UNIVERSITY PARTNER

UNIVERSITY OF WOLVERHAMPTON

HERALD COLLEGE KATHMANDU

Artificial Intelligence and Machine Learning (6CS012)

Fake News Detection using RNN, LSTM, and Word2Vec Embeddings

Kulbhushan Basnet

Aaranya Lal Maskey

Bigyen Bhandari

Brisha Shrestha

Table of Contents

1. Abstract

This project tries to utilize deep learning techniques to address the real problem of fake news detection. Since misinformation spreads rapidly through various media, it becomes essential to building automated detection systems that preserve the credibility of information. In this study, three deep learning models were implemented and tested against each other: the Simple RNN, the LSTM, and an LSTM+Pretrain GloVe Embeddings model. Using a labeled dataset that contains articles of both real news and fake news, the models were tested for the accuracy of their classification of news content. The results reveal that the plain vanilla LSTM achieved the highest accuracy rate of 97.98%, far outpacing the Simple RNN of 53.88%, and LSTM+Word2Vec of 93.75%.

2. Introduction

Fake news is the major challenge to accuracy of information and public dialogue on larger scales. Such articles are usually manufactured to look like authentic news reports but contain false or misleading information (Wise, 2025). In the absence of manual fact-checking, which is time-consuming and cannot keep up with the pace at which information is made available online, automated solutions become really desirable.

The goal of this project is to design and compare deep-learning models that can automatically determine whether news articles are *real* or *fake* based on textual information. Implementing such a model would have many implications like the following:

* Keeping information credible on digital platform
* Providing performant AI tools for journalists and fact-checkers
* Helping readers evaluate the reliability of online content themselves
* Limiting the spread and effect of misinformation

2.1Previous Works

2.1.1 Text based Sentiment Analysis using LSTM

The paper focuses on the automated process of sentiment analysis, determining the emotional tone behind text to extract subjective information from various forms of textual data. Focusing on the limitations of manual analysis, the study explores deep learning techniques, particularly Long Short-Term Memory (LSTM) networks, as a good solution for handling sequential text data. LSTMs are known for their ability to capture long-term dependencies, which makes them effective in understanding contextual nuances in language. The research demonstrates the application of LSTM for classifying text likely tweets, reviews, or similar content, into sentiment categories such as positive, negative, and neutral. Furthermore, the paper provides a comparative analysis against traditional

machine learning approaches, underlining LSTM's superior performance in learning complex patterns within text data to facilitate more accurate sentiment predictions.

This work contributes to the body of research in natural language processing (NLP) by addressing the challenges of processing unstructured text and automating sentiment analysis, thereby reducing the need for labor-intensive manual analysis in real-world applications. (Murthy, et al., 2020)

3. Dataset

The analysis utilized a labeled dataset of news articles with binary classification "True" or "Fake".

Source- Provided by Mr. Siman Giri.

Data Size- The dataset contains 20,000 articles split between the two classes.

Features- The primary feature is the text content of each article, along with their label.

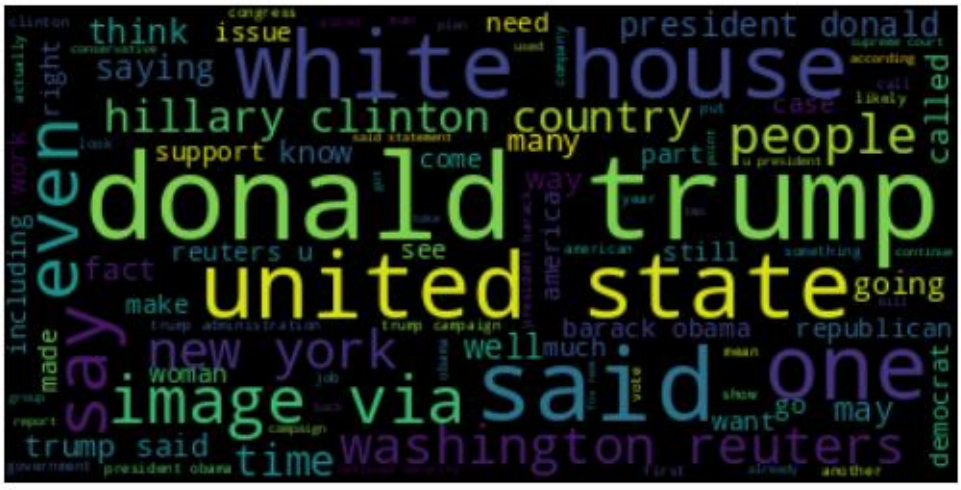

Figure 1 Word Cloud

The word cloud shown above visualizes the most frequently occurring words in the news dataset. Larger words appear more often, indicating their prominence in the text. Terms like "Donald Trump," "White House," "United States," and "Hillary Clinton" suggest a strong focus on US politics. Common reporting words such as "said," "say," and "image" vireflect the structure of typical news articles. This visualization helps highlight the central

themes and key entities discussed in the dataset, which is useful for understanding the context and guiding the development of a fake news detection model.

4. Methodology

4.1 Text Preprocessing

```
def clean_text(text):
    text = text.lower()
    text = re.sub(r'http\S+', '', text)
    text = re.sub(r'@\w+', '', text)
    text = re.sub(r'#\S+', '', text)
    text = re.sub(r'\d+', '', text)
    text = re.sub(r'[^a-zA-Z\s]', '', text)
    text = contractions.fix(text)
    tokens = word_tokenize(text)
    stop_words = set(stopwords.words('english'))
    tokens = [word for word in tokens if word not in stop_words]
    lemmatizer = WordNetLemmatizer()
    tokens = [lemmatizer.lemmatize(word) for word in tokens]
    text = ' '.join(tokens)
    text = re.sub(r'\s+', ' ', text).strip()
    return text
```

Figure 2 Text Cleaning Pipeline

The preprocessing pipeline was intended to bring about consistency in texts and to highlight key content words that may assist in deciding whether or not an article is genuine. Text cleaning was carried out exhaustively and included the following:

* Transform each text into lowercase-only
* Removed URLs, user mentions, hashtags, and numeric values
* Removed punctuation and special characters
* Expanded a contraction (changed "don't" to "do not")
* Break text into a list of tokens (tokenization)
* Filter out stopwords
* Lemmatize words to their root form
* Removed extra spaces

4.2 Model Architecture

Three distinct model architectures were implemented and compared:

4.2.1 Simple RNN Model

This model utilizes a basic recurrent neural network to process sequential data.

- Embedding layer: Embeds the tokenized words into dense vector representations.
- SimpleRNN layer (64 units): Deals with the sequence of data and tries to capture short-term relationships between words.
- Dense output layer: A single neuron output with sigmoid activation function for binary classification.

This architecture is computationally efficient but may struggle with long-range dependencies in sequences.

4.2.2 LSTM Model

The LSTM models use long short-term memory units, contrariwise, the traditional SimpleRNNs are somewhat unable to do so.

- Embedding layer: Learns vector representations of words in the input data.
- LSTM layer (64 units): Gating mechanisms for short- and long-range dependency capturing in sequential data.
- Dense output layer: Sigmoid-activated unit for binary classification.

LSTM units can treat long sequences well and at the same time minimize the vanishing gradient effect.

4.2.3 LSTM with Word2Vec Embeddings

This model builds on the LSTM architecture by providing the LSTM with Word2Vec embeddings to take advantage of the learned semantic relationships from a large corpus of data.
A description of the model architecture:

- Embedding layer with Word2Vec vectors: The embedding layer is initialized with 100-dimensional Word2Vec embeddings (e.g., GoogleNews-vectors-negative300.bin or your own trained version). The embedding matrix takes the words in the LSTM vocabulary and maps the words in vocabulary to their Word2Vec vector, based on the initialized Word2Vec data. Words that do not exist in the initialized Word2Vec vocabulary get initialized with random or zero vectors. This embedding layer is mostly frozen (trainable=False), meaning that the LSTM learns the Word2Vec data retains the semantic information captured from the original Word2Vec learning.
- LSTM layer (64 units): The LSTM layer exposes the embedded input so LSTM can learn the sequential and contextual aspects of the data for learning the temporal dependence in the text data.
- Dense output layer: The model has a dense output layer with one neuron (used for binary classification (real vs fake news) task) using the sigmoid activation function.

The Word2Vec embeddings allow the LSTM model to tap into a rich source of semantic knowledge of structure of word relationships, having the potential to better the performance of the LSTM on a classification task, especially when there was limited data used for training.

4.3 Training Configuration

Loss Function: Binary cross-entropy, appropriate for binary classification

Optimizer: Adam optimizer, which adapts learning rates for different parameters

Hyperparameters:

- Epochs: Maximum 20 with early stopping
- Validation Split: 20% of training data
- Early Stopping: Patience of 3 epochs monitoring validation loss
- Sequence Length: 95th percentile of training sequence lengths with post-padding

5. Experiments and Results

5.1 Model Performance Comparison and Evaluation

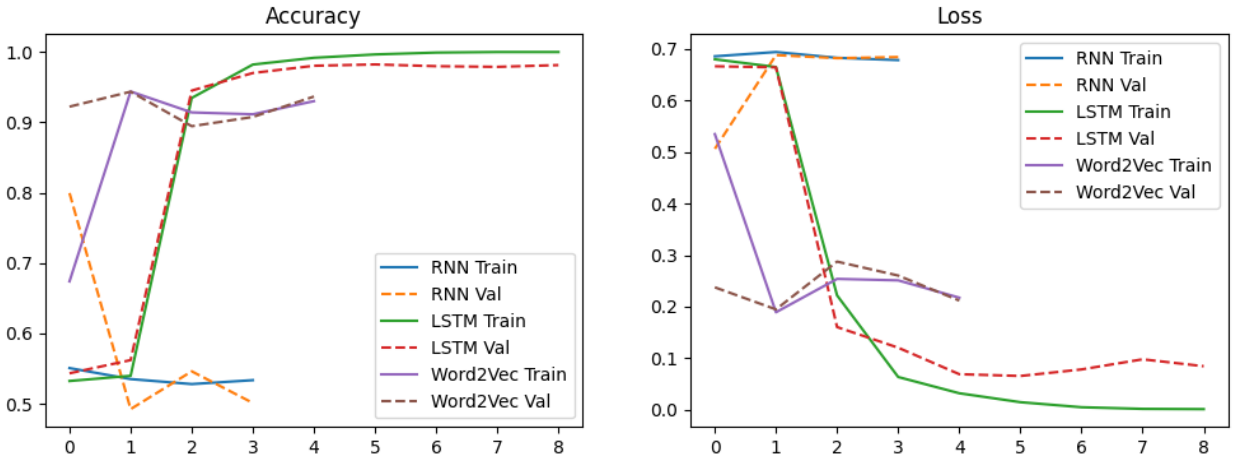

Figure 3 Model Accuracies and Losses

RNN Model

- Accuracy:0.79975 (79.97%)
- Classification Performance:
    - Precision for Negative Class (0): 0.75
    - Precision for Positive Class (1): 0.87
    - Recall for Negative Class (0): 0.89
    - Recall for Positive Class (1): 0.71
    - F1-Score for Negative Class (0): 0.82
    - F1-Score for Positive Class (1): 0.78

Key Observation: Severe class imbalance and poor performance, with extremely low recall for the positive class

LSTM Model

- Accuracy: 0.97875 (97.87%)
- Classification Performance:
    - Precision for both classes: 0.98
    - Recall for both classes: 0.98
    - F1-Score for both classes: 0.98

Key Observation: Outstanding and balanced performance across both classes

Word2Vec-Enhanced LSTM Model

- Accuracy: 0.94575 (94.45%)
- Classification Performance:
    - Precision for Negative Class (0): 0.99
    - Precision for Positive Class (1): 0.91
    - Recall for Negative Class (0): 0.90
    - Recall for Positive Class (1): 0.99
    - F1-Score for Negative Class (0): 0.94
    - F1-Score for Positive Class (1): 0.95

Looking across the metrics, the LSTM vastly outperformed the RNN. The RNN suffered from indications of vanishing gradients and difficulties learning from the signification class imbalance. In contrast, the application of the LSTM resulted in stable and accurate learning across the totality of the training process. Even while the LSTM required more computation and training time as a result of its more advanced architecture, the performance improvements, including exploding accuracy from 79% to 97.87%, warrant its use. In addition, gated memory would make it an ideal candidate for employing better generalizations and balanced predictions.

Key Observation: Excellent overall performance with slight variability between classes

1. RNN Confusion Matrix

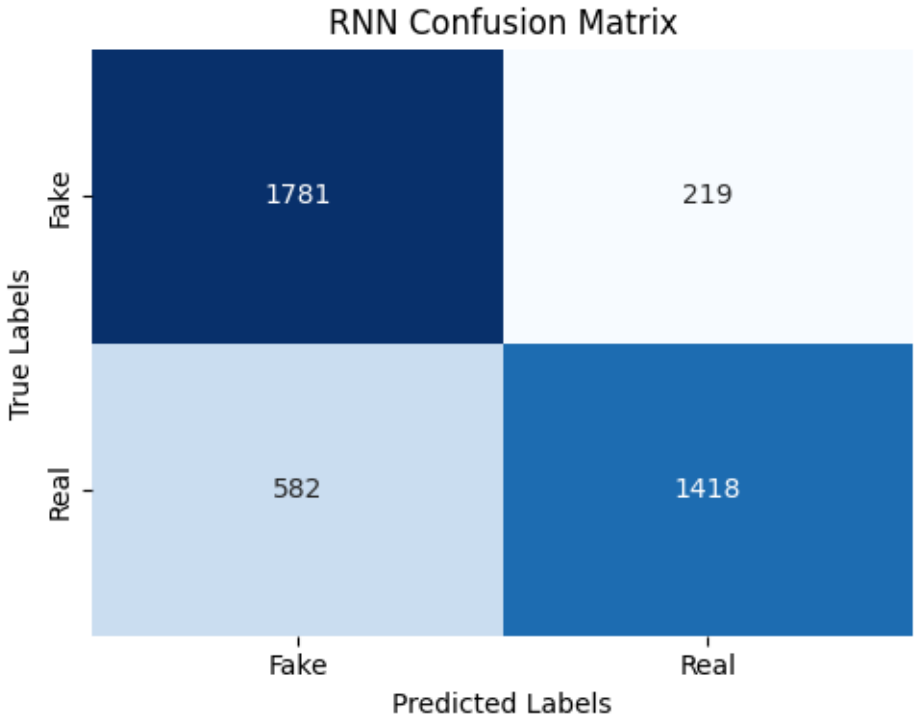

Figure 4 RNN Model- Confusion Matrix

Strong bias towards predicting the negative class.

2. LSTM Confusion Matrix

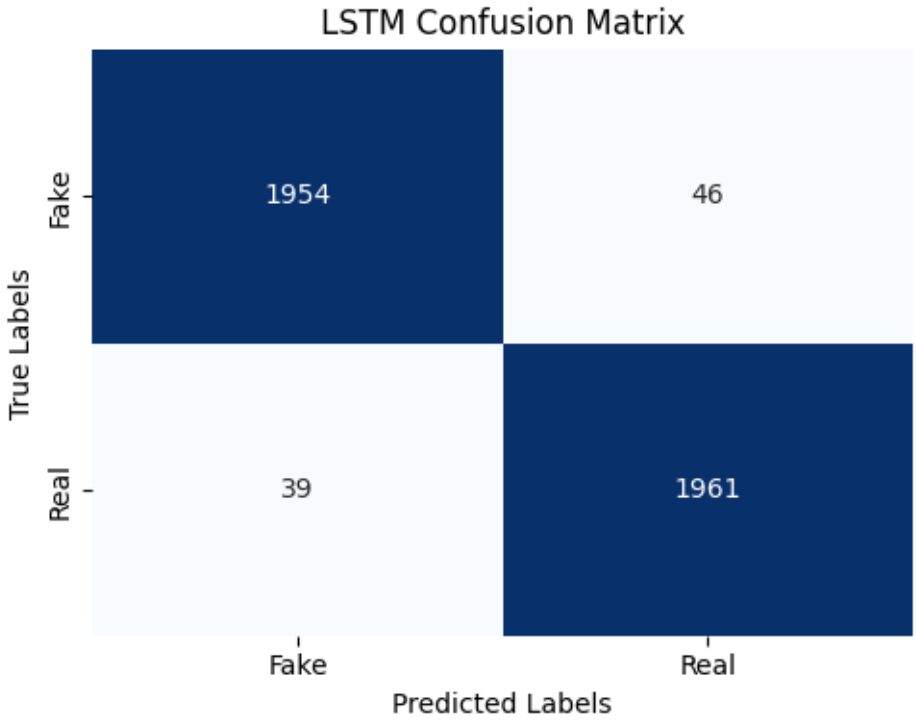

Figure 5 LSTM Model- Confusion Matrix

The model achieved near-perfect classification with minimal misclassifications, demonstrating a strong ability to distinguish between the classes. Predictions were well-balanced across both classes, indicating that the model did not favor one class over the other and maintained consistent performance throughout.

3. Word2Vec Confusion Matrix

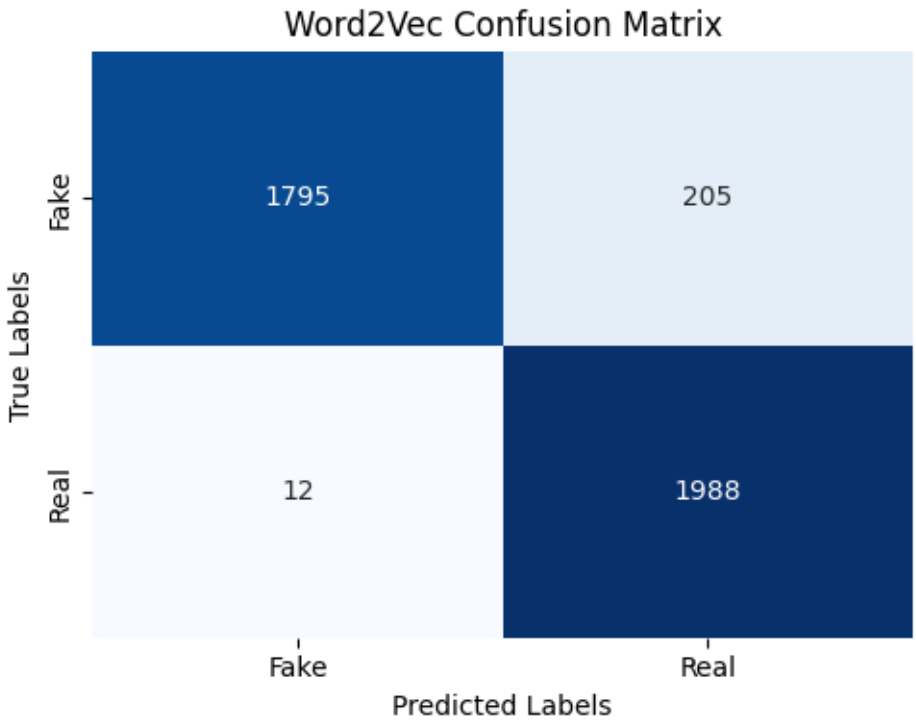

Figure 6 Word2Vec- Confusion Matrix

Slight increase in false positives (205) but still maintained high accuracy

5.2 Practical Application

A Gradio-based user interface was developed to demonstrate practical application, allowing users to:

- Input news text for analysis
- Select the model for classification (RNN, LSTM, or Word2Vec)
- Receive a classification result with confidence score

5.3 Training with different embeddings

A. Basic Recurrent Neural Network (RNN)

The baseline model was implemented with Simple RNN architecture coupled with dynamic embeddings. This way, the network was able to develop word representations in an organic way throughout training by adapting embeddings to maximize classification results. This gave the model an average of **79.98%** validation accuracy, which was natural considering the Simple RNN's weaknesses in modeling long-distance dependencies in text sequences.

B. Long Short-Term Memory (LSTM) with Trainable Embeddings

The LSTM architecture, however, replaced the Simple RNN and kept the trainable embedding setup. This gave the model an advantage in modeling longer-range dependencies in the text sequences. While training, the embeddings were refined to fit sentiment patterns, thus yielding a significant improvement in performance with **97.88% accuracy**—a 17.9% gain from the basic RNN.

C. LSTM with Static Word2Vec Embeddings

Pre-trained Word2Vec embeddings, trained on the Google News corpus, were used with an LSTM framework in the third variant. The embeddings, while fixed in time (no updates), allowed the model to leverage general linguistic patterns and idiosyncrasies

embedded within Word2Vec without specifically training from the task-at-hand. Although competitive (**94.58% accuracy**), this approach seemed to lag the training LSTM system by 3.3%, thus hinting that refining embedding for the specific task may outperform the general semantic richness of a pre-trained vector**.**

6. Conclusion and Future Work

The project that was undertaken focused on a variety of deep learning methods for the detection of fake news on textual information only. Three types of models were tested — a Simple RNN, a Long Short-Term Memory (LSTM), and an LSTM augmented with Word2Vec embeddings. Among the three, a simple LSTM performed best with an accuracy reaching 97.87%, much better than the Simple RNN at 79.97% and the Word2Vec embedding-based LSTM at 94.57%. These results highlight the power of LSTM networks to capture long-term dependencies, making them very well-suited for the analysis of sequential text data.

Adding Word2Vec embeddings should have helped make the model pick up on word semantics, yet it somehow still underperformed versus a plain LSTM trained from scratch. Potentially there could be an issue of mismatch between the pre-trained Word2Vec vectors and the particular characteristics of language in the dataset. The Simple RNN model suffered from lower recall and phenomena of class imbalance, exposing its inability to handle longer sequences and context preservation.

To prove that these could be used in practice, the models were deployed through a Gradio-based interface, thereby demonstrating their usability as an almost ready tool for evaluating news credibility under real-life settings.

These results being encouraging, there are still some options to improve and enlarge the system:

- Multiclass Classification: Extend beyond binary classification to categorize news into different types (e.g., satire, biased, propaganda, etc.).
- Larger and More Diverse Datasets: Incorporate multilingual or cross-domain datasets to improve generalizability and robustness across different contexts and languages.
- Fine-tuning Pretrained Models: Experiment with transformer-based architectures like BERT, RoBERTa, or DistilBERT, which have shown good results in NLP tasks.
- Explainability and Interpretability: Integrate explainable AI techniques to make the model decisions more transparent and trustworthy.
- Real-Time Detection Pipeline: Develop a scalable pipeline capable of classifying news articles in real time, including web scraping and live updates.

By addressing these areas, the system could be evolved into a highly reliable and general-purpose fake news detection framework adaptable to the dynamic nature of online information.