Fake News Detection Using Neural Network Language Models

Christopher Farrer
farrerc@wwu.edu

Maxwell Schultz
schultm8@wwu.edu

Alex Sitzman
sitzmaa@wwu.edu

Abstract

This project aims to classify news article titles as sarcastic or non-sarcastic based on their content and context. Initially focused on sarcasm detection, the project pivoted to fake news detection due to dataset limitations. We used datasets from Kaggle and implemented multiple models, including Neural Bag of Words, Recurrent Neural Networks (RNNs), and Transformers. Our findings reveal that while Transformer models exhibit high accuracy, they struggle with nuanced contexts such as satire, and result in longer training times over other models. This report details our methodology, experimental results, and future directions for improving sarcasm detection.

1 Introduction

Neural Networks have become pivotal in various NLP tasks due to their ability to learn complex patterns from large datasets. This project aimed to leverage neural networks and traditional machine learning models to detect sarcasm in news headlines. Sarcasm detection holds significant ethical benefits, as it can improve the understanding and moderation of online communication, reducing misunderstandings and promoting healthier online interactions. However, limitations in the dataset necessitated a pivot to fake news detection. Fake news detection carries ethical benefits as well, as it helps combat misinformation, which can influence public opinion and decision-making processes. By ensuring the authenticity of news articles, such systems contribute to a more informed and rational public discourse. This paper discusses our approach, experimental setup, results, and the implications of our findings.

2 Related Work

Fake News Detection The proliferation of fake news has become a major concern, impacting political, economic, and social landscapes. (Agarwal et al., 2019) conducted a comprehensive study on fake news detection using a combination of NLP techniques and machine learning classifiers. They employed methods such as bag-of-words, n-grams, count vectorizer, and TF-IDF, training their dataset on five different classifiers. Their findings indicated that precision, recall, and F1 scores are critical metrics in determining the most effective model for fake news detection .

Sarcasm Detection Sarcasm detection poses unique challenges due to its inherently ambiguous nature. (Jain et al., 2017) explored sarcasm detection in tweets by utilizing ensemble-based approaches, specifically a voted ensemble classifier and a random forest classifier. Their research highlighted the difficulty in detecting sarcasm due to the evolving nature of language and the use of emoticons, which can alter the polarity of text. Their methodology diverged from traditional sentiment analysis by focusing on the presence of positive sentiment attached to negative situations, thus improving the detection accuracy in social media contexts .

Similarly, (Băroiu and Ștefan Trăușan-Matu, 2022) provided a systematic literature review on automatic sarcasm detection, tracing its evolution from 2010 to the present. They emphasized the growing popularity of multi-modal approaches and transformer-based architectures in recent years. Their work not only critiqued past research but also proposed future directions, underscoring the necessity for advanced models capable of handling the complexities of sarcasm in natural language .

3 Approach

Our approach utilizes the PyTorch library to implement and train various sequence classification models for sentiment analysis, specifically targeting movie reviews. The implementation details and models are based on the tutorials provided by (Trevett, 2023), which offer a comprehensive guide on using PyTorch for sentiment analysis tasks. Below, we outline the main steps and models used in our approach.

3.1 Models and Tutorials

3.1.1 Neural Bag of Words:

This tutorial introduces the basic workflow of a sequence classification project using a neural bag-of-words model. It covers data loading and preprocessing using the datasets and torchtext libraries. The model is simple yet effective for understanding the foundational concepts of sequence classification.

3.1.2 Recurrent Neural Networks (RNN):

Building on the basic workflow, this tutorial focuses on improving the model's performance by switching to a recurrent neural network (RNN) model. Specifically, it implements a long short-term memory (LSTM) RNN, which is one of the most commonly used variants of RNNs due to its ability to handle long-range dependencies in sequential data.

3.1.3 Transformers:

The final tutorial demonstrates how to use the transformers library to load a pre-trained transformer model, specifically BERT (Bidirectional Encoder Representations from Transformers). BERT, introduced in the paper by Devlin et al., provides high performance for various NLP tasks, including sequence classification. The tutorial covers loading the pre-trained BERT model and fine-tuning it for sentiment analysis.

3.2 Datasets

- News articles tagged by sarcasm sentiment (Misra, 2019).
- Fake news classification dataset (Shahane, 2024).
  Note: This dataset required some white space culling and the removal of non-English characters to be used.

4 Experiments

4.1 Preprocessing

The Fake News Classification dataset included many blank spaces, white space, and in some cases, non-English characters (e.g., Arabic). Our models were not configured to handle these inconsistencies. To prepare the dataset for training, the following preprocessing steps were performed:

- **Whitespace Removal:** We removed leading and trailing whitespace from all text fields to ensure consistent input for the models.
- **Non-English Character Removal:** Rows containing non-English characters were removed to avoid potential encoding issues and maintain uniformity in the dataset. This step resulted in the exclusion of around 500 rows.
- **Tokenization:** The text data was tokenized into individual words using the NLTK library. This step involved splitting the text into tokens, which are the smallest units of meaning.
- **Dataset Splitting:** We split our initial dataset into a 15-85 split for test and training data, then in good practice split our training data set again 15-85 for our valid and training sets so that the test set remained untouched through all processes. The valid set could be used for hyperparameter training as well if we chose to expand upon that

4.2 Neural Bag of Words

The Neural Bag of Words (NBOW) model was one of the simpler models used in our experiments. It serves as a baseline for comparison with more complex models. The parameters for this model were as follows:

- **Vocabulary Creation:** As part of the preprocessing procedure, our neural bag-of-words model generated a vocabulary from the dataset. A default index was assigned to handle unknown words and prevent errors.
- **Indexing:** We implemented an indexing system to map words in the dataset to their corresponding vocabulary entries.
- **Data Loaders:** Data loaders were constructed to efficiently load batches of data into the model during training.
- **Vectorization:** The model utilized GloVe embeddings for vector representation of the words.

4.3 Recurrent Neural Network

We implemented a Recurrent Neural Network (RNN) using Long Short-Term Memory (LSTM) units to capture the sequential dependencies in the text data. The parameters for the RNN model were as follows:

- **Pretrained Embeddings:** Pretrained GloVe embeddings were loaded and used to initialize the embedding layer of the model.
- **Optimization:** The Adam optimizer was configured with a learning rate of 5×10^{-4}.
- **Loss Function:** Cross-entropy loss was used as the criterion for training the model.
- **Training and Evaluation:** Functions were created for training and evaluating the model. These functions included loss calculation, accuracy computation, and gradient updates.
- **Training Loop:** The model was trained for ten epochs. The best model, based on validation loss, was saved. Metrics for training and validation loss and accuracy were recorded and plotted.
- **Testing:** The model's performance was evaluated on the test dataset, and the final test loss and accuracy were reported.
- **Sentiment Prediction:** A function was developed to predict the sentiment of input text. This function tokenized the input, performed inference using the trained model, and returned the predicted class and associated probability.

4.4 Transformer

The Transformer model, specifically BERT (Bidirectional Encoder Representations from Transformers), was fine-tuned for the task of fake news detection. The parameters for the Transformer model were as follows:

- **Collation:** A collate function was defined to pad sequences and create batches for the data loader. This function ensured consistent input sizes for the model.
- **Data Loaders:** Data loaders were created for the training, validation, and test datasets to facilitate batch processing.
- **Model Definition:** The Transformer model was defined using the `AutoModel` from the `transformers` library. A fully connected layer was added to adapt the model for our specific classification task. Optionally, the transformer parameters were frozen to prevent further training.
- **Optimization:** The Adam optimizer was configured with a learning rate of 1×10^{-5}.
- **Loss Function:** Cross-entropy loss was used as the criterion for training the model.
- **Training and Evaluation:** Functions were created for training and evaluating the model. These functions included loss calculation, accuracy computation, and gradient updates.
- **Training Loop:** The model was trained for three epochs. The best model, based on validation loss, was saved. Metrics for training and validation loss and accuracy were recorded and plotted.
- **Testing:** The model's performance was evaluated on the test dataset, and the final test loss and accuracy were reported.
- **Sentiment Prediction:** A function was developed to predict the sentiment of input text. This function tokenized the input, performed inference using the trained model, and returned the predicted class and associated probability.

Each model was evaluated using standard metrics such as accuracy, precision, recall, and F1-score to compare their performance on the sarcasm and fake news detection tasks.

5 Results

5.1 Neural Bag Of Words

The Neural Bag of Words (NBoW) model is a straightforward yet powerful architecture for text classification tasks. Here's a breakdown of the key findings:

- **Model Parameters:** With a total of 15,070,802 trainable parameters, the model demonstrates considerable flexibility in learning representations from the input data.

- **Pre-trained Word Embeddings:** Leveraging pre-trained GloVe word embeddings facilitates the initialization of the embedding layer weights. These embeddings enrich the model's understanding of word semantics.

- **Training and Evaluation:** Through training with cross-entropy loss and Adam optimizer, the model iteratively improves its performance over multiple epochs. Evaluation on both training and validation datasets monitors progress and generalization.

- **Training Progress:** Over 10 epochs, both training and validation losses decrease while accuracies increase, indicating the model's ability to learn meaningful representations and generalize effectively.

- **Test Performance:** Evaluation on a separate test dataset demonstrates the model's robustness, with a test loss of .127 and a test accuracy of 96.1%. Precision was 0.964, Recall was 0.945, and f_1 was 0.954.

- **Predictions:** Application of the model on sample text inputs showcases its ability to predict sentiment labels with associated probability scores, indicating confidence in its predictions.

- **Shortcomings:** The model was unable to identify false stories with unknown words, in the fake article "In an unexpected and whimsical turn of events, the United States Senate has passed a bill requiring all citizens to adopt a unicorn by 2025. The Unicorn Adoption Act passed ..." the model identified the article as true, with 55% confidence as it had never seen the word unicorn before

5.2 Recurrent Neural Network

Recurrent Neural Networks (RNNs) offer a sophisticated approach to text classification tasks, with the following insights gleaned from the experiment:

- **Training and Optimization:** The model was trained using the Adam optimizer with cross-entropy loss. During training, the model iteratively adjusted its parameters to minimize the loss function, optimizing classification performance.

- **Hyperparameters:** Hyperparameters such as embedding dimension, hidden dimension, number of layers, and dropout rate were carefully selected to balance model complexity and performance.

- **Training Progress:** Over 10 epochs, both training and validation losses decreased, indicating the model's ability to learn meaningful representations from the data. Concurrently, training and validation accuracies increased, signifying improved classification performance.

- **Evaluation:** Evaluation on a separate test dataset revealed the model's robustness, with a test loss of 0.068 and a test accuracy of 97.7%. These metrics demonstrate the model's ability to generalize well to unseen data.

- **Prediction Analysis:** Application of the trained model on sample text inputs showcased its capability to predict sentiment labels with associated probability scores. This analysis highlighted the model's confidence in its predictions and its ability to handle nuanced sentiment expressions.

5.3 Transformer

The utilization of Transformers, particularly BERT (Bidirectional Encoder Representations from Transformers), presents a significant advancement in text classification tasks. Here's a comprehensive analysis of the experimentation with Transformers:

- **Data Processing:** Initially, the dataset underwent preprocessing steps to ensure compatibility with the Transformer model. Tokenization, performed using the Hugging Face transformers library, encoded text inputs into numerical representations suitable for the Transformer architecture.

- **Model Architecture:** The BERT model, initialized with pre-trained weights ('bert-base-uncased'), served as the backbone of the Transformer-based text classifier. It consisted of multiple transformer layers, facilitating bidirectional context understanding.

- **Fine-Tuning Approach:** The BERT model was fine-tuned on the task-specific dataset using a simple linear classification head. This fine-tuning process adapted the pre-trained

Transformer to the nuances of the text classification task at hand.

- **Training Dynamics:** During training, the model iteratively updated its parameters to minimize the cross-entropy loss using the Adam optimizer. Training and validation losses were monitored across epochs to assess model convergence and prevent overfitting.
- **Evaluation Metrics:** Evaluation on a separate test dataset revealed the model's performance, yielding a test loss of 0.024 and a test accuracy of 99.2%. These metrics reflect the model's ability to generalize well to unseen data and effectively classify text inputs.
- **Analysis and Interpretation:** Visualization of training dynamics through loss and accuracy curves provided insights into the model's learning progress and convergence behavior. Additionally, model predictions on sample text inputs demonstrated its capability to classify sentiment effectively.

6 Discussion

Our results show that each of the models are efficient function approximators capable of a test accuracy of 95% and greater. The smaller models (NBoW, and RNN) overall required a much shorter compute time to finish training while giving near perfect accuracy on the training set. These models generalized well to the data which we gave, but supplementary testing showed that certain fake news datapoints which express themselves in very natural speech can trick the models.
These examples showed the power of neural networks and their flexibility in a variety of optimization problems, especially applications within natural language processing. The main limitation of our model was the lack of semantic understanding in relation to actual sentiments within the training context. These contexts are important to identifying whether material is fake news, as the task can sometimes require the subtle informational details which this dataset cannot give.

7 Limitations

Our study identified several limitations in both sarcasm and fake news detection:

- **Challenges in Sarcasm Detection:** Sarcasm detection proved challenging due to the nuanced and context-dependent nature of sarcasm, which our models often failed to capture accurately. The dataset's limitations, such as lack of contextual information and varying styles of sarcastic expression, contributed to this difficulty.
- **Resource Demands of Transformer Models:** While our Transformer models showed high accuracy in fake news detection, they required substantial computational resources and longer training times compared to other models. This high demand for resources limits the feasibility of deploying such models in real-time applications or on devices with limited computational power.
- **Preprocessing Impact on Fake News Dataset:** Our preprocessing steps for the fake news dataset, including the removal of non-English characters and white space culling, may have led to the loss of potentially valuable information, potentially affecting the model's performance.
- **Language and Cultural Limitations:** Our models were trained and tested primarily on English language data, which restricts the generalizability of our findings to other languages and cultural contexts. Future work should focus on addressing these limitations by incorporating more diverse datasets and improving model efficiency.

8 Conclusion and Future Work

In all three models the observed accuracy was substantial and indicated a strong understanding of sentiment by the neural networks. However the model had large shortcomings when it came to identifying fake news that was outside of the dataset we used. A more diverse dataset with stronger context will be required to allow the model to correctly identify fake news in different location. The root of the models' issues come from a lack of full real world understanding, and subtleties within the data which requires a more granular dataset.
In future projects, we shall implement the use of a much larger GPT model which will be able to understand more subtle semantic contexts of the input material. A larger dataset would also be beneficial in giving more contexts of fake news for the model to learn from. This larger set should contain a wider timeline of news articles as to not train towards the biases of a specific period in time.

References

Vasu Agarwal, H. Parveen Sultana, Srijan Malhotra, and Amitrajit Sarkar. 2019. Analysis of classifiers for fake news detection. In *International Conference on Recent Trends in Advanced Computing 2019 (ICRTAC 2019)*. Elsevier.

Alexandru-Costin Băroiu and Ștefan Trăușan-Matu. 2022. Automatic sarcasm detection: Systematic literature review. *Information*, 13(8):399.

Tanya Jain, Nilesh Agrawal, Garima Goyal, and Niyati Aggrawal. 2017. Sarcasm detection of tweets: A comparative study. In *2017 Tenth International Conference on Contemporary Computing (IC3)*, Noida, India. IEEE.

Rishabh Misra. 2019. News headlines dataset for sarcasm detection. Kaggle. Updated 5 years ago.

Saurabh Shahane. 2024. Fake news classification on welfake dataset. Kaggle.

Ben Trevett. 2023. Pytorch sentiment analysis. GitHub repository.