Detecting Fake News By Machine Learning Algorithm

Harry Zhang Jingzhen Wang Nicholas Li

Abstract

Fake news, characterized by the deliberate dissemination of false or misleading information, poses significant societal challenges, particularly with the rapid spread of misinformation on social media platforms. This study focuses on developing and evaluating machine learning and deep learning methods to detect fake news based on text content, source, and authorship. Using datasets from the Kaggle Fake News Competition and the LIAR dataset, we preprocess and analyze textual data, employing logistic regression, softmax regression, large language models (LLMs) such as GPT-4o-mini, and Bidirectional Encoder Representations from Transformers (BERT) for classification. Pre-trained word embeddings, including Word2Vec, are utilized to represent textual data in vectorized formats for training and evaluation. We assess model performance using accuracy, F1 score, and confusion matrices, highlighting strengths and weaknesses of each method. Results demonstrate the efficacy of neural network-based approaches, particularly hybrid models like BERT, in capturing nuanced patterns in fake news. This work contributes to the growing field of automated fake news detection by integrating traditional machine learning methods with advanced language models, paving the way for more robust solutions to combat misinformation.

1. Introduction

Fake News refers to news reports or information that primarily contain false or misleading content, deliberately spreading incorrect information. Its purpose is to influence public opinion, mislead readers, and even trigger social or political unrest. Fake news can appear in various forms, including fabricated news reports, exaggeration or distortion of facts, false headlines, or mixing incorrect information with true content. The growth of social media has exacerbated the spread of fake news. Thus, in the internet age, it has become particularly important for social media and news organizations to identify and promptly block the spread of false information.

Fine-definition of Problem

This study focuses on classifying fake news depending on the message itself, the source of the message, and the author of the message. Usually, fake news has a variance of format, such as videos, broadcasts, posts, online contents, messages, etc. In this project, we will focus solely on the text formats. Often, the detection of fake news relies on checking the source information. However, this requires a large amount of human power and time. Thus, we strive to find methods that can detect the fake news on the message itself and the source of the message.

In our approach, we were able to use a hybrid neural network approach using BERT to achieve 97.89% accuracy. We also tried several other less complex approaches, as well as using GPT-4o mini as a baseline, which were less effective. This shows that fake news classification is a complex problem, with no simple way to determine whether a particular news article is authentic or not.

2. Related Work

The problem of fake news detection has garnered significant attention in recent years, driven by the surge in misinformation spread through digital platforms. Several key works have laid the groundwork for this field, each contributing unique datasets, methodologies, and analyses.

One of the foundational datasets for fake news detection is the LIAR dataset, introduced by Wang in the paper *"Liar, Liar Pants on Fire: A New Benchmark Dataset for Fake News Detection"* (Wang, 2017). This dataset consists of 12.8k news items, each annotated with features such as the speaker, context, and political affiliation, with truthfulness manually labeled. However, the dataset uses only short word snippets, and so results are heavily based on the context of the words as well. Our model aims to achieve a high accuracy using just the text itself.

A comprehensive review of fake news detection techniques is provided by Shu et al. in their work *"Fake News Detection on Social Media: A Data Mining Perspective"* (Shu et al., 2017). This paper presents a wide-ranging survey of fake news characterization, highlighting psychological and social theory perspectives. It explores data mining techniques used for detection, the key evaluation metrics, and relevant

datasets, offering a crucial overview of the landscape of fake news detection from a social media standpoint.

Recent advances in deep learning have pushed the boundaries of fake news detection accuracy. The work by Kadek Sastrawan., *"Detection of Fake News Using Deep Learning CNN–RNN Based Methods"* (Sastrawan et al., 2022), demonstrates the application of deep learning architectures such as CNN, Bidirectional LSTM, and ResNet in conjunction with pre-trained word embeddings. Their findings show that Bidirectional LSTM consistently outperformed CNN and ResNet across multiple datasets, underscoring the potential of hybrid neural network architectures for capturing nuanced patterns in fake news content. Our results echo these findings, with a BERT hybrid model producing the most accurate classification.

In addition, Hu et al.'s paper, *"Deep Learning for Fake News Detection: A Comprehensive Survey"* (Hu et al., 2022), provides a systematic review of deep learning methods focused on content-based, social context, and external knowledge features. The survey categorizes these approaches into supervised, weakly supervised, and unsupervised methods, offering a critical analysis of their effectiveness and the challenges associated with each. Notably, the survey noted that supervised methods were limited by the difficulty of attaining accurately labeled datasets, as doing so "is often complex, time-consuming, costly to procure, or unavailable due to privacy or data access constraints," as well as varying over time due to the rapidly changing nature of news.

Finally, the Kaggle Fake News Challenge provides an annual platform for benchmarking fake news detection algorithms. Participants have achieved accuracies of up to 95% using techniques such as GloVe and LSTM, reflecting the effectiveness of state-of-the-art machine learning models in this domain. The challenge highlights the potential for innovation and the continued evolution of methodologies in detecting fake news. We used multiple metrics in order to test our model, using accuracy, and additional heuristics like an F1 score and confusion matrix to evaluate how successful our approach and parameters were. In our project, we also used the Fake News Competition dataset (Lifferth, 2018), which we found to be a suitable source for training relative to the current state of the art.

Together, these works form the foundation upon which our project will build, integrating proven methods while exploring new directions in fake news detection.

3. Dataset and Evaluation

Dataset Section

In this project, we applied the data set used in Kaggle Fake News Competition. https://www.kaggle.com/competitions/fake-news/overview

They are provided with two main datasets: `train.csv` and `test.csv`, which contain news articles with various attributes. These datasets are used to train and evaluate the machine learning model for predicting whether a news article is reliable or potentially unreliable.

A. Dataset Source and Description

- `train.csv`: This dataset is used for training the model. It contains 20,827 news articles with five attributes:
  - `id`: A unique identifier for each news article.
  - `title`: The title of the news article.
  - `author`: The author of the article (this field may be missing for some articles).
  - `text`: The body of the article (this field may be incomplete in some cases).
  - `label`: A binary label that indicates whether the article is potentially unreliable (1) or reliable (0).
- `test.csv`: This dataset contains similar attributes as the training dataset but lacks the `label` column. It will be used to make predictions, and the model's performance will be evaluated based on its ability to classify the articles in this dataset correctly.
  - `id`: A unique identifier for each news article.
  - `title`: The title of the news article.
  - `author`: The author of the article.
  - `text`: The body of the article.
- `submit.csv`: This is a sample submission file that demonstrates the expected format for the submission. It contains two columns:
  - `id`: The unique identifier of each article from `test.csv`.
  - `label`: The predicted label indicating whether the article is potentially unreliable (1) or reliable (0).

  We also tried the LIAR dataset https://github.com/tfs4/liar_dataset/blob/master/ which provides labeled text snippets, labeled into 6 levels of truthfulness. This allows us to try multiple-classification tasks, where we used a softmax regression. The dataset consists of two main files: `train.csv` and `test.csv`.
  - `train.csv`: This file contains 10,240 labeled pieces of text used to train the model. Each article has the following attributes:
    - `id`: A unique identifier for each news article.

    - * `truthfulness`: pants-fire, FALSE, barely-true, half-true, mostly-true, or TRUE
    - * `text`: A short snippet of text
    - * `topic`: What the text is in relation to
    - * `author`: Who wrote the text
    - * `occupation`: What the author does
    - * `state`: from what state they're from
    - * `affiliation`: What group or political party they're associated with
    - * `truthfulness details`: Specific labels used to determine truthfulness
    - * `context`: What this text is from
  - `test.csv`: This file is similar to `train.csv`, but it only contains 1,267 pieces of text

B. Preprocessing

We performed the following preprocessing steps on the dataset:

- **Missing Values**: Some of the news articles are missing the `author` or `text` attributes. These missing values were handled by filling them with placeholders or removing them if the article was completely empty. This ensures that incomplete records do not interfere with model training and prediction.
- **Text Preprocessing**: To standardize and enhance the quality of input text, we applied several preprocessing techniques:
  - Lowercasing: All text was converted to lowercase for consistency.
  - Punctuation Removal: We removed punctuation to prevent model confusion between different forms of the same word.
  - Stopword Removal: Common words that don't add significant meaning (e.g., "the", "and") were removed.
  - Tokenization and Vectorization: The processed text data was transformed into numerical features using techniques such as TF-IDF and word embeddings, making it suitable for model training.
- **Data Splitting**: The `train.csv` file was split into three sets using a 7:1:2 ratio, resulting in 70% for the training set, 10% for the development set (dev set), and 20% for the test set. This split allowed us to train the model on a substantial portion of the data (70%) while reserving 10% for hyperparameter tuning and model validation, and the final 20% for unbiased testing.
  - Training Set: Approximately 14,579 examples, with a balanced distribution of reliable and unreliable articles. This set was used for training the model.
  - Development Set: Approximately 2,083 examples, also balanced between the two classes. This set was used to tune hyperparameters and evaluate the model's performance during development.
  - Test Set: Approximately 4,165 examples, balanced as well, which served as an independent set for final model evaluation.

C. Data Augmentation

No data augmentation techniques were applied, as the dataset contains textual data, which is not typically augmented in the same manner as image data.

Evaluation Section

A. Accuracy

The evaluation for this project focuses on **accuracy** as the primary metric. Accuracy is defined as the ratio of correct predictions to the total number of predictions made by the model. The formula for accuracy is given by:

$$\text{Accuracy} = \frac{\text{Correct Predictions}}{\text{Correct Predictions} + \text{Incorrect Predictions}}$$

Accuracy is a straightforward measure of overall performance, showing the proportion of articles the model correctly classified out of all predictions. Since this is a binary classification task with a balanced dataset (equal or similar numbers of reliable and unreliable articles), accuracy gives a good general indicator of how well each method performs.

While accuracy provides a broad overview, it doesn't capture nuances in model performance for each class (reliable vs. unreliable). For instance, if the model performs well on one class but poorly on the other, accuracy alone won't reveal this imbalance.

B. F1 score

The F1 Score is the harmonic mean of precision and recall. It balances the trade-off between these two measures:

- Precision (for the "unreliable" class) measures how many articles labeled as "unreliable" are actually unreliable
- Recall (for the "unreliable" class) measures how many of the actual unreliable articles were correctly identified by the model.

The F1 Score is particularly useful in cases where we want to ensure that both precision and recall are high, especially

for the "unreliable" class. In real-world applications, correctly identifying unreliable news is critical, as misclassifying unreliable articles as reliable could lead to the spread of misinformation. The F1 Score helps us evaluate the model's performance in distinguishing these crucial cases more effectively.

However the F1 score doesn't provide specific insights into each one individually. If the model has high precision but low recall (or vice versa), the F1 Score might obscure this detail.

C. Confusion matrix

A Confusion Matrix is a table that displays the true positives, true negatives, false positives, and false negatives in the model's predictions.

The Confusion Matrix offers detailed insight into the types of errors the model makes, which is essential for understanding its strengths and weaknesses. By examining each cell in the matrix, we can see:

- True Positives (unreliable articles correctly classified as unreliable).
- True Negatives (reliable articles correctly classified as reliable).
- False Positives (reliable articles incorrectly classified as unreliable)
- False Negatives (unreliable articles incorrectly classified as reliable).

This breakdown helps us understand specific areas where the model may need improvement, such as reducing false negatives to avoid letting unreliable news be classified as reliable.

While the Confusion Matrix gives detailed insights, it does not summarize model performance into a single score, which can make it harder to interpret without additional metrics.

4. Baselines and Methods

We evaluated four different methods for classifying news articles as reliable or unreliable. These methods include both baseline and machine learning techniques, as well as a language model-based approach. Below, we describe each method in detail:

4.1. Method 1: Applied LLM Model for classification (Baseline method)

For the final method, we leveraged a Large Language Model (LLM), specifically the GPT-4o-mini model from OpenAI, to classify news articles based on their title, author, and content. This method involved crafting a prompt for each article and submitting it to the LLM, which returned a label of either "reliable" or "unreliable." Here's a step-by-step outline of this approach:

- Input Preparation: The text input for each article was created by combining its title, author, and up to the first 1000 characters of the text field to fit within the LLM's token limits.
- Prompt Engineering: A standardized prompt was used to instruct the LLM to classify each article. The prompt asked the model to determine the article's reliability and output either "reliable" or "unreliable."
- Model Parameters: We used GPT-4o-mini without additional fine-tuning. The prompt design was adjusted iteratively to improve model interpretation of the task. No hyperparameters, such as temperature or maximum token length, were modified due to the straightforward nature of the classification task.

This method is expected to benefit from the LLM's contextual understanding of language, allowing it to capture subtleties in the text that traditional ML models might miss.

4.2. Method 2: Logistic Regression

For logistic regression, we also used the dataset from the Kaggle Fake News Competition. First, To transform the news text into a format suitable for logistic regression, each news piece was first processed by tokenizing, lowercasing, and removing stop words using the Natural Language Toolkit (NLTK). We then represented each word with pre-trained Word2Vec embeddings from the Google News dataset. For each paragraph of news, an average vector was calculated by averaging the embeddings of all words in the paragraph, resulting in a 300-dimensional vector representing the entire piece of news. The length of the paragraphs ranged from a few hundred words to 1500 words. This 300-dimensional vector served as the input feature for each news item. The transformed vectors were then split into training and testing sets, with 80% used for training and 20% for testing. We trained a logistic regression model on the training set to classify each piece of news as either "reliable" (0) or "unreliable" (1), based on the binary labels provided in the dataset. Model performance was evaluated on the test set using accuracy and a confusion matrix to assess classification accuracy.

4.3. Method 3: Softmax Regression

For softmax regression, we used the LIAR dataset, which split the data into 6 labels: pants-fire (worst), FALSE, barely-true, half-true, mostly-true, and TRUE (best). Similar to

the logistic regression approach, we first processed each piece of data (which was a short news piece, similar to twitter tweets) by tokenizing, lowercasing, and removing stop words with NLTK. Then, we represented each word using the pre-trained Word2Vec embeddings from the Google News dataset. For each input "tweet", we created an average vector by averaging the embeddings of all words in the paragraph, resulting in 300-dimensional vector for the piece of text.

We split the transformed vector into "training" and "testing" sets. Approximately 90% was training data, and 10% was testing data. We trained the softmax regression model on the training set, classifying each text input as one of the 6 categories. Model performance was evaluated based on % correct (accuracy), and loose % correct, allowing for 1 level of incorrectness. Then, a confusion matrix is used to assess classification accuracy.

In our experimentation, we decided on 0.1 as a learning rate, as it maximized accuracy without taking an excessive amount of time to run. We found that more iterations generally led to better results, without significant overfitting. However, the model was unable to achieve high accuracy with the training data, likely because it was only trained on the raw text and couldn't decipher complexities the same way a neural network could.

4.4. Method 4: Transformer(BERT) on Binary Dataset

For the final method, we implemented BERT (Bidirectional Encoder Representations from Transformers), a state-of-the-art transformer model, for classifying news articles. Thanks to the BERT developed by Google and an open source notebook code, which goes through BERT, developed by jerifate.(Devlin et al., 2019) (Jerifate, 2021)

Steps

PREPROCESSING

- Combined the `author` and `text` fields into a single text input.
- Tokenized the input using BERT's tokenizer, truncating or padding sequences to a maximum length of 512 tokens.

FEATURE ENGINEERING

- Applied BERT embeddings to transform the tokenized input into contextualized representations.
- Used a classification head (fully connected layer) on top of the BERT output to produce binary classifications.

HYPERPARAMETERS

- We experimented with batch sizes of 16, 32, and 64. We also tried training with 1 to 30 epochs. The combination that yielded the best result was a batch size of 32 and 10 epochs.
- We tried dropout level of 0.1, 0.2, and 0.5. And the best dropout level is 0.2.

The result of hyperparameters test is on the experiment part.

TRAINING SETUP

- **Dataset split:** 70% training, 10% validation, 20% testing.
- **Optimizer:** AdamW with a learning rate of 2×10^{-5}.
- **Batch size:** 32.
- **Number of epochs:** 10 (chosen based on validation performance).
- Used early stopping to prevent overfitting.
- Use Dropout probability of 0.2

EVALUATION

- **Metrics:** Accuracy, F1 Score, and a confusion matrix.
- Compared performance across training, validation, and test sets.

Advantages of BERT Over previous methods

BERT demonstrates exceptional contextual understanding by capturing bidirectional dependencies in text, enabling it to effectively classify nuanced content with a higher degree of precision. Unlike traditional models that rely heavily on manual feature engineering, BERT automates the process by generating rich, contextualized embeddings that are inherently more informative and accurate. This capability eliminates the need for labor-intensive preprocessing while delivering superior results. Additionally, BERT addresses the limitations observed in earlier approaches, such as logistic regression and large language models (LLMs). By leveraging fine-tuning techniques and its advanced embedding framework, BERT significantly outperforms these methods, showcasing its ability to adapt and excel in complex classification tasks.

4.5. Method 5: Transformer(BERT) on Multi-Category Dataset

We also experimented with using BERT on the LIAR dataset. LIAR has a total of 12 features: news text, topic of the news,

author's name, occupation, state, affiliation, context of the news, and 5 other features representing truthfulness of other text that came from the same author. The first 7 features are all in strings, and the last 5 features are integers.

Steps

PREPROCESSING

- Search for any feature that are null, and replace them with a empty string.
- Concatenate the first 7 features that are string, to form a single text input.
- The other 5 features are separate stored, and will be used as input for an additional layer in the next step.
- Label mapping, where the labels {true, mostly-true, half-true, barely-true, false, pants-fire} are mapped to numbers. Here we experimented with different mapping method, and determine that mapping to {1, 0.8, 0.6, 0.4, 0.2, 0} respectively produced the best result.

MODELING

- Loads the pre-trained BERT model using TFBertModel from the transformers library. This model is used to extract embeddings (features) from text.
- Loads the tokenizer, which converts raw text into input tokens compatible with the BERT model.
- Defines the bert encode function, which tokenizes each input text, truncate the text to a fixed maximum length of 60, and it returns two arrays: input_ids which is the encoded token IDs for each word, and attention_masks indicating which tokens are padding (0) and which are valid (1).
- Construct two input layers for the input_ids and attention_masks. Also construct an additional_features layer, which takes in the 5 integer features.
- We experimented with two different model design, one with input_ids and attention_masks only, and the other with input_ids, attention_masks, and additional-feature. Both of these are eventually passed into the pre-trained BERT model.

HYPERPARAMETERS

- We experimented with batch sizes of 16, 32, and 64. We also tried training with 10, 20, and 30 epochs. The combination that yielded the best result was a batch size of 32 and 10 epochs.
- We tried different combinations of relu and sigmoid for our layer design.

TRAINING SETUP

- dataset split: 70% training, 10% validation, 20% testing.
- Optimizer AdamW with a learning rate of $1 \times 10^{-}5$
- Batch size: 32
- Number of epochs: 10 (chosen based on validation performance)
- Dropout probability of 0.2

5. Experiments

The results from each method were evaluated on a test set. Below is a table summarizing the performance metrics for each approach:

Method	Accuracy	F1 Score
LLM (GPT-4 mini)	70%	0.6999
Logistic Regression	85.29%	0.86
Softmax Regression	23.99%	0.307 (max of all labels)
BERT on Binary Dataset	98%	0.98
BERT on Multi-Category Dataset	0.33%	0.39

Table 1. Accuracy and F1 score for each model

5.1. Method 1: Applied LLM Model for classification (Baseline method)

For the LLM-based method, the model achieved an accuracy of 70% and an F1 score of 0.6999. The confusion matrix shows that the model correctly identified 18 unreliable and 17 reliable articles, while misclassifying 7 unreliable and 8 reliable ones. This indicates that while the LLM captures some nuances, there is room for improvement, particularly in distinguishing between subtly misleading. Below we attached the confusion matrix for LLM-baseline method.

5.2. Method 2: Logistic Regression

For the logistic regression method, the model achieved an accuracy of 85.29% and an F1 score of 0.86. The confusion matrix shows that the model correctly identified the vast majority of data, while misclassifying a small proportion of the test set. Below we attached the confusion matrix for logistic regression method.

5.3. Method 3: Softmax Regression

The softmax regression produced results that are much better than random guessing–but still below acceptable accuracy,

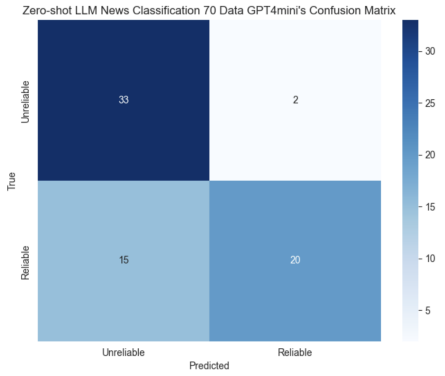

Figure 1. Confusion matrix for the LLM baseline method

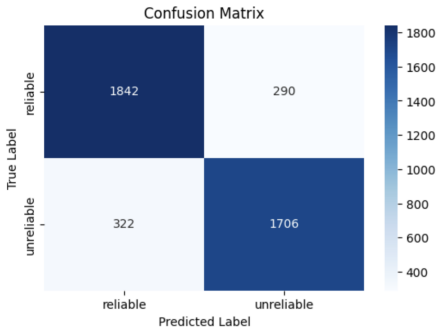

Figure 2. Confusion matrix for the logistic regression method

even when allowing for a 1 class offset in "trueness". The softmax regression is able to capture basic patterns, but is unable to accurately model based on intricacies in the inputs. This is most likely because there are many features in the LIAR dataset that we have not included in the learning algorithm yet, such the the political party affiliation, subject, and location of speech, etc. Later when we will incorporate these information as inputs for the algorithm. The confusion matrix we have for softmax regression is shown below.

5.4. Method 4: Transformer(BERT) on Binary Dataset

HYPERPARAMETERS

We tried dropout levels of 0.1, 0.2, and 0.5. The 0.2 performance best in our trial.

Drop-out Level	Dev Accuracy
0.1	93.99%
0.2	96.10%
0.5	93.79%

Table 2. Relationship Between Dropout Level and Dev Accuracy

For epochs, the dev loss reaches the deepest point at the 12th epoch. And value from 10 to 13 are very close. We

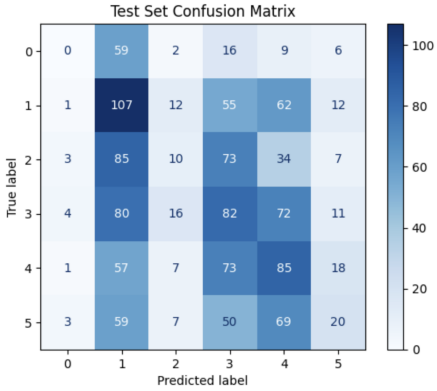

Figure 3. Confusion matrix for the softmax regression method

choose to use 10 epochs to get better training time.

For batch size, we tested batch size = 16, 32, and 64. The 32 has the best dev acc.

Batch Size	Dev Accuracy
16	94.89%
32	97.30%
64	95.80%

Table 3. Relationship Between Dropout Level and Dev Accuracy

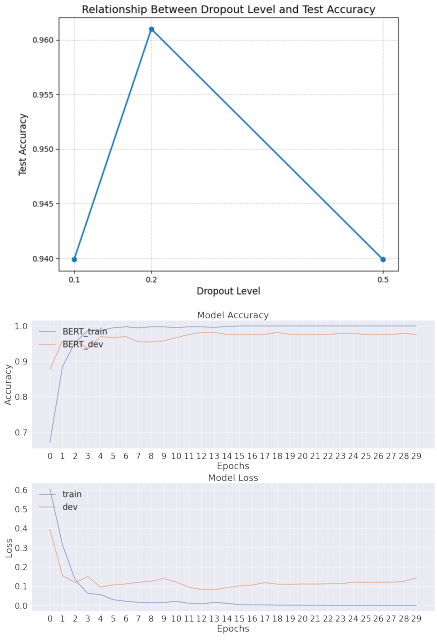

Figure 4. hyper parameters' graph

RESULTS

For the BERT-based method, the model achieved an accuracy of 98% and an F1 score of 0.98. The confusion matrix illustrates that the model performed exceptionally well, correctly classifying the vast majority of the test set. Specifically, it accurately identified 2124 reliable news articles and 1952 unreliable news articles.

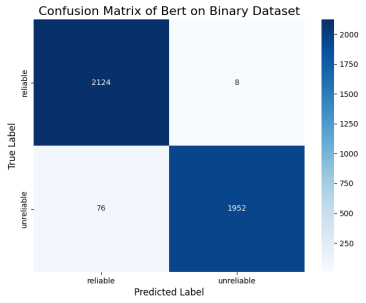

Figure 5. Confusion matrix for the BERT method on Binary Dataset

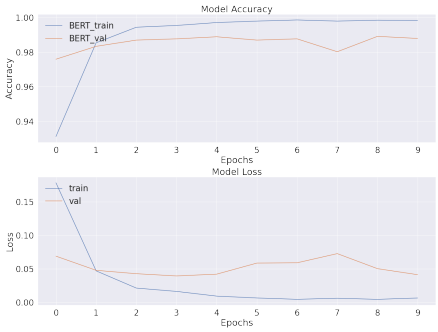

Figure 6. Training History for the BERT method on Binary Dataset

The model misclassified only a small proportion of data, with 76 unreliable articles predicted as reliable and 8 reliable articles predicted as unreliable. This demonstrates the model's strong ability to handle binary classification tasks effectively.

5.5. Method 5: Transformer(BERT) on Multi-Categories Dataset

When trying different batch size and epoch, we got the following result:

Batch size	Total epoch	Accuracy
16	10	27%
32	10	33%
64	10	22%
32	20	31%
32	30	31%

Table 4. Accuracy for differen combinations of batch size and epoch

From the first three runs where we used the same number of epoch but we changed the batch size, we can see that using batch size of 32 produce the best accuracy. Then we tried increasing total number of epoch in the last two runs, and concluded that batch size of 32 and epoch of 10 are the best hyper parameters. Below is a graph showing the training history for batch size of 32 and epoch of 10.

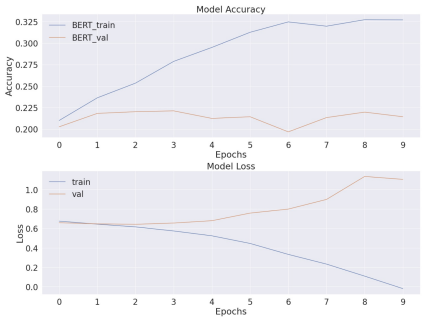

Figure 7. Training History for the BERT method on multiple feature dataset

Then we moved on to trying different model design. The above experiment on hyper parameters were ran on the model where only input_ids and attention_masks were fed as input. Then we tried adding the additional-feature layer, containing the 5 integer features, into the model. This time the model only produced an accuracy of 0.24 after 10 epoch. Below is a graph showing the training history for batch size of 32 and epoch of 10.

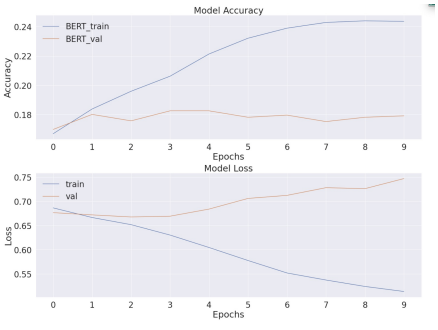

Figure 8. Training History for the BERT method on multiple feature dataset

The model that does not contain the additional features out-performed the model that does contain it. Thus, the additional features does not contribute to the model at all, and we decided to exclude it. Below is the confusion matrix generated using the hyper parameters and model designs that performed the best.

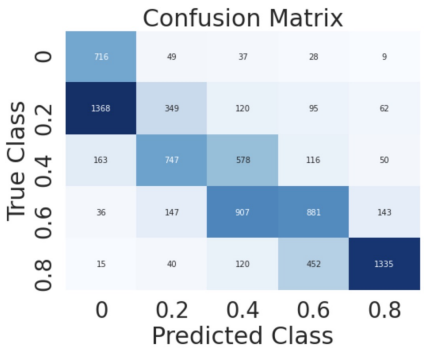

Figure 9. Training History for the BERT method on multiple feature dataset

Although the accuracy of BERT model applied on this dataset wasn't very high, from the confusion matrix we can still see that the model is making some strong predictions. However, the model fails to precisely categorize each piece of news, but can still tell its truthfulness on a spectrum. Considering that there are 5 different labels for the model, we believe this performance is already outstanding, and demonstrates BERT's ability to extract usefull features from a text and learn about its meaning and truthfulness.

6. Discussion

After running the classification methods, we performed manual error analysis to identify patterns and understand why certain examples were misclassified.

6.1. LLM Method Analysis:

Specifically, we focused on false positives, where articles labeled as reliable (0) were incorrectly classified by the model as unreliable (1). Here we discuss two such cases, analyze potential causes, and propose ways to improve model accuracy based on the observed errors.

- First False Positive Example:
  - Article: "France and Germany Unite to Oppose President Trump's Refugee Ban" - Breitbart
  - Label: Reliable (0)
  - Predicted Label: Unreliable (1)

  This article discusses political reactions to a controversial U.S. policy (Trump's refugee ban), with strong opinions expressed by political figures. The model may have been influenced by the presence of emotionally charged language and references to political conflict, which are often common in unreliable news. Additionally, the presence of a contentious topic (refugee policies) may have triggered the model to associate the article with potentially unreliable information, as such topics are frequently sensationalized in unreliable sources.

  Some potential improvements we could make on this example are:

  - Contextual Feature Engineering: Adding contextual features that detect the source credibility (e.g., source domain and publication reputation) could help the model distinguish between reputable sources discussing controversial topics and genuinely unreliable news.
  - Sentiment Calibration: Incorporating sentiment analysis to account for the tone and intensity of language may help the model recognize that strong language does not always imply unreliability. This could reduce false positives for articles that are credible yet discuss divisive issues.

- Second False Positive Example
  - Article: "UC Berkeley Investigates Alleged Employee Who Assaulted MILO Fan During Riot" - Breitbart
  - Label: Reliable (0)
  - Predicted Label: Unreliable (1)

  This article covers an investigation into violence during a public event, mentioning aggressive language, rioting, and alleged involvement of university employees. The model likely misclassified this article as unreliable due to the sensational nature of the content, including terms associated with violence, gang affiliations, and social media controversies. Articles that report on riots or public conflicts, even from reputable sources, may trigger the model's unreliable label if it heavily relies on identifying emotionally charged keywords.

  Some potential improvements we could make on this example are:

  - Keyword Sensitivity Adjustment: Adjusting the model to recognize and discount certain sensational keywords (such as "riot," "violence," and "gang") when they appear in reputable sources could help it better handle articles discussing high-profile events factually.
  - Source-specific Fine-Tuning: Fine-tuning the model on a dataset with labeled articles from different news sources (e.g., Breitbart, CNN, BBC)

and associated reliability ratings could help it learn the credibility of certain sources, even when they cover sensational topics.

From these false positive examples, a recurring theme emerged: articles covering divisive, high-profile topics with emotionally charged language were more likely to be misclassified as unreliable. This suggests that the model currently over-emphasizes linguistic cues associated with sensationalism, which can result in false positives when credible sources report on contentious events.

Based on our error analysis, we suggest two concrete strategies to improve the model's accuracy:

- Incorporate Source Reliability Features: Adding a feature that evaluates the source's reputation could help the model differentiate between credible sources reporting on controversial topics and genuinely unreliable sources. This could involve building a database of news sources with associated reliability scores or using a third-party service to assess source credibility.
- Refine Language Sensitivity: Adjust the model's sensitivity to sensational language and keywords. This could involve training it to better recognize credible reporting styles, even when covering violent or controversial events, by analyzing word choice, tone, and context. By doing so, the model could reduce false positives related to articles that are factually reporting on divisive topics.

6.2. Logistic Regression Method Analysis:

We printed out the titles of false-negative (true news predicted false) and false-positive (false news predicted true) news. Some examples for false-negatives are:

(1) Humiliated Hillary Tries To Hide What Camera Caught 15 Mins Before Rally
(2) Samantha Bee Explores the Dangerous Rise of the Alt Right in American Politics (Video)
(3) Review: 'Lion' Brings Tears for a Lost Boy, Wiped Dry by Google - The New York Times
(4) Clinton Campaign STUNNED As FBI Reportedly Reopens Probe Into Hillary Clinton Emails
(5) Pence Will Speak at Anti-Abortion Rally - The New York Times
(6) Trump Fans Rally Across the Nation to Support the President - The New York Times
(7) 11 Fiction Podcasts Worth a Listen - The New York Times
(8) All 100 Senators Contacted Russian Government This Week

And some examples for false-positives are:

(1) A Back-Channel Plan for Ukraine and Russia, Courtesy of Trump Associates - The New York Times
(2) Russian Researchers Discover Secret Nazi Military Base 'Treasure Hunter' in the Arctic [Photos]
(3) In Major League Soccer, Argentines Find a Home and Success - The New York Times
(4) Having Won, Boris Johnson and 'Brexit' Leaders Fumble - The New York Times
(5) How To Make Briquettes From Daily Waste
(6) News: PR Disaster: The President Of Panasonic Has Been Forced To Resign After 60,000 Panasonic TVs Ascended To Heaven Without Warning
(7) Franken Calls for 'Independent Investigation' Into Trump's 'Putin Crush' - Breitbart
(8) Venezuela Muzzles Legislature, Moving Closer to One-Man Rule - The New York Times

One pattern we observe is that a lot of the false-positives are international news, while most of false negatives are domestic news. This could stem from several factors related to the dataset, model training, and the nature of the logistic regression approach used. Domestic news might use language that differs in tone, style, or vocabulary compared to international news. If the model has learned certain linguistic features that tend to appear in the domestic or international context, this could lead to incorrect classifications, especially if domestic and international news sources handle sensationalism or formal language differently. Moreover, international news may contain topics or keywords frequently associated with sensational or controversial themes, such as "Russia," "Brexit," "Nazi," or "Venezuela." The vector representation of these words may impair the judgment of the algorithm.

6.3. BERT Method on Binary Dataset Analysis:

6.3.1. Observations on Misclassified Samples

Example 1:

- **Text Characteristics:** The text contains long and complex sentences with highly specific details about political events and social issues.
- **Prediction:** The model classified it as `FALSE` (not fake) when the ground truth was `TRUE` (fake).
- **Analysis:**
  - The text is dense with factual information, which might make the model perceive it as reliable, even though it's fabricated or misleading.
  - The model struggles with identifying subtle misinformation embedded in extensive details.

Example 2:

- **Text Characteristics:** A heavily opinionated piece with references to constitutional history and political movements.
- **Prediction:** Classified as `FALSE` (not fake) instead of `TRUE` (fake).
- **Analysis:**
  - The model appears to have difficulty distinguishing between highly opinionated content and factual reliability.
  - The use of formal language and historical context may mislead the model into interpreting the content as authentic.

Example 3:

- **Text Characteristics:** A piece focused on diversity in the beauty industry, including nuanced discussions about social constructs and representation.
- **Prediction:** Misclassified as `TRUE` (fake) when the ground truth was `FALSE` (not fake).
- **Analysis:**
  - The model struggles with non-news topics that deviate from the political or societal themes prevalent in the training data.
  - The conversational tone and less structured narrative might have contributed to the error.

6.3.2. Patterns in Model Errors

- **Long and Complex Texts:** The model struggles with lengthy articles containing dense, highly detailed content. It likely fails to parse and evaluate the content's overall coherence and truthfulness.
- **Subtle Misinformation:** Articles embedding minor inaccuracies within generally factual statements are harder for the model to flag as fake.
- **Non-News Content:** When presented with content that does not fit the typical format of news articles (e.g., lifestyle, opinion pieces), the model tends to misclassify them.
- **Highly Polarized Content:** The model occasionally misinterprets highly polarized or strongly opinionated articles, likely because they resemble the tone of actual news articles in the training data.

6.3.3. Comparison to Expectations

- **Performance vs. Expectations:**
  - The model performs well on traditional news formats but struggles with nuanced cases such as opinionated pieces or content mixing fact and fiction.
  - While the accuracy and F1 scores are high overall, the errors on these edge cases indicate room for improvement in understanding context and subtle cues.
- **Surprises:**
  - **Surprised by:** The model's inability to recognize factual content in less formal or conversational styles, as seen in Example 3.
  - **Could process correctly:** Articles with clear factual inconsistencies or obvious errors.

6.3.4. Suggestions for Improvement

- **Enhanced Preprocessing:** Include preprocessing techniques that highlight fact-checking signals (e.g., named entities, hyperlinks, and references).
- **Fine-Tuning with Specific Examples:** Augment training data with samples similar to the misclassified examples (e.g., dense, opinionated, or less structured content).

6.4. BERT Method on Multi-Category Dataset Method Analysis:

6.4.1. Observations on Misclassified Samples

We focused out observation on news pieces that had the worst prediction, meaning its predicted truthfulness was the furthest away from the label. Below are some news that were on the very false spectrum ("pants-fire" or false) but was classified as true or mostly-true:

(1) Under Lt. Gov. Lee Fisher, Ohio is 44th in the country in terms of getting money actually into worker retraining.
(2) Says an Obama administration policy prohibits people who work with at-risk youth from promoting marriage as a way to avoid poverty.
(3) Says House Democrats voted to use your tax dollars for abortions by voting against bill defunding Planned Parenthood.
(4) Elorza wants to teach our public school children about the non-existence of God.
(5) McCain "still thinks it's okay when women don't earn equal pay for equal work."

and these are news that were true or mostly-true but was predicted as false or "pants-fire":

(1) The House has never failed to pass a budget in the modern era.
(2) Nothing in the Constitution explicitly guarantees our right to vote.
(3) I have spent virtually every weekend since Memorial Day in the Panhandle.
(4) There is nothing in the current state public records law that prohibits sensitive or confidential business information from being just that, confidential.
(5) The poverty rate amongst women is the worst its been in 17 years and the extreme poverty rate is the worst its ever been.

6.4.2. Patterns in model errors

- The news that are very false but was classified as true seem to all contain the name of some political figure or is a quote from some some institution. The model probably over-predicted the truthfulness after seeing these names or institutions, because usually these are specific information that adds to a news' accuracy.

- The news that are true but are classified as false all used very extreme wording, such as "never", "nothing", or "worst". Political figures often use these words to exaggerate things to win affection of certain groups of voter, and therefore it is usually a sign that it is fake. Thus the model treated these words as indicators for false news, although some of them may be completely real.

6.4.3. Suggestions for improvement

To address the identified patterns in model errors, several improvements can be made to the algorithm. First, incorporate contextual embeddings for named entities by adding a Named Entity Recognition (NER) layer or pre-processing step to explicitly detect and differentiate names of political figures or institutions. This will allow the model to treat these entities as contextual features rather than direct indicators of truthfulness. Second, enhance the tokenizer by incorporating custom preprocessing rules that flag specific words, such as "never," "nothing," or "worst," and feed them as separate features to the model. This can help the model understand their contextual significance rather than blindly associating them with false news. Additionally, introducing an attention mechanism or fine-tuning BERT with an emphasis on extreme wording patterns may enable the model to better capture nuanced linguistic features. Lastly, augment the training data with counterexamples, such as true news articles containing extreme wording and false news without extreme wording, to improve the model's robustness to such biases. These improvements can help mitigate overgeneralization and lead to more accurate classifications.

7. Conclusion

In this project, we explored various methods for detecting fake news, including both baseline machine learning techniques (Logistic and Softmax Regression), a sophisticated Large Language Model (LLM), and a pre-trained bidirectional transformer model (BERT) for text classification. Each approach allowed us to evaluate and understand different facets of fake news detection, from simple linear classifications to more nuanced language understanding provided by the LLM and transformer model.

Our results demonstrated that the BERT method achieved the highest accuracy, while the LLM provided useful insights but also highlighted areas for improvement in handling emotionally charged and high-profile topics. The baseline logistic and softmax regression approaches were able to classify more successfully than random chance, but were otherwise unable to decode the complex nuances in the text to achieve a high accuracy in classification. Through manual error analysis, we identified patterns in misclassifications, particularly false positives where reliable articles were incorrectly classified as unreliable. Overall, however, we were unable to meaningfully improve the accuracy of the smaller models, because they weren't able to capture the complex relationships between words in the same way as BERT.

Throughout the process of this final project, we learned about the limitations of linear regression models and LLMs, and the capabilities of BERT in understanding and classifying text. We also gained a better understanding of the resources publicly available for developing and testing machine learning models by evaluating different techniques and datasets, both locally and using cloud-based servers like Kaggle Cloud. Our research also helped us understand how automated fake news detection systems work, and the accuracy of the current state-of-the-art.

All of our code can be found at https://github.com/Dionysssss/Fake_News_Classifier

References

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2019.

Hu, L., Wei, S., Zhao, Z., and Wu, B. Deep learning for fake news detection: A comprehensive survey. *AI Open*, 3:133–155, 2022. ISSN 2666-

6510. doi: https://doi.org/10.1016/j.aiopen.2022.09.001. URL https://www.sciencedirect.com/science/article/pii/S2666651022000134.

Jerifate, j. Fake news: Basic work tensorflow with bert , June 2021. URL https://www.kaggle.com/code/jerifate/fake-news-basic-work-tensorflow-with-bert. Kaggle.

Lifferth, W. Fake news. https://kaggle.com/competitions/fake-news, 2018. Kaggle.

Sastrawan, I. K., Bayupati, I., and Arsa, D. M. S. Detection of fake news using deep learning cnn–rnn based methods. *ICT Express*, 8(3):396–408, 2022. ISSN 2405-9595. doi: https://doi.org/10.1016/j.icte.2021.10.003. URL https://www.sciencedirect.com/science/article/pii/S2405959521001375.

Shu, K., Sliva, A., Wang, S., Tang, J., and Liu, H. Fake news detection on social media: A data mining perspective. *ACM SIGKDD Explorations Newsletter*, 19(1):22–36, 2017.

Wang, W. Y. Liar, liar pants on fire: A new benchmark dataset for fake news detection. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*, pp. 422–426, Vancouver, Canada, 2017. Association for Computational Linguistics.