



University of
Salford
MANCHESTER

ASSESSMENT REPORT

ON

Text Classification

**Submitted
by**

**TOMISIN SAMUEL ISEDOWO
(@00749815)**

In fulfillment of

Natural Language Processing

For

MASTER OF SCIENCE IN ARTIFICIAL INTELLIGENCE

April 2024.

INTRODUCTION

Text classification is important in many aspects of natural language processing (NLP). Every day, a massive volume of unstructured text data is gathered from numerous sources, including articles, tweets, and posts on social media. It is possible to examine these unstructured messages and extract information. Text classification can be used for several natural language processing (NLP) applications, including subject identification, sentiment analysis, and prediction.

During this task, a binary classification was implemented. Binary classification is a type of text classification which intends to classify text into two classes. In this case, this task determines whether some news is fake, or some news is true based on its content.

When implementing a text classification, there are several models that can be used. For this task, three embedding models will be trained using the merged fake news and true news dataset. The three embedding models include NNLM-en-dim50/2, NNLM-en-dim50-with-normalization/2 and NNLM-en-dim128-with-normalization/2. All the three models have the capacity to convert sentences into embeddings vector which is the best way to represent the text.

The Fake and True dataset, which contains different news scraped from different article will be processed by removing stop words, new lines, words containing numbers, punctuations and text in square brackets, making the text lowercase, and some other preprocessing steps that is required to get the dataset ready for training.

The objective of this task is to use the merged fake and true news dataset to train the three pre-trained embedding models for the model to be able to automatically detect the fake news from the true news with a textual content. This model will help to address a range of societal, political, and cybersecurity challenges, ultimately contributing to a more informed and resilient society.

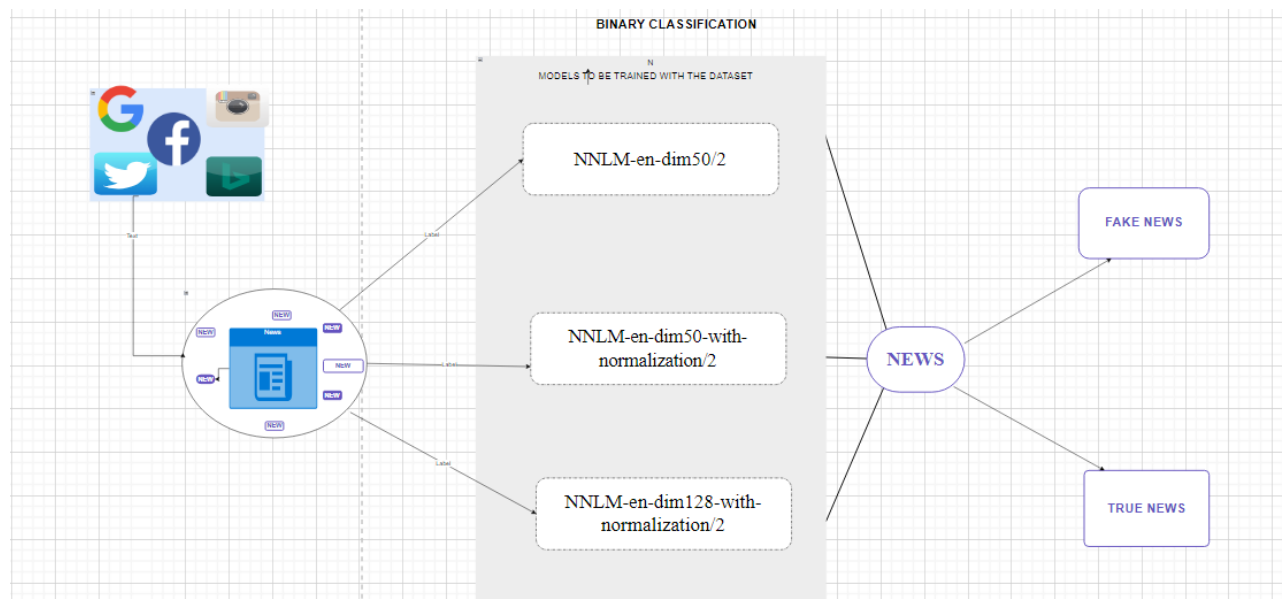


Fig 1: Binary Text classification architecture

LITERATURE REVIEW

Text Classification is the process of assigning several predefined labels to a large quantity of text. Preparing the text, training, predicting, extracting and choosing features, and assessing performance are the steps involved in text classification operations (Jindal et al, 2015). The predefined class assigned to the dataset depends on the dataset. During this task, we will be using a variety of pre-trained models used basically for text classification tasks and based on TensorFlow which is an open-source machine learning platform. Models like (Neural Network Language Model) NNLM-en-dim50/2, NNLM-en-dim50-with-normalization/2 and NNLM-en-dim128-with-normalization/2) will be used. NNLM-en-dim50/2 is a pre-trained text embedding model from TensorFlow Hub that helps to tackle the issue of text preprocessing and has a fixed size which makes processing to be easier. NNLM-en-dim128-with-normalization/2 is the same as NNLM-en-dim50-with-normalization/2 but has more text normalization which helps to remove punctuations or characters that handle in-vocabulary embeddings for tokens on the dataset.

There has been review of various studies related to text classification models, therefore, this task focused on ((Neural Network Language Model) NNLM-en-dim50/2, NNLM-en-dim50-with-normalization/2 and NNLM-en-dim128-with-normalization/2 for the binary text clarification which classify the dataset into fake and true news.

Subsequent research can continue to advance the state of the art in text embedding models, leading to more accurate, interpretable, and robust representations for a wide range of NLP applications.

DATASET

For this task, the Fake and True news dataset will be used to train the Model. In today's world, there are lots of means of disseminating false or misleading information, especially on social media.

Developing a fake news detection model can be highly beneficial especially for a country like Nigeria where many youths are affected mentally by so many false news flying every seconds especially in the context of its socio-political landscape and the prevalence of misinformation.

Here's how such a model could be useful:

1. **Combatting Misinformation:** I will say Nigeria, like many other countries, faces challenges related to the spread of misinformation, particularly through social media platforms. A fake news detection model could help identify and flag misleading or false information, thereby reducing the impact of misinformation on public opinion and decision-making.
2. **Protecting Democratic Processes:** In the context of elections and democratic processes, fake news can be used to manipulate public opinion, undermine trust in institutions, and even influence voting behavior. A robust fake news detection model can help safeguard the integrity of democratic processes by identifying and debunking false information related to elections and political candidates.

EXPLANATION AND PREPARATION OF THE DATASET

During this project, two datasets were gotten from Kaggle named Fake new and True news. The dataset is a csv file. The two datasets were uploaded to the Cloud environment (Google Colab) to prepare the dataset for processing.

After uploading the dataset, proper study was carried out on the dataset and necessary cleaning was done to prepare the dataset for the training.

This wordopt function was employed which appears to be a text preprocessing function designed to clean and normalize text data. The cleaning and normalization help to preprocess the input text data, making it cleaner and more suitable for further analysis such as text classification or sentiment analysis.

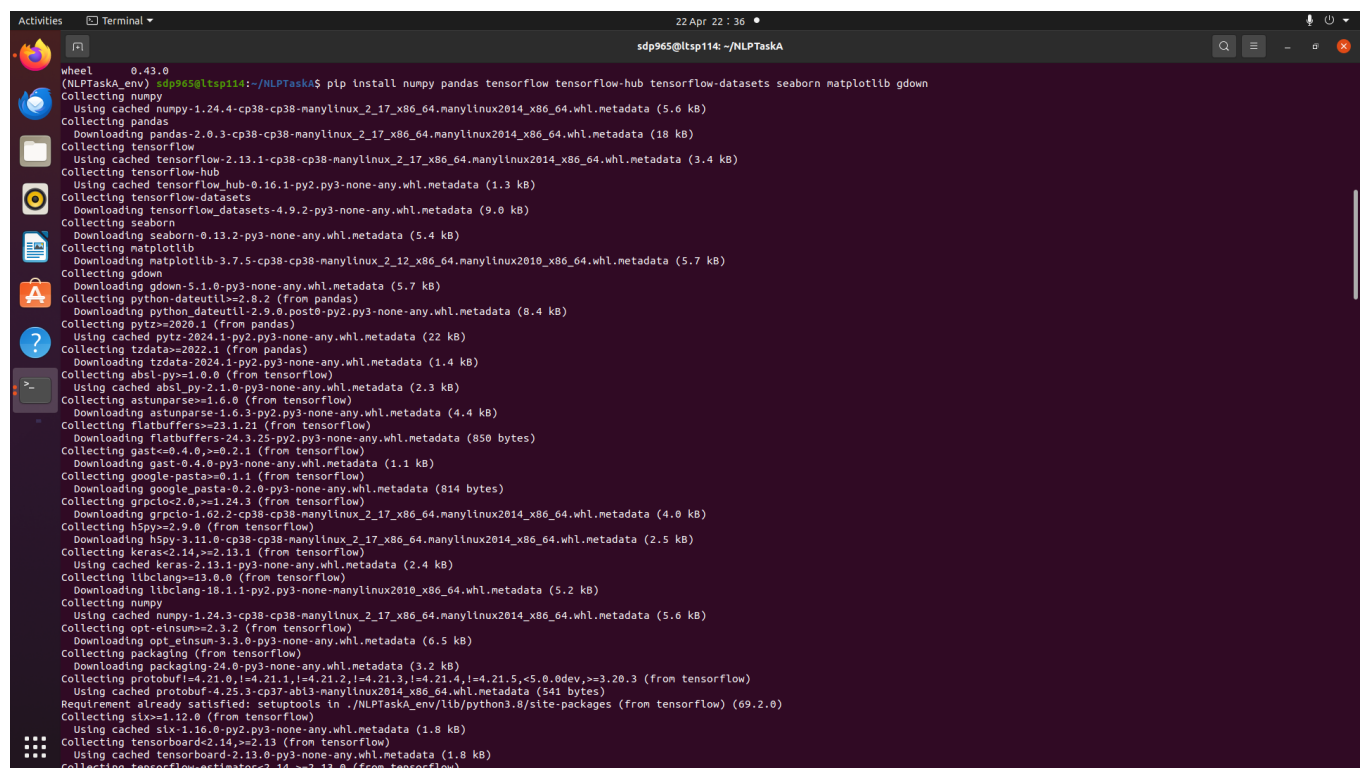
Let's break down each step:

- **text= text.lower():** This line converts all the text to lowercase. It's a common preprocessing step to ensure consistency in the text data, as it prevents the model from treating words with different cases as different entities.
- **text = re.sub('[.*?\\]', '', text):** This line removes square brackets and any text within them. It's often used to remove metadata or references that are enclosed within square brackets.
- **text = re.sub("\\W", " ", text):** This line replaces any non-word characters (such as punctuation and special characters) with a space. It effectively removes these characters from the text.
- **text = re.sub('https?:/\\S+|www\\.\\S+', '', text):** This helps to remove URLs from the each line of text on the dataset.
- **text = re.sub('[%s]' % re.escape (string.punctuation), '', text):** This helps to remove unnecessary punctuation characters from the text using the string. Punctuation set. Punctuation can often be noise in text data and removing it helps simplify the text for analysis.
- **text = re.sub('\\n', '', text):** This line removes newline characters from the text. Newlines are often used for formatting purposes and may not be relevant for text analysis tasks.
- **text = re.sub('\\w*\\d\\w*', '', text):** This line take all the alphanumeric characters that contain digits and remove them. It effectively removes any alphanumeric tokens that include numbers, which may not be relevant for some text analysis tasks.

METHODOLOGY

IMPLEMENTATION WAS DONE ON LOCAL ENVIRONMENT(LINUX) AND IN THE CLOUD (GOOGLE COLAB)

Linux stands as a cornerstone of the computing world, offering a powerful and versatile platform for users and developers because the environment is open source. Linux has become a mainstay in servers, desktops, mobile devices, and even embedded systems. The terminal aids in installing all the necessary packages needed for this task. Virtual environment was created for the purpose of this assessment which helps to isolate python packages and dependencies for the project, the virtual environment was activated using “source NLPTaskA_env/bin/activate”. The necessary packages are installed using this “\$ pip install numpy pandas tensorflow tensorflow-hub tensorflow-datasets seaborn matplotlib gdown before running the codes.



```
Activities Terminal 22 Apr 22:36 sdp965@ltsp114: ~/NLPTaskA
wheel 0.43.0
(NLPTaskA_env) sdp965@ltsp114:~/NLPTaskA$ pip install numpy pandas tensorflow tensorflow-hub tensorflow-datasets seaborn matplotlib gdown
Collecting numpy
  Using cached numpy-1.24.4-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (5.6 kB)
Collecting pandas
  Downloading pandas-2.0.3-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (18 kB)
Collecting tensorflow
  Using cached tensorflow-2.13.1-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (3.4 kB)
Collecting tensorflow-hub
  Using cached tensorflow_hub-0.16.1-py2.py3-none-any.whl.metadata (1.3 kB)
Collecting tensorflow-datasets
  Downloading tensorflow_datasets-4.9.2-py3-none-any.whl.metadata (9.0 kB)
Collecting seaborn
  Downloading seaborn-0.13.2-py3-none-any.whl.metadata (5.4 kB)
Collecting matplotlib
  Downloading matplotlib-3.7.5-cp38-cp38-manylinux_2_12_x86_64.manylinux2010_x86_64.whl.metadata (5.7 kB)
Collecting gdown
  Downloading gdown-5.1.0-py3-none-any.whl.metadata (5.7 kB)
Collecting python-dateutil<=2.8.2 (from pandas)
  Downloading python_dateutil-2.9.0.post0-py2.py3-none-any.whl.metadata (8.4 kB)
Collecting pytz<=2020.1 (from pandas)
  Using cached pytz-2024.1-py2.py3-none-any.whl.metadata (22 kB)
Collecting tzdata>=2022.1 (from pandas)
  Downloading tzdata-2024.1-py2.py3-none-any.whl.metadata (1.4 kB)
Collecting absl-py<=1.0.0 (from tensorflow)
  Using cached absl_py-2.1.0-py3-none-any.whl.metadata (2.3 kB)
Collecting astunparse<=1.6.0 (from tensorflow)
  Downloading astunparse-1.6.3-py2.py3-none-any.whl.metadata (4.4 kB)
Collecting flatbuffers<=23.1.21 (from tensorflow)
  Downloading flatbuffers-24.3.25-py2.py3-none-any.whl.metadata (850 bytes)
Collecting gast<=0.4.0,>=0.2.1 (from tensorflow)
  Downloading gast-0.4.0-py3-none-any.whl.metadata (1.1 kB)
Collecting google-pasta<=0.1.1 (from tensorflow)
  Downloading google_pasta-0.2.0-py3-none-any.whl.metadata (814 bytes)
Collecting grpcio<=2.0,>=1.20.2 (from tensorflow)
  Downloading grpcio-1.62.2-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (4.0 kB)
Collecting h5py<=2.9.0 (from tensorflow)
  Downloading h5py-3.11.0-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (2.5 kB)
Collecting keras<2.14,>=2.13.1 (from tensorflow)
  Using cached keras-2.13.1-py3-none-any.whl.metadata (2.4 kB)
Collecting libclang<=13.0.0 (from tensorflow)
  Downloading libclang-18.1.1-py2.py3-none-any.whl.metadata (5.2 kB)
Collecting numpy
  Using cached numpy-1.24.3-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (5.6 kB)
Collecting opt-einsum<=3.3.2 (from tensorflow)
  Downloading opt_einsum-3.3.0-py3-none-any.whl.metadata (6.5 kB)
Collecting packaging (from tensorflow)
  Downloading packaging-24.0-py3-none-any.whl.metadata (3.2 kB)
Collecting protobuf<4.21.0,!<4.21.1,!<4.21.2,!<4.21.3,!<4.21.4,!<4.21.5,<5.0.dev,>=3.20.3 (from tensorflow)
  Using cached protobuf-4.25.3-cp37-abi3-manylinux2014_x86_64.whl.metadata (541 bytes)
Requirement already satisfied: setuptools in ./NLPTaskA_env/lib/python3.8/site-packages (from tensorflow) (69.2.0)
Collecting six>=1.12.0 (from tensorflow)
  Using cached six-1.16.0-py2.py3-none-any.whl.metadata (1.8 kB)
Collecting tensorboard<2.14,>=2.13 (from tensorflow)
  Using cached tensorboard-2.13.0-py3-none-any.whl.metadata (1.8 kB)
Collecting tensorflow-estimator<2.14,>=2.13.0 (from tensorflow)
```

Fig 2: Implementation on Linux terminal

Google Colab denotes Google Collaboratory, which is a cloud-based Jupiter notebook environment. It is an adaptable platform for data analysis, machine learning, and Python programming. in a browser-based interface (Cloud), with no setup required. Google Colab offers free access to computing resources, including GPUs and TPUs, making it an ideal platform for machine learning and data analysis tasks to have an efficient and accessible solution for a wide range of tasks, from machine learning research to educational projects and collaborative work.

USED LIBRARIES IN THE TASK

The below libraries are used:

- **NumPy (np):** NumPy provides support for large, multi-dimensional arrays, along with some mathematical functions to work on the arrays efficiently.
- **pandas (pd):** Pandas is a powerful data manipulation and analysis library built on NumPy. It make it easy to work with structured data such as CSV files, Excel spreadsheets, and databases. Pandas provide functionalities for data cleaning, transformation, and analysis.
- **TensorFlow (tf):** TensorFlow is a library for building and deploying machine learning models, particularly deep learning models. TensorFlow supports both English like APIs and lower-level APIs for more flexibility in model building.
- **tensorflow_hub (hub):** TensorFlow Hub is a library for reusable machine learning modules. It provides a repository of pre-trained models and modules that can be easily integrated into TensorFlow workflows. These pre-trained models cover various tasks such as image classification, text embedding, and natural language processing.
- **tensorflow_datasets (tfds):** TensorFlow Datasets is a library that provides access to a collection of commonly used datasets for machine learning research. It simplifies the process of downloading and preprocessing datasets, making them ready for use with TensorFlow models.
- **warnings:** The warnings module provides a simple interface to control the display of warning messages in Python.

- **seaborn (sns):** Seaborn helps to simplify the process of creating complex visualizations such as histograms, scatter plots, and heatmaps.
- **matplotlib.pyplot (plt):** Matplotlib library is commonly used for creating plots representation like line plots, bar charts, and pie charts.
- **sklearn.model_selection:** Scikit-learn is a popular machine learning library in Python. The model_selection module provides utilities for splitting datasets into training and testing sets, cross-validation, and parameter tuning.
- **sklearn.metrics:** The sklearn.metrics helps to see the accuracy of the result with parameters such as F1 score, precision, accuracy and confusion matrix etc.
- **re:** The 're' module helps in text preprocessing, parsing, and searching.
- **string:** The string module includes utilities that handles characters, formatting strings, and working with ASCII characters. In this context, it may be used for removing punctuation characters from text data.

These libraries together provide a comprehensive set of tools for data manipulation, visualization, machine learning, and text processing in Python.

EXPERIMENTAL PROCEDURE, SETTING, AND OPTIMIZATION OF MODEL HYPERPARAMETERS

DATA PREPARATION

Data preparation is a crucial step in Text classification which involves cleaning, transforming, and organizing dataset into a form that is suitable to train the model.

While preparing the dataset for this task, the Fake and True news datasets are merged to have a single dataset. Also, a few columns that are not needed are removed. Then checked for stop words, changed all text to lowercase, removed links from the text etc.

We converted the class which is the label to vector. The conversion is done using a label encoder from sklearn library the model understanding.

The cleaned dataset is split into training, testing, and validation set. The dataset contains 80% to training (X_train and y_train), 20% of the dataset was split into 10% testing (X_test, y_test) and 10% validation (X_val, y_val).

MODEL SELECTION

The Neural Network Language Model (NNLM) is a token-based text embedding model. NNLM is a type of neural network architecture designed for natural language processing (NLP) tasks, such as language modeling and text generation. It's primarily used for learning distributed representations of words (word embeddings) from large text.

HYPERPARAMETERS USED

The list below shows the libraries used:

Activation functions: The function used for this task include ReLU (Rectified Linear Unit).

ReLU: Rectified Linear Unit is an activation function that helps the model to understand complex relationships in the data.

Optimizer: Optimizer used for this task is Adam which denote Adaptive Moment Estimation. This helps in adjusting the parameters during the implementing of the training dataset.

Number of Neurons: The number of neurons used during this task is 32. Other layers like 16, 32, 64 was tried in this task and 32 seems like the best for the model.

Metric: This task is a binary class classification. For binary classification tasks, several metrics are commonly used to evaluate the performance of a model. The metric compares the model prediction with the original label and calculates the percentage of the correctly predicted ones to the total predicted. For this task, confusion metrics were adopted.

Loss Function: Sparse Categorical Cross entropy is used as the loss function because the task is a multi-class classification. This computes the loss between the original labels and the predicted ones to know the loss percentage of how many times the model predicted wrongly. We want the model to predict correctly so reducing the loss function will improve the ability of the model.

Batch size: A batch size of 512 is used for the models used which means the model will train on 512 training data in its first iteration. The amount of training data used in a single iteration during the training process.

Epochs: For this task, the epoch is set at 10 so the model will iterate ten times in the training section. The number of epochs helps to determine the number of times the model has to train over the training dataset.

OPTIMIZING MODEL ARCHITECTURE AND HYPERPARAMETERS

This analysis delves into the training and validation performance of the model over 10 epochs. The model initiates training on the first epoch out of 10. Each epoch comprises 41 batches.

The time notation, 69s 2s/step, indicates the duration of each step (or batch). Each step consumes approximately 2 seconds, with the entire epoch lasting around 69 seconds.

The reported values, loss: 0.0030- accuracy: 0.9999, denote the loss and accuracy of the training data for the current epoch. Here, the loss is 0.0030, and the accuracy stands at 99.99%. Conversely, val_loss: 0.0420 - val_accuracy: 0.9883, represents the loss and accuracy on the validation data for the same epoch. The validation loss measures 0.0420, with an accuracy of 98.83%.

As the epochs progress, both training and validation metrics exhibit an upward trend, signaling improvement in model performance. Concurrent monitoring of validation metrics alongside training metrics helps identify potential overfitting. A significant disparity between training and validation accuracy might indicate overfitting, necessitating the adoption of regularization techniques.

VISUALIZATION OF RESULTS

MODEL: NNLM-en-dim50/2

```
batch_size=512,  
validation_data=(x_val, y_val),  
verbose=1)  
  
Epoch 1/10  
41/41 [=====] - 82s 2s/step - loss: 0.3941 - accuracy: 0.8623 - val_loss: 0.2263 - val_accuracy: 0.9319  
Epoch 2/10  
41/41 [=====] - 69s 2s/step - loss: 0.1531 - accuracy: 0.9550 - val_loss: 0.1189 - val_accuracy: 0.9669  
Epoch 3/10  
41/41 [=====] - 67s 2s/step - loss: 0.0764 - accuracy: 0.9804 - val_loss: 0.0783 - val_accuracy: 0.9777  
Epoch 4/10  
41/41 [=====] - 69s 2s/step - loss: 0.0419 - accuracy: 0.9913 - val_loss: 0.0597 - val_accuracy: 0.9832  
Epoch 5/10  
41/41 [=====] - 68s 2s/step - loss: 0.0246 - accuracy: 0.9962 - val_loss: 0.0504 - val_accuracy: 0.9851  
Epoch 6/10  
41/41 [=====] - 69s 2s/step - loss: 0.0148 - accuracy: 0.9983 - val_loss: 0.0452 - val_accuracy: 0.9868  
Epoch 7/10  
41/41 [=====] - 68s 2s/step - loss: 0.0092 - accuracy: 0.9991 - val_loss: 0.0432 - val_accuracy: 0.9869  
Epoch 8/10  
41/41 [=====] - 67s 2s/step - loss: 0.0061 - accuracy: 0.9997 - val_loss: 0.0423 - val_accuracy: 0.9877  
Epoch 9/10  
41/41 [=====] - 78s 2s/step - loss: 0.0042 - accuracy: 0.9998 - val_loss: 0.0421 - val_accuracy: 0.9875  
Epoch 10/10  
41/41 [=====] - 68s 2s/step - loss: 0.0030 - accuracy: 0.9999 - val_loss: 0.0420 - val_accuracy: 0.9883  
  
[33] #This is evaluating the model with the whole test dataset
```

Fig 3: NNLM-en-dim50/2 training log

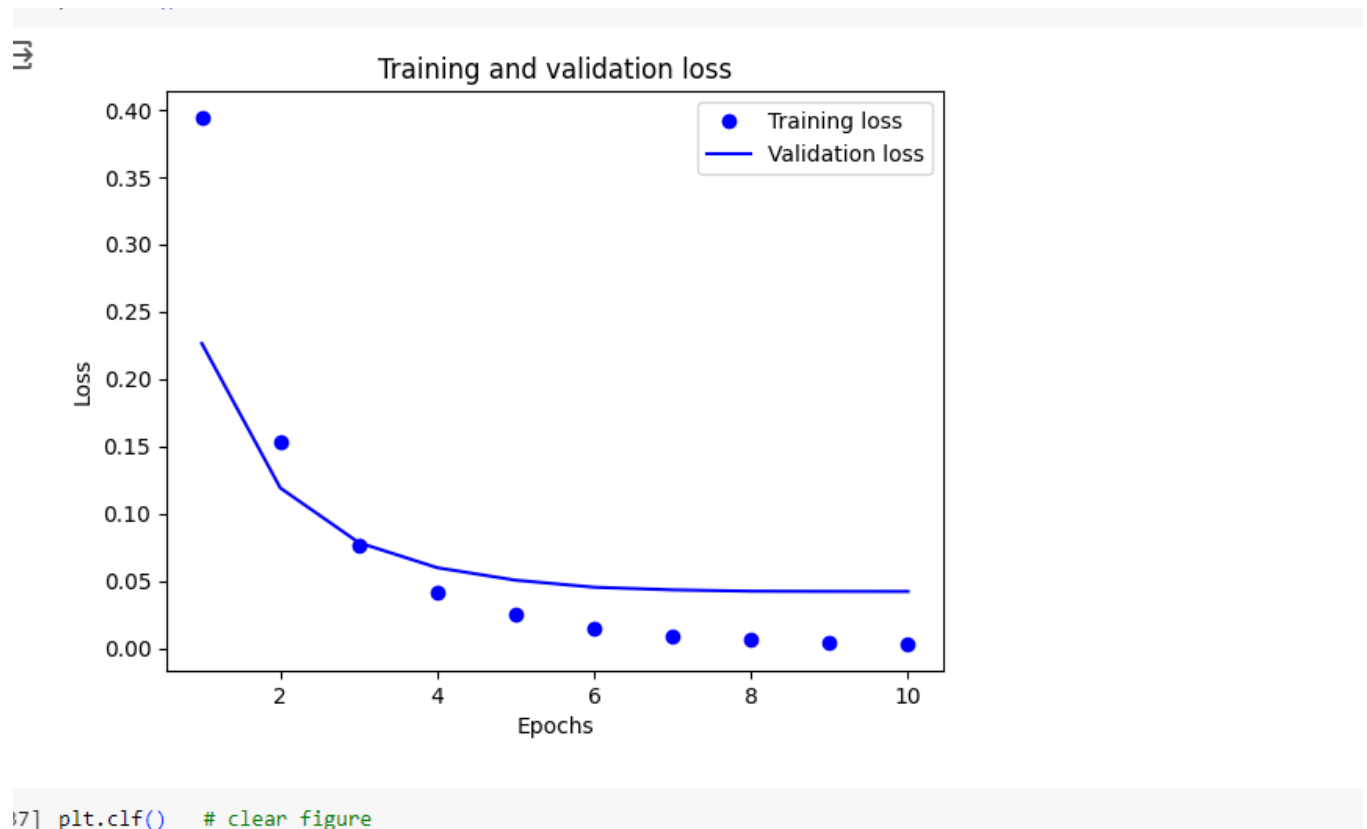


Fig 4: NNLM-en-dim50/2 training and validation loss graph

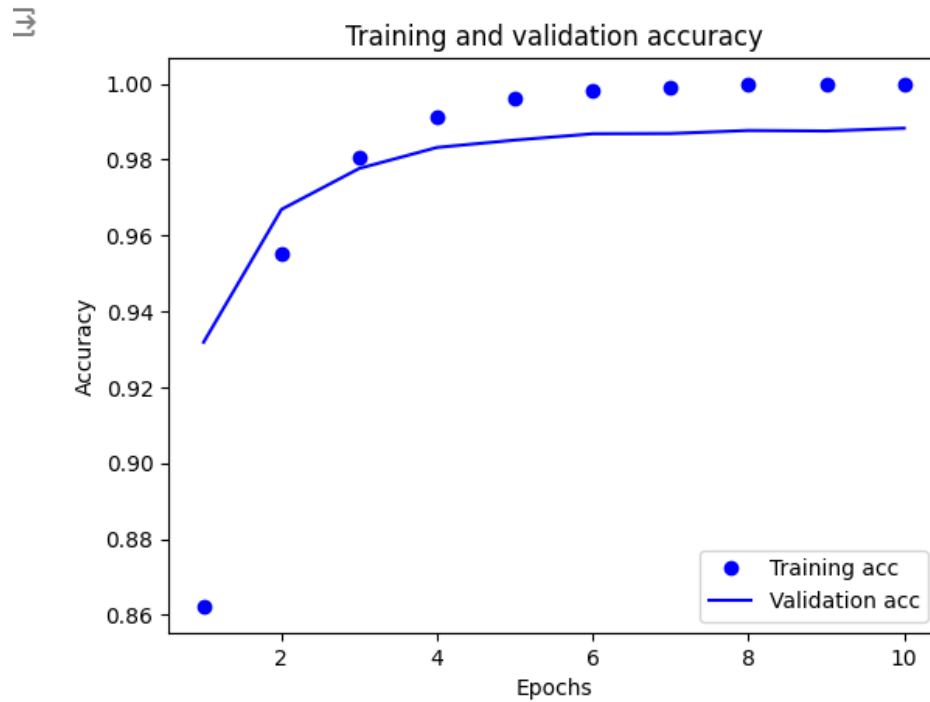


Fig 5: NNLM-en-dim50/2 training and validation accuracy graph

```
disp = ConfusionMatrixDisplay(confusion_matrix=confusion, display_labels=[0, 1])
```

Accuracy: 0.9878619153674832
Precision: 0.9937634408602151
Recall: 0.982982344182089
F1: 0.9883434926745803
Confusion Matrix:

4250	29
80	4621

```
[39] # Compute confusion matrix  
conf_matrix = confusion_matrix(y_test, y_pred)
```

Fig 6: NNLM-en-dim50/2 confusion matrix and classification report

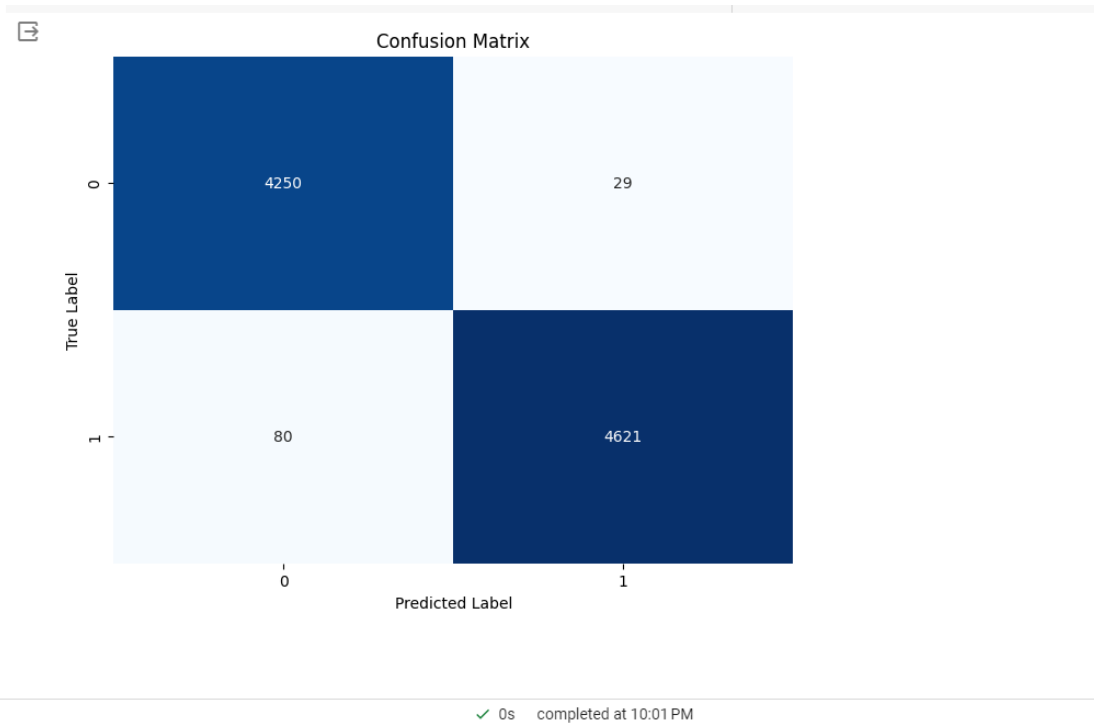


Fig 7: NNLM-en-dim50/2 confusion matrix

MODEL: NNLM-en-dim50-with-normalization/2

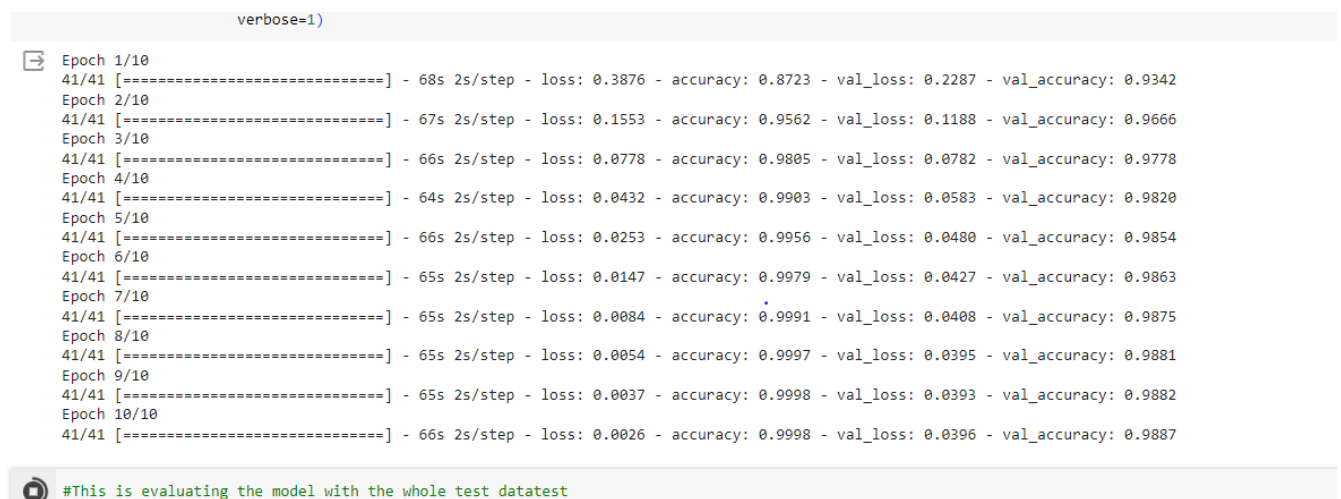
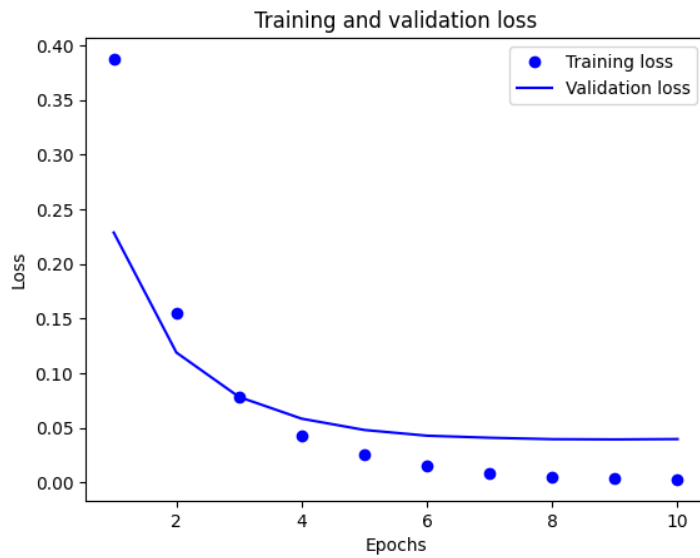
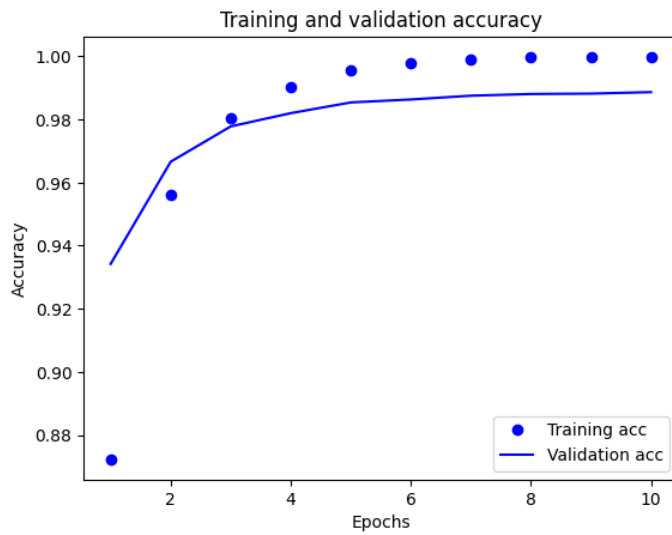


Fig 8: NNLM-en-dim50-with-normalization/2 training log



```
plt.clf() # clean figure
```

Fig 9: NNLM-en-dim50-with-normalization/2 training and validation loss graph



```
501 import numpy as np
```

Fig 10: NNLM-en-dim50-with-normalization/2 training and validation accuracy graph

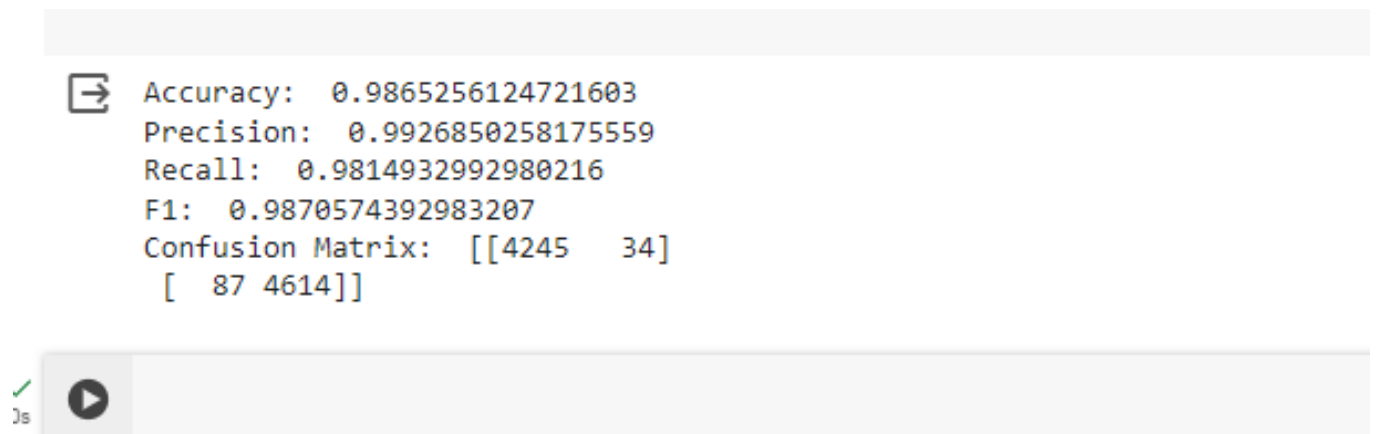


Fig 11: NNLM-en-dim50-with-normalization/2 confusion matrix and classification report

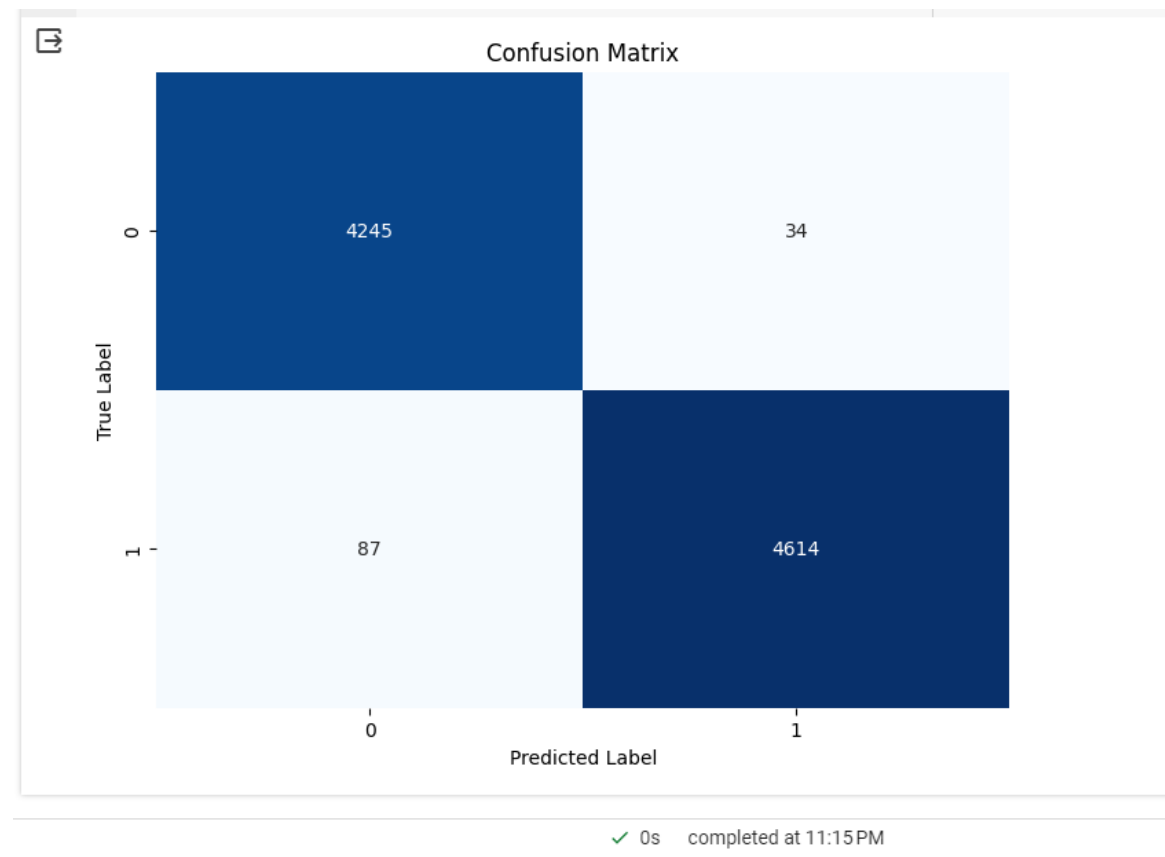


Fig 12: NNLM-en-dim50-with-normalization/2 confusion matrix

NNLM-en-dim128-with-normalization/2

```
*** Epoch 1/10
41/41 [=====] - 175s 4s/step - loss: 0.3522 - accuracy: 0.8946 - val_loss: 0.1631 - val_accuracy: 0.9552
Epoch 2/10
41/41 [=====] - 163s 4s/step - loss: 0.1011 - accuracy: 0.9732 - val_loss: 0.0781 - val_accuracy: 0.9785
Epoch 3/10
41/41 [=====] - 163s 4s/step - loss: 0.0416 - accuracy: 0.9904 - val_loss: 0.0525 - val_accuracy: 0.9852
Epoch 4/10
41/41 [=====] - 172s 4s/step - loss: 0.0194 - accuracy: 0.9968 - val_loss: 0.0439 - val_accuracy: 0.9874
Epoch 5/10
41/41 [=====] - 156s 4s/step - loss: 0.0101 - accuracy: 0.9990 - val_loss: 0.0406 - val_accuracy: 0.9885
Epoch 6/10
41/41 [=====] - 160s 4s/step - loss: 0.0055 - accuracy: 0.9997 - val_loss: 0.0394 - val_accuracy: 0.9887
Epoch 7/10
41/41 [=====] - 163s 4s/step - loss: 0.0033 - accuracy: 0.9998 - val_loss: 0.0392 - val_accuracy: 0.9891
Epoch 8/10
41/41 [=====] - 174s 4s/step - loss: 0.0022 - accuracy: 0.9999 - val_loss: 0.0397 - val_accuracy: 0.9889
Epoch 9/10
41/41 [=====] - 162s 4s/step - loss: 0.0016 - accuracy: 0.9999 - val_loss: 0.0403 - val_accuracy: 0.9889
Epoch 10/10
41/41 [=====] - ETA: 0s - loss: 0.0013 - accuracy: 1.0000
```

Fig 13: NNLM-en-dim128-with-normalization/2 training log

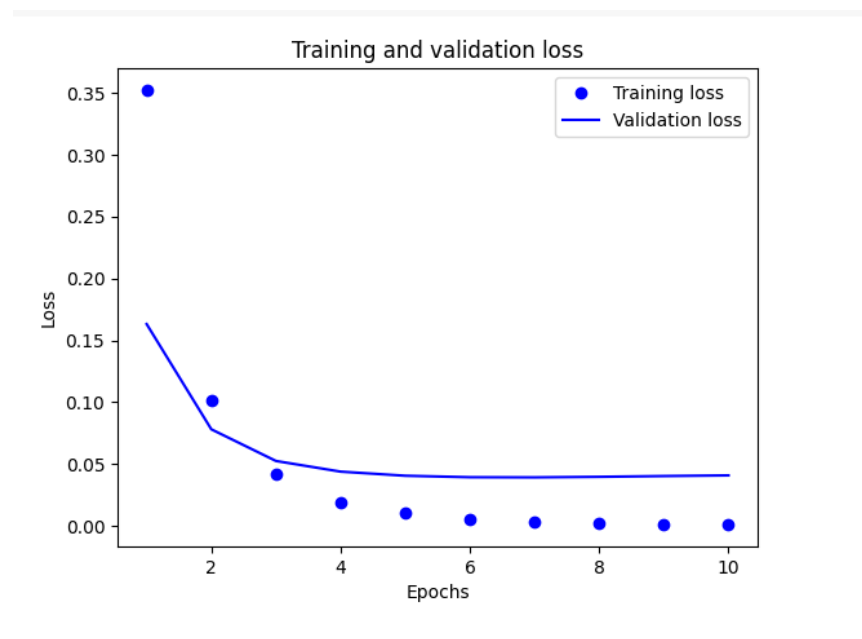


Fig 14: NNLM-en-dim128-with-normalization/2 training and validation loss graph

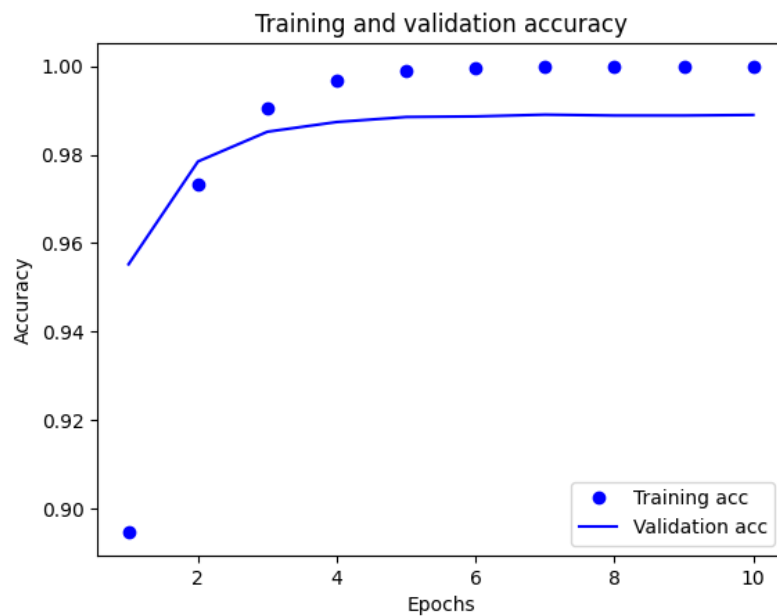


Fig 15: NNLM-en-dim128-with-normalization/2 training and validation accuracy graph

```
Accuracy: 0.9864142538975501
Precision: 0.99247149924715
Recall: 0.9814932992980216
F1: 0.986951871657754
Confusion Matrix: [[4244  35]
 [ 87 4614]]
```

✓ [64]
0s

Fig 16: NNLM-en-dim128-with-normalization/2 confusion matrix and classification report

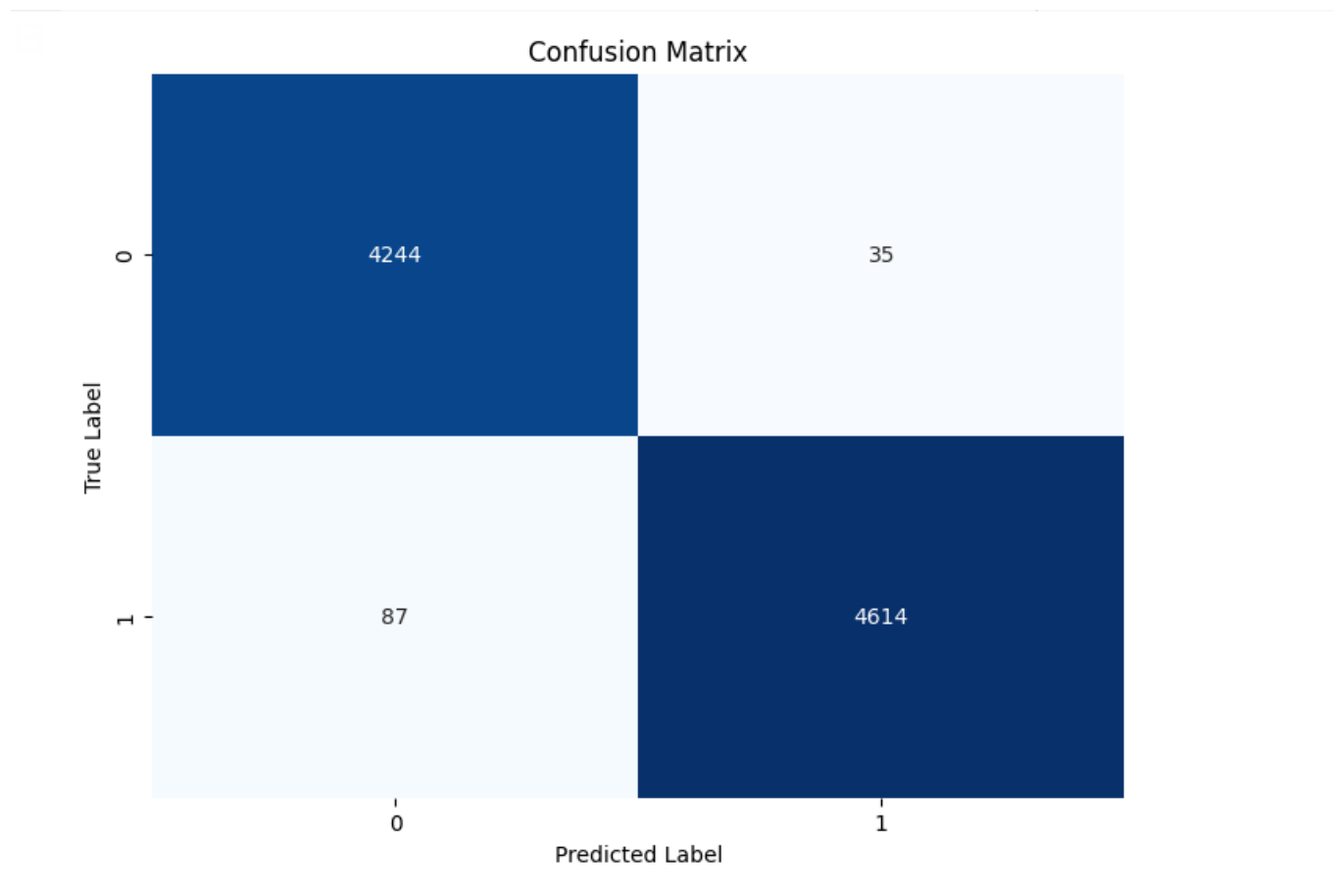


Fig 17: NNLM-en-dim128-with-normalization/2 confusion matrix

RESULTS ANALYSIS AND DISCUSSION

For this task, a confusion metric was used to evaluate the model. The confusion matrix helps to evaluate the performance of the three models. The help to read and understand the performance of the three models by presenting the actual and predicted classes in a tabular format. These metrics help in understanding how well the model is performing and where it may be making errors. They are crucial for evaluating the effectiveness of a classification model. After evaluating the three models, it is evident that the NNLM-en-dim50/2 consistently outperforms the others in terms of accuracy. This indicates that it is more effective in correctly predicting for binary class text classification. This shows that NNLM-en-dim50/2 is a suitable choice for the task, providing higher accuracy compared to alternative models.

Using neural network-based language models (such as NNLM), fake news detection involves several ethical, legal, and professional considerations:

1. Ethical Considerations:

Fairness and Bias: Neural networks can inadvertently amplify biases present in the training data, leading to unfair treatment of certain groups.

Transparency and Interpretability: Neural network models are often seen as "black boxes," making it challenging to understand how they arrive at their decisions. Ensuring transparency and providing explanations for model predictions can enhance accountability and trustworthiness.

Privacy: Fake news detection datasets may contain sensitive information about individuals or organizations. It's crucial to handle data privacy and anonymity appropriately to protect the identities and rights of individuals mentioned in the news articles.

2. Legal Considerations

Data Privacy Regulations: Depending on the jurisdiction, there may be legal requirements by law regarding the collection of information. Regarding this regulations by European Union, this is essential to avoid legal consequences.

Intellectual Property Rights: Ensure that the dataset used for training the fake news detection model does not infringe on any copyrights or intellectual property rights. Unauthorized use of copyrighted material can lead to legal disputes and liability.

3. Professional Considerations:

Accuracy and Reliability: Fake news detection models should strive for high accuracy and reliability to minimize the spread of misinformation. It's essential to rigorously evaluate the model's performance and continuously improve its effectiveness.

Responsible Deployment: Deploying fake news detection models in real-world applications requires careful consideration of potential consequences. Models should be thoroughly tested to minimize the risk of false positives or false negatives, which could harm individuals or organizations.

Engagement with Stakeholders: Engaging with stakeholders, including journalists, news organizations, and the public, can provide valuable insights into the ethical implications of fake news detection. Collaborating with relevant stakeholders can help ensure that the model's deployment aligns with societal values and norms

CONCLUSION

Getting this result, the merged dataset was used to train the classification model and the model can classify news with respect to the predicted class correctly, which is our aim in this task. The best accuracy was the result gotten from NNLM-en-dim50/2 model. The training process involved iterating over the dataset for a fixed number of epochs while adjusting the model parameters to minimize the loss function. Each epoch consisted of multiple batches, and the model's performance was evaluated on both the training and validation datasets after each epoch.

The trained text classification model shows promising performance with high accuracy and effective generalization. Further fine-tuning and optimization may lead to even better results. This report provides insights into the model's training process and performance evaluation, laying the groundwork for future improvements and applications.

REFERENCES

Jindal R., Malhotra R., Jain A. (2015). Techniques for text classification: Literature review and current trends. *Webology*, 12(2), 1–28.

Papers with Code - Text Classification. (n.d.). Paperswithcode.com.

<https://paperswithcode.com>. <https://api.research-repository>.

Roque, L. (2022, November 30). *Neural Machine Translation using a Seq2Seq Architecture and Attention (ENG to POR)*. Medium. https://towardsdatascience.com/neural-machine-translation-using-a-seq2seq-architecture-and-attention-eng-to-por-fe3cc4191175?gi=aea149fd168c&source=rss_artificial_intelligence-5

Wyawahare, C. (2020, January 10). *Tools and Libraries for Computer Vision*. Analytics Vidhya. <https://medium.com/analytics-vidhya/tools-and-libraries-for-computer-vision-ac2769a2744e>