Fake News Detection - CS 5785 Final Project

Matthew Maitland, Sofia Beyerlein, Shreeya Indap
Cornell Tech
2 West Loop, New York NY
{mjm638, sb2669, si223}@cornell.edu

Abstract

This project explores machine learning approaches to classify news articles as fake or real using their titles and text. A baseline Logistic Regression was built with TF-IDF vectorization with the goal of comparing this method to more advanced techniques. The baseline model achieved an accuracy of 97.3%, while a fine-tuned DistilBERT model improved accuracy to 98.9%. Preprocessing techniques such as lemmatization, punctuation removal, and contraction handling enhanced data quality. Motivated by the desire to complete a project that is both technologically challenging as well as impactful, the results demonstrate the superior performance of transformer-based models.

1. Introduction

Fake news has become a significant challenge in the digital age, influencing public opinion, politics, and much more. This project aims to address this issue by leveraging machine learning techniques for the classification of news articles as real or fake.

The Fake News Classification Dataset, utilized in this study, contains over 40,000 unique English-language news articles, each labeled as true (1) or false (0). The dataset is split into 3 unique subsets, for training, validation, and testing. Train contains 24,353 unique instances, while test and validation each contain 8,117. All three subsets exhibit a similar distribution of True/False labels, with the proportion of 'True' instances ranging from 53.1% to 54.4%. These distributions are shown below, in Figure 1 and Figure 2.

Using this dataset, we developed a pipeline that incorporates both traditional machine learning methods and advanced deep learning approaches. Our primary objective is to compare the effectiveness these models in accurately classifying fake news articles.

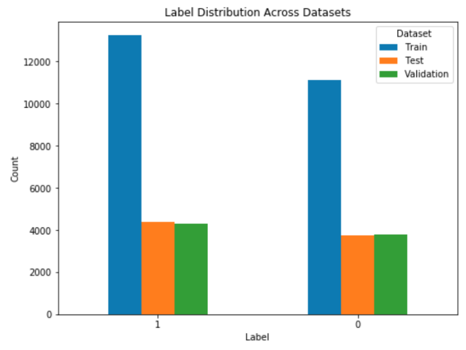

Figure 1. Bar chart showing the count of 'True' and 'False' labels across the training, validation, and test subsets.

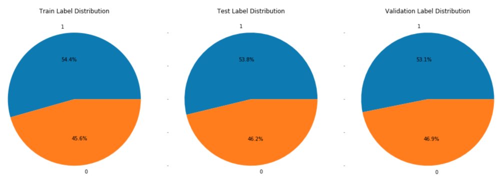

Figure 2. Pie chart showing the percentage distribution of 'True' and 'False' labels across the training, validation, and test subsets.

2. Related Works

Since this dataset is public on Kaggle, there are a few examples of others tackling the same task of fake news classification, many of which leveraged similar methods (including both Logistic Regression and Fine-Tuned Bert models). However, none of the 15 public notebooks were able to achieve a higher accuracy score on the test set than our deep learning approach. The highest score found on the public notebooks was an accuracy score of 98.46, which falls short of our score of 98.87.

Considering there were a variety of different approaches that fine tune a Bert model which came up short in comparison to our performance, it is clear that network architecture, as well as logical preprocessing steps are integral to building the most successful model.

3. Methods

3.1. Preprocessing

To prepare the dataset for model training and evaluation, we implemented a comprehensive text preprocessing pipeline. This pipeline aimed to standardize, clean, and enhance the quality of the textual data for improved performance in machine learning models. The following steps were performed:

1. **Lowercasing:** All text data was converted to lowercase to ensure the text was uniform and to mitigate case sensitivity issues.

2. **Removing Missing Data:** Any rows containing missing or null values in the text field were removed from the dataset to avoid processing incomplete data. However, in this specific dataset, there were no entries to remove.

3. **Lemmatization:** Each word in the text was lemmatized using the WordNet Lemmatizer. Lemmatization is the process of reducing words to their base form, helping to normalize variations of the same word. For example, the words ”running,” ”ran,” and ”runs” would all be reduced to their base form ”run,” ensuring consistency.

4. **Removing Punctuation:** All punctuation marks were removed from the text to focus on the core content of the articles.

5. **Expanding Contractions:** Common contractions (e.g., ”can't” to ”cannot”) were expanded using the ‘contractions‘ library to ensure consistency.

6. **Additional Cleaning:**

    - URLs and special HTML entities (e.g., ‘&;‘) were removed.
    - Special characters and numbers were stripped from the text.
    - Single-character words and extra whitespace were eliminated to refine the dataset further.

7. **Validation:** After preprocessing, the cleaned text was reviewed to ensure that it retained meaningful content and was free from issues introduced during cleaning.

This preprocessing pipeline was applied uniformly to the training, validation, and test datasets. The resulting clean text provided a foundation for feature extraction and model training.

3.2. Exploratory Data Analysis (EDA)

The main priority of EDA was to gain a sense of the distribution of our data so that we could choose an effective evaluation metric. Since our data was approximately 53% true across all datasets, we determined that the dataset was relatively balanced. Therefore, accuracy would serve as a reliable primary evaluation metric, as it would not be overly influenced by class imbalance. We also confirmed that each dataset (train, validation, and test) likely come from the same distribution, so no advanced re-distribution techniques were required.

To extract this information, we visualized the label distribution using a bar chart (Fig. 1) and pie charts (Fig. 2), which illustrated the proportions of true and false labels in each dataset. These visualizations confirmed that the class distributions were comparable across the subsets.

3.3. Baseline Logistic Regression

As a baseline model, we implemented a simple and interpretable Logistic Regression for classification. To prepare the textual data, we utilized the TF-IDF (Term Frequency-Inverse Document Frequency) vectorizer to convert the text into numerical features. This captures the importance of words in the context of the entire dataset, allowing the model to differentiate between frequently occurring yet uninformative words and more meaningful terms. The logistic regressor t then only leverages on the most salient words.

Steps:

1. **TF-IDF Vectorization:** We applied a TF-IDF vectorizer with a maximum of 5,000 features to transform the training text data into a numerical feature matrix. The same transformation was applied to the test data using the already fitted vectorizer to ensure consistency. This means that even when predicting on the validation and test set, only the top 5,000 most frequent words from the training set were considered.

2. **Model Training:** A Logistic Regression model was trained on the TF-IDF-transformed training dataset, with the binary label as the target variable. The model was initialized with a random seed of 42 for reproducibility.

3. **Evaluation:** The trained model was evaluated on the test dataset. Predictions were generated, and the model's performance was measured using metrics such as accuracy, precision, recall, and F1-score. These metrics provided a comprehensive understanding of the baseline model's classification performance.

3.4. Embedding and Tokenization

To use Transformers, we incorporated the DistilBERT model for text embedding. The embedding and tokenization process prepared the textual data for input into the neural network. DistilBERT, a lightweight version of BERT, provided contextual embeddings to capturing semantics in our data.

Steps:

1. **Loading the Pretrained Model:** We used the `DistilBertTokenizer` and `DistilBertModel`, both initialized with the pretrained `distilbert-base-uncased` model. This ensured that the embeddings contained knowledge learned from a large corpus of English text from Bert's pretraining.

2. **Tokenization and Padding:** Using the DistilBERT tokenizer. Each input was:

    - **Tokenized:** Split into subword tokens based on the tokenizer's vocabulary.
    - **Truncated:** Limited to a maximum sequence length of 128 tokens.
    - **Padded:** Shorter sequences were padded to ensure uniform input size.

3. **Output:** The tokenization process returned:

    - `input_ids`: Numeric representations of the tokenized text.
    - `attention_mask`: Binary masks indicating non-padded tokens (1) and padded tokens (0).

4. **Application to Datasets:** Tokenization was applied separately to the training, validation, and test datasets, ensuring consistent preparation across all subsets.

This process transformed raw text into a format suitable for input into the DistilBERT neural network.

3.5. Neural Network

The neural network was designed to process tokenized text inputs and classify news articles as either fake or real. It is comprised of two main components:

- **DistilBERT**:

  `distilbert-base-uncased` is a pre-trained language model that is highly effective when fine-tuned for specific tasks. By leveraging a fine-tuned model, we focus exclusively on the neural network components that are directly relevant to our classification task, as the model already has a comprehensive understanding of language semantics. As a result, this approach significantly reduces the complexity and training time while enhancing performance on domain-specific tasks like fake news detection.

- **Classification Head**: A fully connected layer is added on top of the DistilBERT embeddings. This layer reduces the 768-dimensional embeddings to a single output corresponding to the predicted label (0 for fake news, 1 for real news). This is the fine tuning step on top of `distilbert-base-uncased`, giving it additional information to learn about how to make our specific classification.

The neural network is implemented in a custom PyTorch class, `DistilBertForFakeNewsClassification`, which integrates the DistilBERT model with the classification head.

3.6. Training

To train our neural network, we utilized the AdamW optimizer with a learning rate of 2×10^{-5} and applied the binary cross-entropy loss function with logits (`BCEWithLogitsLoss`). The model was trained for 5 epochs with a batch size of 32, ensuring that the network could effectively learn while maintaining reasonable efficiency.

During each epoch, the model processed batches of tokenized input data, calculated loss, and updated its parameters using backpropagation. To monitor performance and prevent overfitting, validation accuracy was conducted at the end of every epoch. After this step, we predicted on the test data, and achieved a score of 98.67%.

The final training step was to combine our training and validation datasets, and re-run the the training loop. Once trained on all available training data, we achieved a final test accuracy of 98.87%.

4. Results

In this section, we present the performance of the models evaluated on the test dataset. The results are divided into three subsections: the baseline Logistic Regression model, the Neural Network (NN) trained on only the training data, and the NN trained on the combined training and validation datasets.

4.1. Logistic Regression Baseline

The Logistic Regression model, used as a baseline, was trained on the TF-IDF vectorized representations of the training data. The performance on the test dataset is summarized below:

- **Accuracy**: 97.26%

Confusion Matrix:

$$\begin{bmatrix} 3646 & 107 \\ 115 & 4249 \end{bmatrix}$$

Classification Report:

```
              precision    recall  f1-score   support
   Fake News     0.97      0.97      0.97      3753
   Real News     0.98      0.97      0.97      4364

    accuracy                         0.97      8117
   macro avg       0.97      0.97      0.97      8117
weighted avg       0.97      0.97      0.97      8117
```

4.2. Neural Network Trained on Training Data Only

The Neural Network model, fine-tuned on the training data, demonstrated strong performance. The training loss across 5 epochs and the validation accuracy are detailed below:

- **Training Loss (Epoch 1–5)**: [0.0650, 0.0280, 0.0144, 0.0077, 0.0036]
- **Validation Accuracy (Epoch 1–5)**: [98.82%, 98.80%, 98.78%, 98.82%, 98.69%]
- **Test Accuracy**: 98.67%

Confusion Matrix:

$$\begin{bmatrix} 3693 & 60 \\ 48 & 4316 \end{bmatrix}$$

Classification Report:

```
              precision    recall  f1-score   support
   Fake News     0.99      0.98      0.99      3753
   Real News     0.99      0.99      0.99      4364

    accuracy                         0.99      8117
   macro avg       0.99      0.99      0.99      8117
weighted avg       0.99      0.99      0.99      8117
```

4.3. Neural Network Trained on Combined Training and Validation Data

The final Neural Network model was trained on the combined training and validation datasets for additional learning opportunities. The training loss across 5 epochs is summarized below:

- **Training Loss (Epoch 1–5)**: [0.0016, 0.0019, 0.0025, 0.0008, 0.0017]
- **Final Test Accuracy**: 98.87%

Confusion Matrix:

$$\begin{bmatrix} 3707 & 46 \\ 46 & 4318 \end{bmatrix}$$

Classification Report:

```
              precision    recall  f1-score   support
   Fake News     0.99      0.99      0.99      3753
   Real News     0.99      0.99      0.99      4364

    accuracy                         0.99      8117
   macro avg       0.99      0.99      0.99      8117
weighted avg       0.99      0.99      0.99      8117
```

5. Discussion

Our results highlight the effectiveness of pre-trained language models like DistilBERT for fake news classification. A clear progression in performance is observed from the Logistic Regression model to the fine-tuned Neural Network, particularly when additional data is incorporated.

5.1. Baseline Model Performance

The Logistic Regression baseline achieved an accuracy of 97.26%, demonstrating the strength of simple linear models when combined with feature extraction techniques like TF-IDF and valid, concise, preprocessing. However, its performance plateaued due to the limitations of sparse feature representations. TF-IDF lacks the ability to model word locality and contextual relationships, which likely led to the slightly higher misclassification rates compared to the Neural Network.

5.2. Neural Network with Training Data Only

The Neural Network trained on the training dataset alone outperformed the Logistic Regression model, achieving a test accuracy of 98.67%. This significant improvement is likely due to DistillBERT embeddings, which are able to capture semantic relationships and contextual nuances within sentences, unlike TF-IDF. During training, we saw that the model loss reduced steadily from 0.065 to 0.0036, indicating that it was actively learning in each epoch. However, the slight drop in validation accuracy in the later epochs suggests that it may not need all 5 epochs.

5.3. Impact of Combining Training and Validation Data

The final model, trained on the combined training and validation datasets, achieved a test accuracy of 98.87%, a slight improvement. This demonstrates the value of incorporating more data during fine-tuning to allow the model to better generalize its learned patterns. However, the minimal increase in performance might suggest that the model had already captured most of the relevant patterns, and additional data is no longer required. In this version, we observed that the loss started extremely low and oscillated very close to zero throughout training. While this behavior could be due to random initialization, it may also indicate that the model was able to capture the required patterns in fewer epochs than the version trained with less data.

5.4. Limitations

Although pleased with our final result, especially since we outperformed other fine-tuned LLMs on the same dataset, there are a few areas where we might have been able to improve. Firstly, our entries were truncated to a maximum of 128 tokens. Although our texts are fairly short, there were a significant number of entries that were truncated. If we had more time and computational resources, it might have been beneficial to allow for longer inputs. Additionally, we did not do extensive hyperparamater tuning due to computational constraints. It is possible that additional tuning could have yielded even better results. Finally, although are dataset was fairly balanced, it did have a very slight skew towards the True class. We felt this was too small to account for, but it is possible that it slightly affected our results.

6. Conclusion

This project compared Logistic Regression with a fine-tuned DistilBERT model for fake news classification. The DistilBERT model outperformed the baseline, achieving 98.87% accuracy, demonstrating the effectiveness of pre-trained transformers. While incorporating additional data improved performance slightly, our results suggest the model captured most of the relevant patterns without the additional data. Overall, this project highlights the value of advanced deep learning methods for combating misinformation, or any general text classification or analysis tasks.

7. References

References

[1] Aadya Singh, *Fake News Classification*, Retrieved from `https://www.kaggle.com/datasets/aadyasingh55/fake-news-classification/data`, 2024.