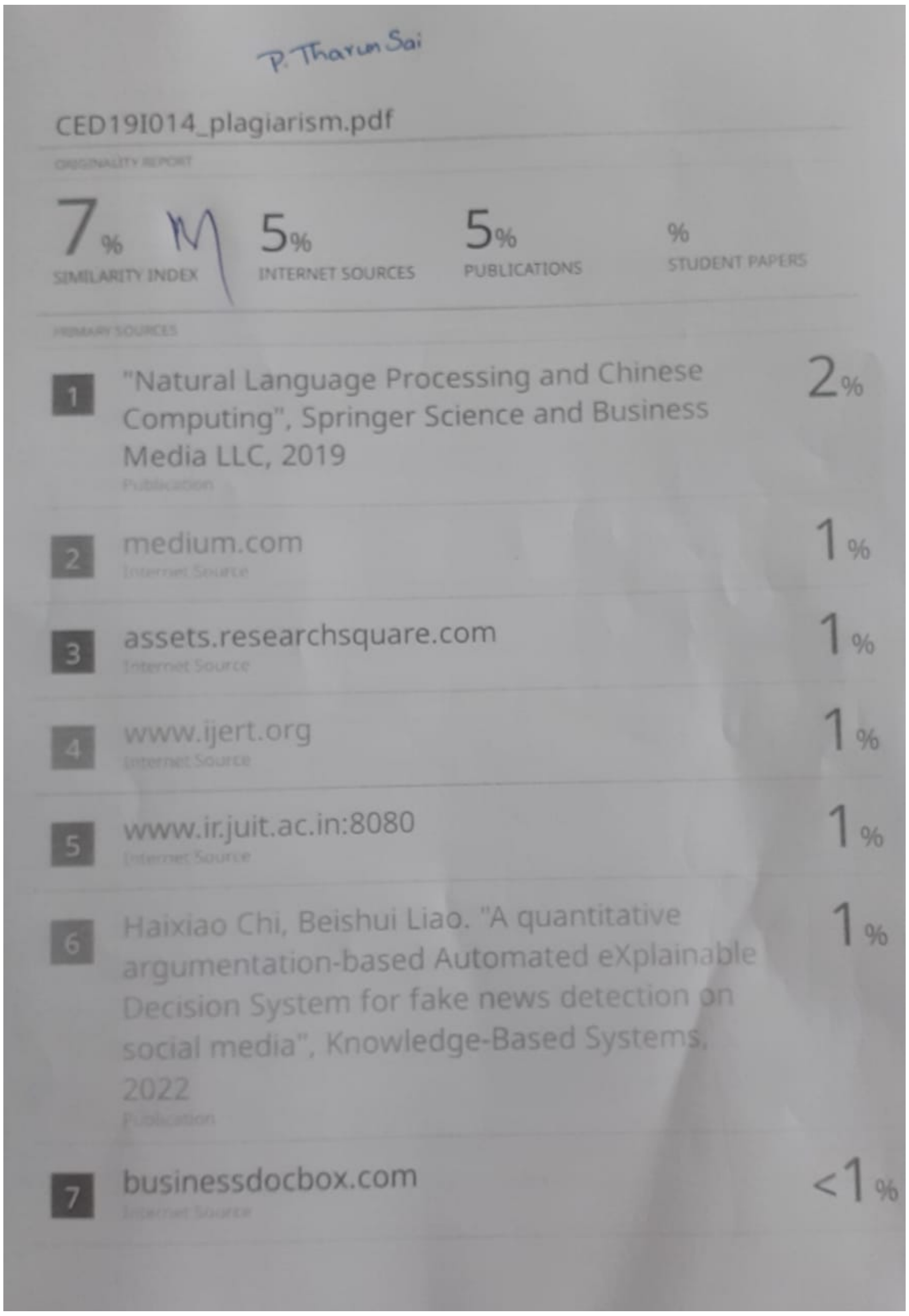
Weekly Review Report



23	11/03/2024 - 17/03/2024	Experimenting with Learning Models		
24	18/03/2024 - 24/03/2024	Comparing Learning Models		
25	25/03/2024 - 31/03/2024	Experimenting with BERT LM on Huggingface		
26	01/04/2024 - 07/04/2024	Finding How to load anduse BERT LM using python.		
27	08/04/2024 - 14/04/2024	Setting Environment in Google Colab for Loading LM		
28	15/04/2024 - 21/04/2024	Loading BERT and training with data		
29	22/04/2024 - 28/04/2024	Freezing Layers and continue Fine-tunning		
30	29/04/2024 - 05/05/2024	Evaluating the LM output		

Figure 1: Weekly Signature

Plagiarism Report



P. Tharun Sai

CED19I014_plagiarism.pdf

ORIGINALITY REPORT

7%	5%	5%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	"Natural Language Processing and Chinese Computing", Springer Science and Business Media LLC, 2019 Publication	2%
2	medium.com Internet Source	1%
3	assets.researchsquare.com Internet Source	1%
4	www.ijert.org Internet Source	1%
5	www.ir.juit.ac.in:8080 Internet Source	1%
6	Haixiao Chi, Beishui Liao. "A quantitative argumentation-based Automated eXplainable Decision System for fake news detection on social media", Knowledge-Based Systems, 2022 Publication	1%
7	businessdocbox.com Internet Source	<1%

Figure 2: Plagarsim