NORTHWESTERN UNIVERSITY | SCHOOL OF PROFESSIONAL STUDIES

Project Report: Fake News Detection Using the LIAR Dataset

MS DSP 453 – Natural Language Processing

Sachin Sharma

December 06, 2024

Introduction

The spread of fake news is a growing problem in today's digital world. With the rise of social media and online platforms, fake news has become easier to create and share, often misleading people and causing harm to society. This makes it crucial to develop methods for identifying and stopping fake news effectively.

The goal of this project is to use machine learning techniques to detect fake news. Specifically, we aim to classify statements into categories like "true" or "fake" by analysing the content and patterns within the text. By leveraging data and advanced algorithms, we can build a model that not only identifies fake news but does so with accuracy and reliability.

Our work started with collecting a dataset of statements labelled as true or fake, along with additional information like the subject, speaker, and their political affiliation. This data was processed and cleaned to make it suitable for machine learning. Then, we applied different models to the problem, including traditional machine learning algorithms like Naive Bayes and Random Forest, as well as advanced deep learning techniques like Neural Network and BERT. In this report, we summarize the challenges faced during the project, the methods we used, and the results we achieved.

Data and Challenges

Describing the Data

The dataset used for this project was comprehensive, containing 12,787 rows and 14 columns. Below is an overview of the key columns:

Column Name	Description
[ID].json	Unique identifier for each record, stored as a JSON file.

label	The truthfulness of the statement, categorized as "true," "mostly true," "half-true," "false," or "pants on fire."
statement	The text of the statement being evaluated.
subject(s)	The topic or category of the statement, e.g., "immigration," "economy," "health-care."
speaker	The person who made the statement.
speaker's job title	The profession or position of the speaker, such as "Governor" or "U.S. Representative."
state info	The state associated with the speaker or statement, e.g., "Texas."
party affiliation	The political party affiliation of the speaker, e.g., "Republican" or "Democrat."
barely true counts	Number of past statements by the speaker rated as "barely true."
false counts	Number of past statements by the speaker rated as "false."
half true counts	Number of past statements by the speaker rated as "half-true."
mostly true counts	Number of past statements by the speaker rated as "mostly true."
pants on fire counts	Number of past statements by the speaker rated as "pants on fire" (a high degree of falsity).
venue	The medium or platform where the statement was made, such as "social media" or "news."

Addressing Challenges

The data presented several unique challenges that required specific preprocessing steps:

- **Missing and Unknown Values:** Some columns in the dataset, such as the speaker's job title, state information, and venue, contained missing or "unknown" values. To maintain consistency and completeness in the data, these missing values were replaced with a placeholder labelled as "Unknown."

- **Imbalance in Categories:** There was an imbalance in the distribution of labels (e.g., more statements categorized as "false" than "pants on fire"). This imbalance could bias the model towards the majority class, requiring consideration during evaluation.
- **Multi-Label Nature of Subjects:** The subject(s) column contained multiple topics separated by commas, making it a multi-label feature. To address this, subjects were tokenized, processed to remove stopwords, and grouped into broader categories like "economy," "politics," and "health-care."
- **Textual Data Preprocessing:** The statement column, which contained raw text data, underwent preprocessing to prepare it for analysis. This included removing URLs, email addresses, numbers, and punctuation, converting all text to lowercase for uniformity, and applying stemming and lemmatization to standardize word forms. Additionally, the text was tokenized into individual words, and TF-IDF vectorization was applied to transform the text into a structured format suitable for analysis.
- **Categorical Variables:** Columns like venue, speaker's job title, and party affiliation contained categorical data, which were transformed into numeric codes for use in machine learning models. For example, "Republican" was mapped to 0, and "Democrat" was mapped to 1 in the party affiliation column.
- **Contextual Information:** Features such as the historical truthfulness of speakers (barely true counts, false counts, etc.) and speaker's political affiliation provided context for classification but required careful integration into the modeling pipeline.

Architecture, Design And Modeling Methods

In our project, we designed a robust Natural Language Processing (NLP) pipeline to classify statements into truthfulness categories. Below is a detailed summary of the architecture, algorithms, key features, and challenges encountered during implementation.

Architecture Overview

Our NLP architecture consisted of the following steps:

- **Data Preprocessing**: Textual data from the statements was cleaned and vectorized using TF-IDF to extract features for machine learning models. Categorical variables, such as the speaker's job title and venue, were encoded using ordinal or one-hot encoding to ensure compatibility with the models. Numerical variables, such as truthfulness counts, were normalized to improve feature scaling.
- **Feature Engineering**: Textual features derived from TF-IDF were combined with contextual features such as party affiliation, state information, and venue to create a richer input dataset. Additionally, multi-label encoding was applied to the subjects column, allowing the model to account for multiple subjects simultaneously and capture more nuanced relationships in the data.
- **Modeling**: Initial experimentation involved applying traditional machine learning algorithms, such as Naive Bayes, Random Forest, and Decision Trees. Further analysis explored deep learning architectures, including a custom neural network and transformers like BERT and RoBERTa, to capture nuanced semantic information from the text for improved predictive performance.
- **Evaluation**: Used metrics like accuracy, precision, recall, F1-score, log loss, confusion matrices, and classification reports to assess model performance.

Algorithms

We evaluated multiple algorithms across various stages of the project.

- Naive Bayes
- Random Forest

- Decision Trees
- Neural Network
- BERT (Base)
- BERT (RoBERTa)

Design and Key Features of the Code

The implementation had several key features that enhanced its functionality and adaptability:

- **Text Vectorization**: We used TF-IDF Vectorization to convert textual data into numerical representations. We selected trigrams (ngram_range=(3,3)) to capture word sequences, improving classification accuracy by emphasizing contextual phrases.
- **Categorical Data Encoding**: Columns like speaker, venue, and party affiliation were converted into numerical values using Pandas Categorical Encoding. This ensured compatibility with machine learning models.
- **Cross-Validation**: K-Fold Cross-Validation (with 5 folds) was implemented for all models to ensure that the evaluation metrics were reliable and generalized.
- **Deep Learning Architecture**: Custom implementations of BERTClassifier and RoBERTaClassifier utilized pre-trained transformers from Hugging Face's library. Both models were fine-tuned on the dataset to enhance performance for the classification task.
- **Evaluation Metrics**: Comprehensive evaluation using multiple metrics and visualizations, including: Confusion Matrices**,** Classification Reports**,** Log Loss**.**

Results and Interpretation

The table below provides metrics achieved by each algorithm and we have provided detailed analysis on result, performance and confusion matrix in the Appendix section:

Experiment	**Model**	**Train Time**	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**Validation Loss**
Experiment 1	Random Forest	86.13 seconds	0.929	0.930	0.911	0.918	0.481
Experiment 2	Naive Bayes	6.96 seconds	**0.999**	**0.999**	**0.999**	**0.999**	**0.0036**
Experiment 3	Decision Trees	6.61 seconds	0.918	0.861	0.918	0.884	0.119
Experiment 4	Neural Networks	2523.91 seconds	0.950	0.954	0.942	0.942	0.116
Experiment 5	BERT Base	293.22 seconds	0.781	0.682	0.781	0.720	0.567
Experiment 6	RoBERTa Base	286.96 seconds	0.790	0.669	0.790	0.718	0.516

Interpretation of Results

Model	**Expectation**	**Actual**
Naive Bayes	Expected to perform well on text data with balanced feature representation due to its simplicity and reliance on conditional probability but might struggle with complex interactions between features.	Surpassed expectations with the highest accuracy (99.89%) and lowest validation loss (0.0036). The preprocessing pipeline (TF-IDF vectorization) effectively highlighted word-level patterns indicative of truthfulness.

Decision Trees	Expected to provide decent accuracy with easy interpretability, leveraging key features like the speaker's job title or political affiliation.	Achieved good accuracy but struggled with overfitting, performing well in training but less generalizable on new data. Strength lay in interpretability, showing feature contributions clearly, though performance lagged behind Naive Bayes.
Random Forest	Expected to deliver robust performance with reasonable accuracy and low susceptibility to overfitting as an ensemble method.	Achieved a commendable accuracy of 92.93% but fell short of Naive Bayes. While it handled categorical and numerical features well, it was computationally intensive and less precise in detecting nuanced patterns from text-derived features.
Neural Network	Anticipated to capture complex relationships between features, achieving high accuracy at the cost of computational resources and requiring meticulous tuning.	Performed well with 95.05% accuracy but required substantial training time (42 minutes) and high resource demands. The marginal improvements over simpler models did not justify the complexity.
Transformers	Expected to excel due to deep contextual understanding and	Underperformed with BERT achieving 78.14% accuracy and RoBERTa 79.01%. Likely due to

	semantic capabilities, significantly outperforming traditional models.	the dataset's structure and the adequacy of TF-IDF in capturing key patterns, making transformer-level complexity unnecessary for this case.

Insights and Findings

This project provided valuable insights into leveraging machine learning and natural language processing (NLP) to evaluate the truthfulness of political statements. By experimenting with various models and approaches, we identified effective methods for analysing our data. Among the tested models, the Naive Bayes algorithm emerged as the top performer, achieving an impressive accuracy of 99.89%. This result highlights that simple models can outperform more complex ones when paired with well-engineered features. The TF-IDF vectorizer proved particularly effective in capturing critical textual patterns, enabling Naive Bayes to excel. Conversely, advanced models like BERT and RoBERTa, while offering deeper linguistic understanding, required significantly more computational resources and provided only marginally better results for this dataset.

Feature engineering played a crucial role in the success of our models. Combining textual data with contextual features, such as the speaker's job title, political affiliation, and venue, enriched the dataset and improved predictions. For instance, statements made in formal contexts like government offices were generally more truthful compared to those from informal settings like social media. Additionally, organizing subjects into meaningful categories and thoroughly cleaning the text data enhanced model accuracy and interpretability. While neural networks and transformer models are well-suited for large and intricate datasets, this project demonstrated that straightforward approaches can be highly efficient and effective for moderate-sized datasets. Overall, this work provides a robust foundation for combating misinformation and supports future advancements in AI-driven truthfulness assessment.

Incorporating ChatGPT: Incorporating ChatGPT into the project proved beneficial for generating detailed explanations of results, refining text preprocessing pipelines, and brainstorming feature engineering techniques. ChatGPT’s ability to articulate complex concepts in simple terms aided in documenting our findings and debugging issues effectively. However, a key challenge was ensuring the alignment between ChatGPT’s generalized advice and the specific nuances of the dataset. For instance, while it provided valuable suggestions for handling imbalanced features, implementing those suggestions required careful adaptation to our dataset’s unique structure. Overall, ChatGPT acted as a complementary tool, enhancing the analytical and reporting aspects of the project.

References

UCSB Natural Language Processing Group. *LIAR: A Benchmark Dataset for Fake News Detection*. Hugging Face. Accessed December 6, 2024 https://huggingface.co/datasets/ucsbnlp/liar

Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. NAACL-HLT. https://arxiv.org/abs/1810.04805v2

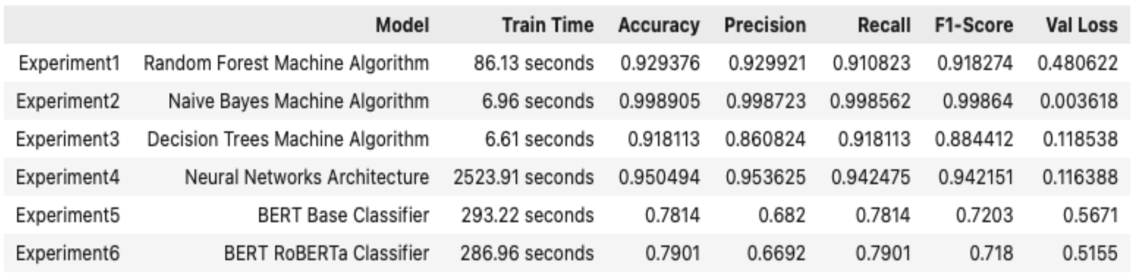
SindhuMadi. *Fake News Detection.* GitHub, June 11, 2021. https://github.com/SindhuMadi/FakeNewsDetection

Pham, Khang. "Text Classification with BERT." *Medium*, May 9, 2023. https://medium.com/@khang.pham.exxact/text-classification-with-bert-7afaacc5e49b

Appendix A- Accuracy Results from Algorithms

The table below displays the Train Time, Accuracy, Precision, Recall, F1-Score and Validation Loss of each of the models developed for Fake News Classification Dataset.

Result: Table with the accuracy, precision, recall and f1-score & process time for ALL the models.



	Model	Train Time	Accuracy	Precision	Recall	F1-Score	Val Loss
Experiment1	Random Forest Machine Algorithm	86.13 seconds	0.929376	0.929921	0.910823	0.918274	0.480622
Experiment2	Naive Bayes Machine Algorithm	6.96 seconds	0.998905	0.998723	0.998562	0.99864	0.003618
Experiment3	Decision Trees Machine Algorithm	6.61 seconds	0.918113	0.860824	0.918113	0.884412	0.118538
Experiment4	Neural Networks Architecture	2523.91 seconds	0.950494	0.953625	0.942475	0.942151	0.116388
Experiment5	BERT Base Classifier	293.22 seconds	0.7814	0.682	0.7814	0.7203	0.5671
Experiment6	BERT RoBERTa Classifier	286.96 seconds	0.7901	0.6692	0.7901	0.718	0.5155

Fig 1: Model Results

Appendix B –Algorithms Performance Plot

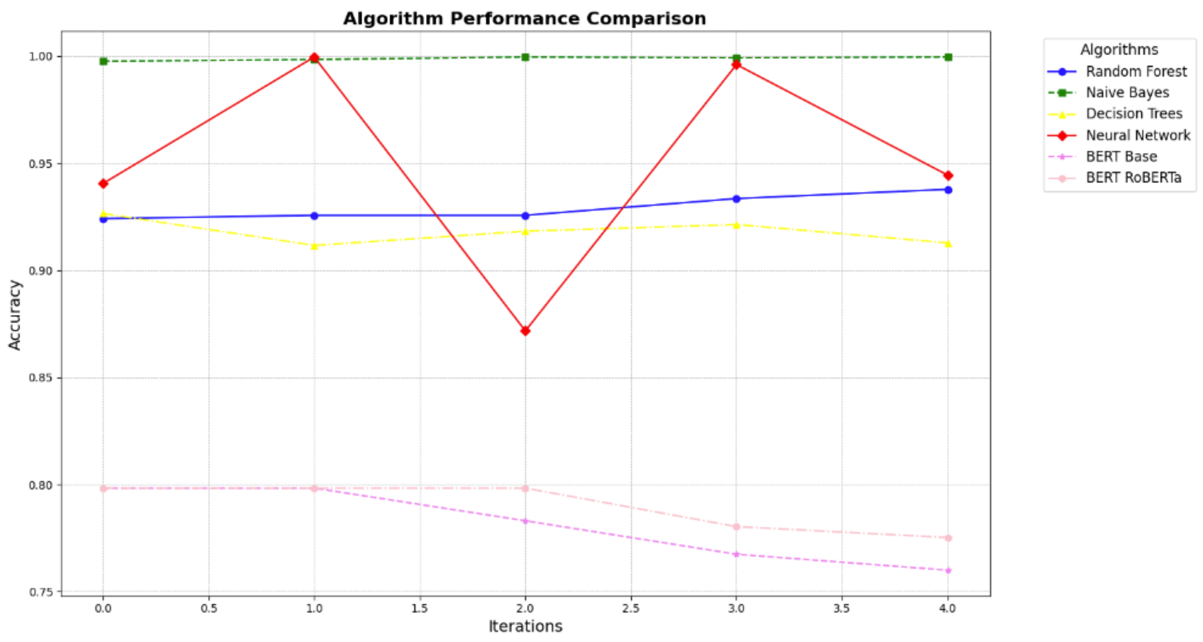


Fig 2: Model Performance

Appendix C – Confusion Matrices Resulting from Algorithms

The images below display the confusion matrices resulting from the application of each Fake News Classification model against the testing dataset.

Random Forest:

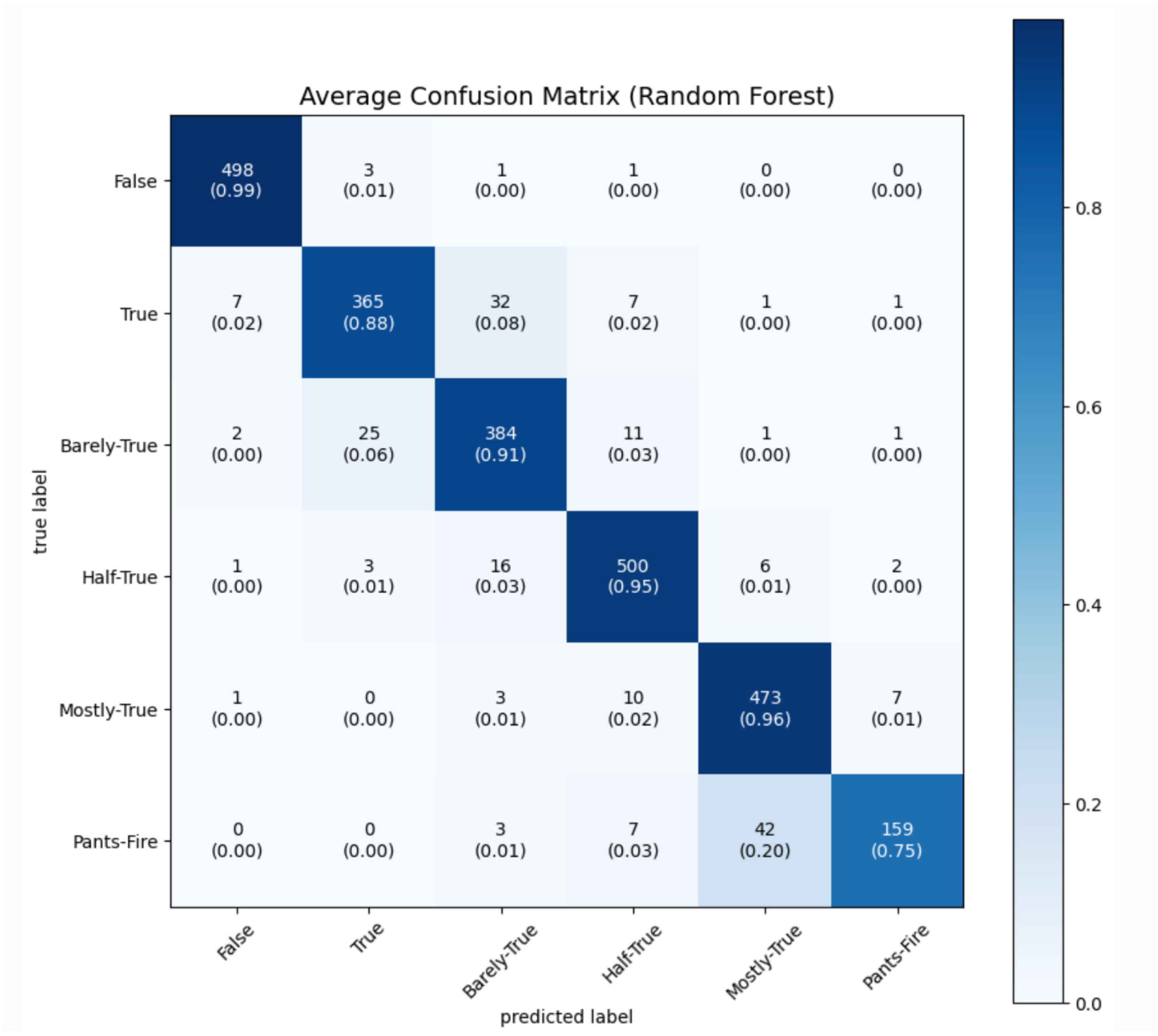


Fig 3: Confusion Matrix Random Forest

Naive Bayes

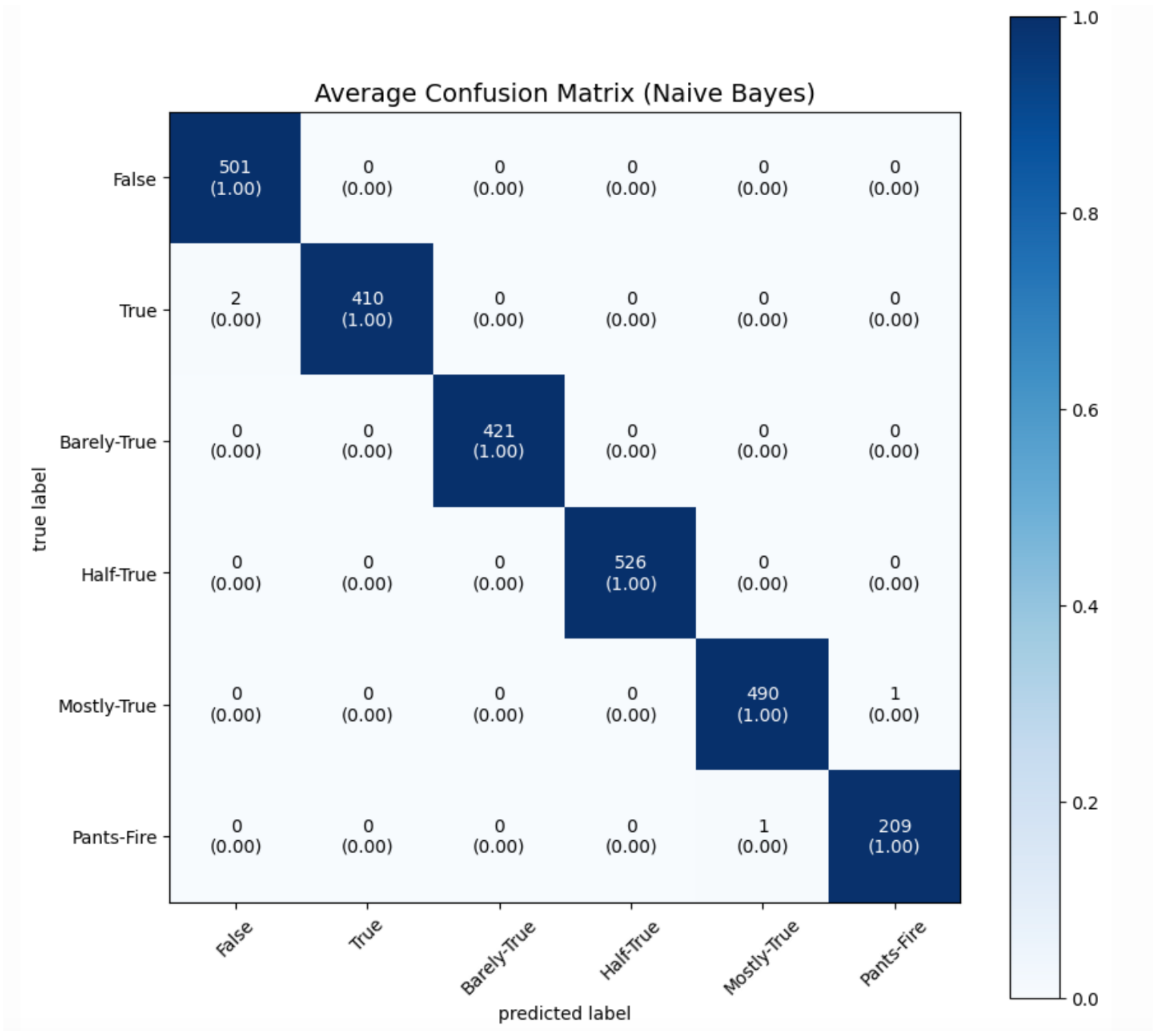


Fig 4: Confusion Matrix Naive Bayes

Decision Tree

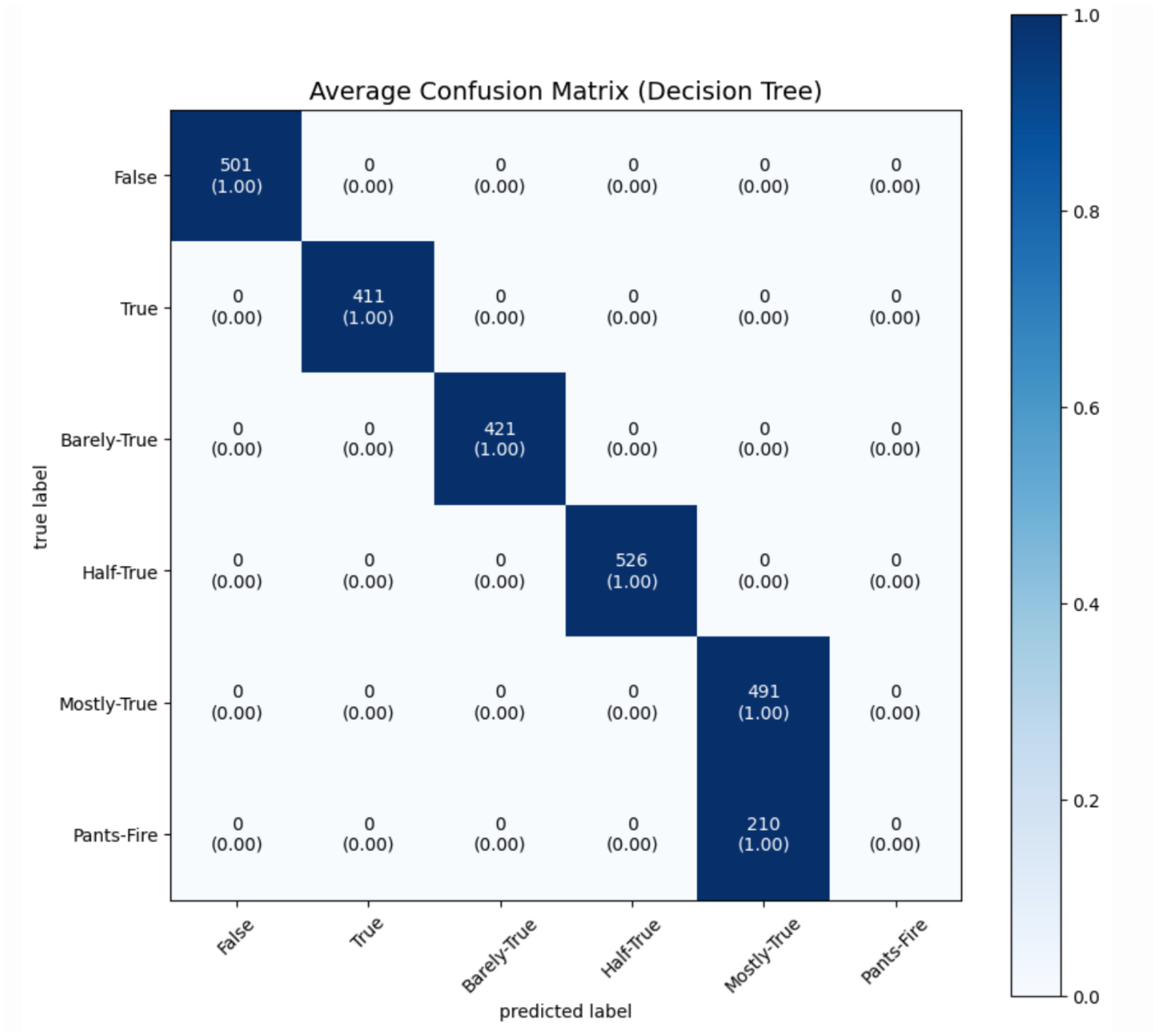


Fig 5: Confusion Matrix Decision Tree

Neural networks

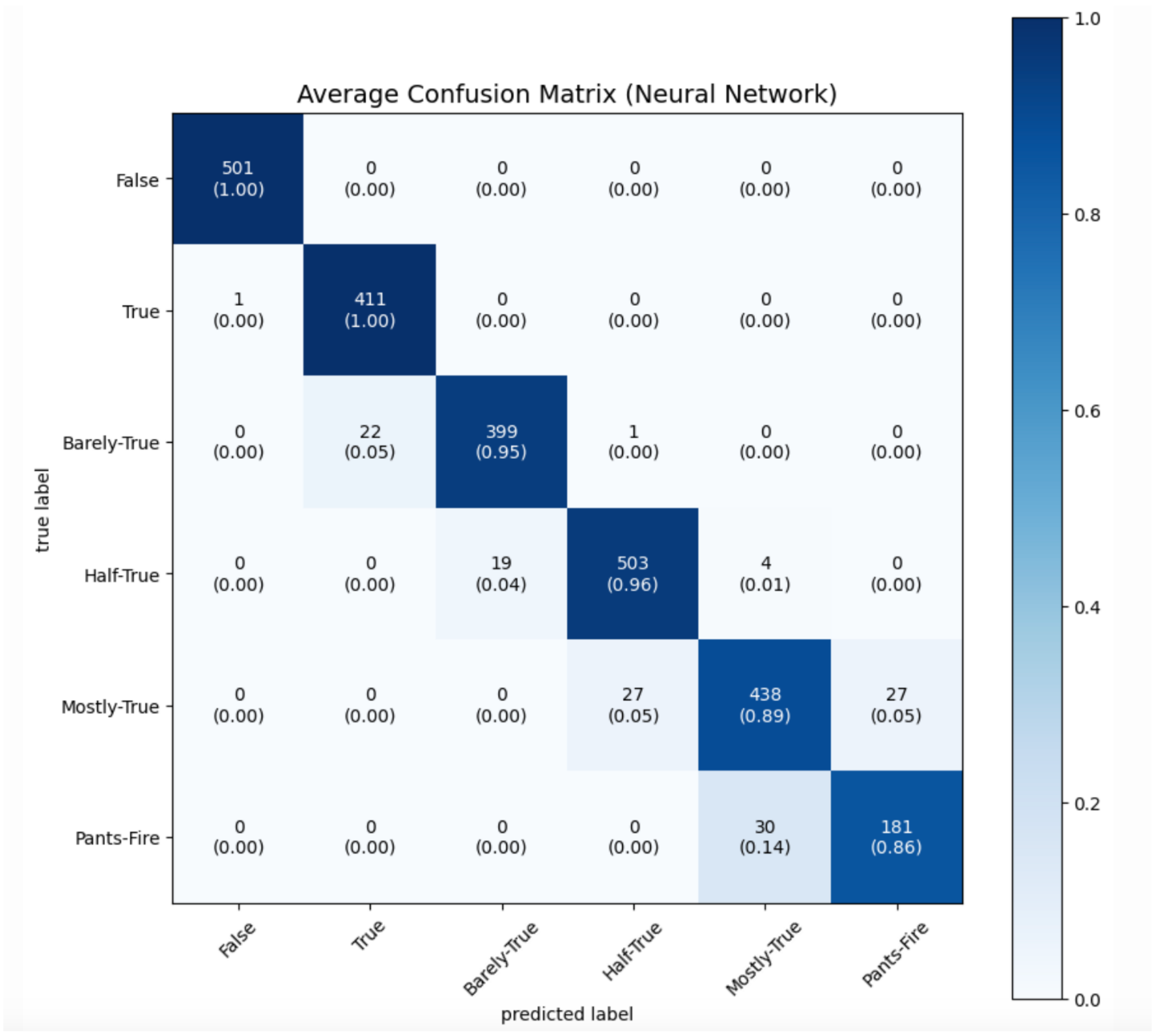


Fig 6: Confusion Matrix Neural Network

BERT Base

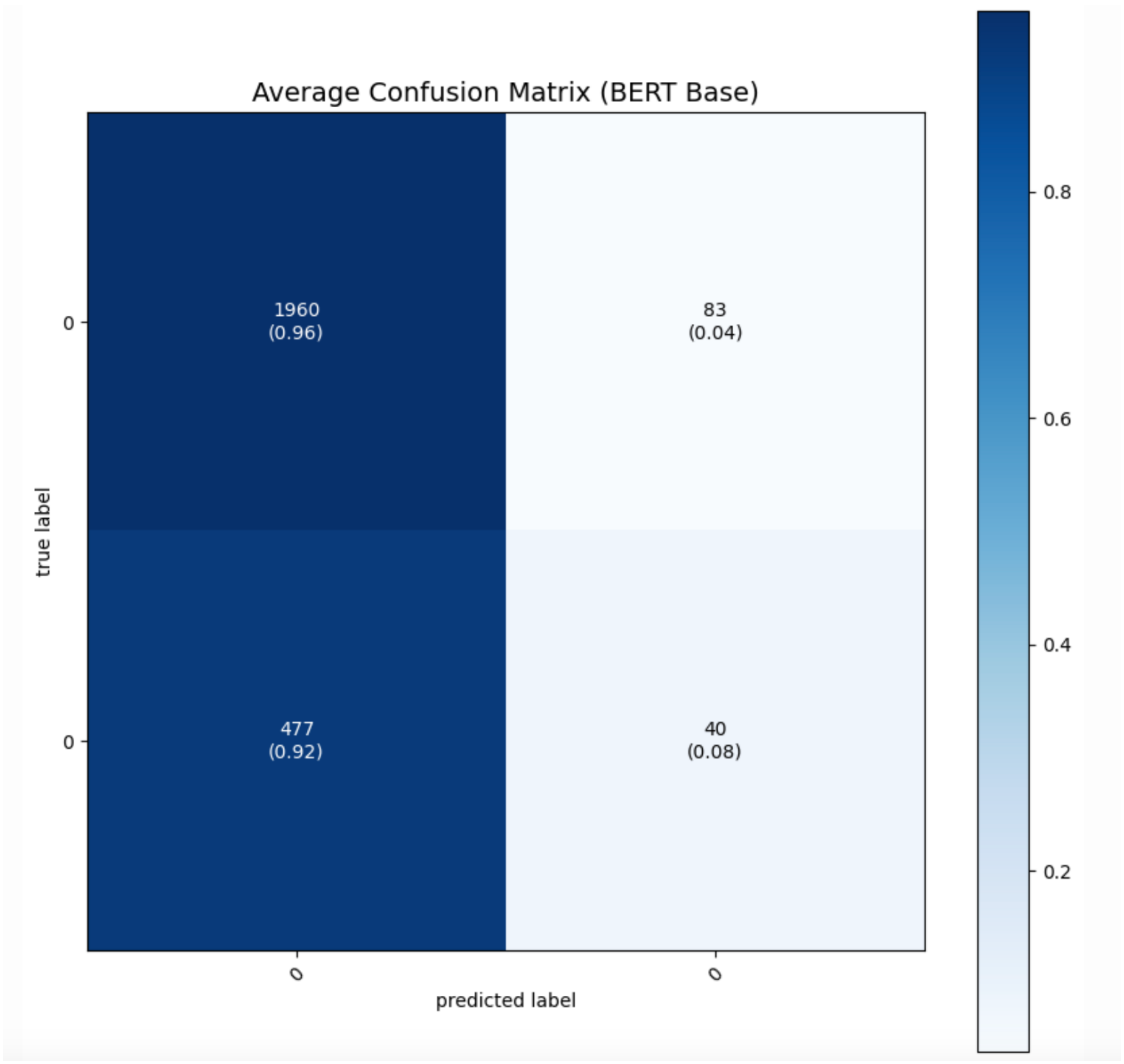


Fig 7: Confusion Matrix BERT Base

BERT RoBERTa

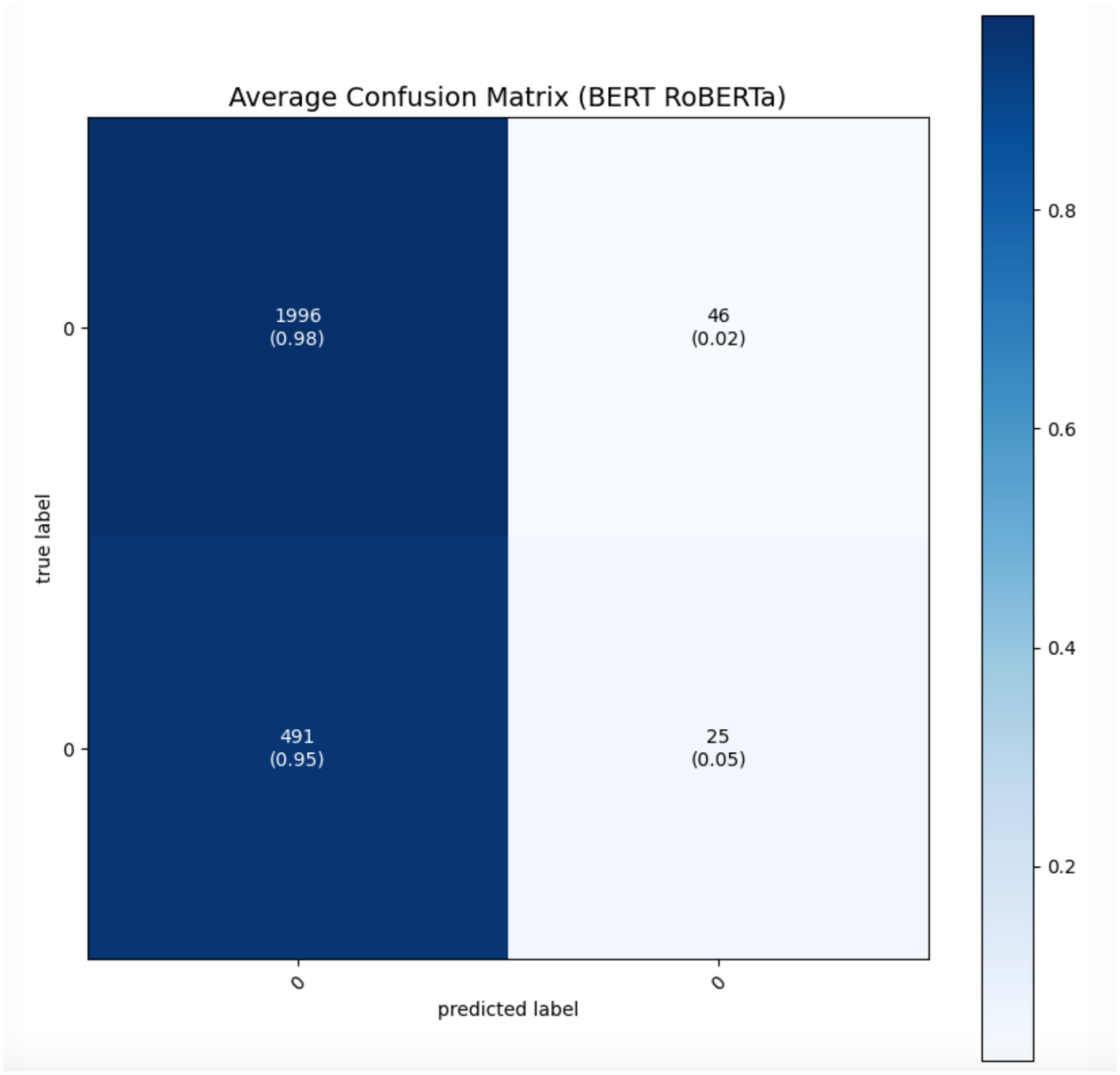


Fig 8: Confusion Matrix BERT RoBERTa