



Fake News Detection Using Recurrent Neural Networks (LSTM)

Abstract: The report explores the use of recurrent neural networks (LSTM). The model was trained on publicly available dataset and evaluated using common metrics such as accuracy and precision.

Written by Muhammad Alfatih Arrazy
As an assignment for 31005 Machine Learning

Introduction

Problem Statement

In recent years, the rapid rise of the internet and social media has made information become more accessible than before. While a good thing for the abundance of information we can now access, it has also made false news and misinformation spread at an alarming rate. Fake news is defined as news made as false or misleading information. This can have serious consequences such as influencing public opinion, political outcomes, and even public health behaviors.

For example, during political elections, fake news can sway voters by the spread of false information regarding candidates, policies, or events. Similarly, during the COVID-19 pandemic, misinformation about vaccines and treatments led to widespread confusion, which caused harm to the general health as studies found it can cause panic, fear, depression, and fatigue (Rocha et al., 2021). With the sheer amount of information published daily on multiple platforms, it is extremely challenging for the average person to verify the authenticity of everything they see on a daily basis.

Importance of the Problem

The problem of fake news can create risk on multiple sectors:

1. Political Impact
Fake news distorts democratic process by spreading misinformation about candidates, policies, and voting procedures.
2. Public Health
During health crises, such as the COVID-19 pandemic, fake news can spread harmful misinformation regarding treatments, vaccines, and prevention methods. This could lead to people refusing to take vaccines, unsafe behavior, and public panic.
3. Financial Market
Misleading information about companies, industries, or the economy could affect stock prices, market value, and investor behavior.
4. Social Impact
Fake news often exploits social divisions such as race and gender, which would lead to increased unrest and division.

Objective of the Project

Key objectives of the project would be:

1. Build an accurate classification model
Building a model that distinguishes fake and real news with a relatively high accuracy.
2. Evaluate model performance
Assess the model's performance using standard evaluation metrics such as accuracy, precision, recall, and F1 score to determine its effectiveness in detecting fake news.
3. Understand the linguistic patterns of fake news
By analyzing the model's predictions, the project aims to identify common linguistic

patterns or features that are indicative of fake news, such as sensationalism, emotional language, or specific grammatical structures.

Data Collection

Source of Data

For this project, I will be using the Fake News Detection Dataset available on Kaggle. This dataset includes a large collection of news articles that are labeled as either real or fake, which will be used to train and evaluate the machine learning model for fake news detection. The dataset provides a robust starting point for building a model to classify news articles across various topics.

From the dataset, there are two separate files. One of them is filled with fake news, while the other one is filled with real news. We will load both files and give them labels to distinguish fake and real news, and then merge them into the same data file.

Dataset Description

Both dataset is structured as follows:

- title: The headline or title of the news article.
- subject: The author of the article (may have missing or null values).
- text: The body of the news article, which will be the primary feature for classification.
- date: The date that the article was published

The only difference between the two of them is one has real news while the other has fake news.

Data Preprocessing

To prepare the data for the machine learning model, we will go through several preprocessing steps. These include combining the data, cleaning the text data, handling missing values, and converting the text into a format suitable for machine learning.

Merging the Dataset & Label Preparation

I will load both files, assign a new column titled 'label' to each dataset (1 for fake news, 0 for real news), and then concatenate them into a single dataset. After that, I shuffled the dataset so it's all positioned randomly. Then, I decided to only use 1000 rows of the data to make the training relatively fast, as the main dataset has around 44000 rows which would take quite a while to use. Lastly, I dropped all rows that had empty text.

```

import pandas as pd

# Load both sets
fake_news = pd.read_csv('/content/drive/MyDrive/News _dataset/Fake.csv')
real_news = pd.read_csv('/content/drive/MyDrive/News _dataset/True.csv')

# 1 is label for fake news, while 0 is label for real news
fake_news['label'] = 1
real_news['label'] = 0

# Combine the datasets
data = pd.concat([fake_news, real_news], ignore_index=True)

# Shuffle the data and randomly take 600 rows
data = data.sample(n=1000).reset_index(drop=True)

data = data.dropna(subset=['text'])

# Check the combined dataset
print(data.head())
print(data.tail())

```

Now the dataset will contain both the real and fake datasets, instead of being separated into two files.

Text Cleaning

The text and title will be cleaned to remove noise and prepare the data for vectorization. This process includes lower casing the text, removing punctuation and special characters, removing stop words, and then tokenizing the text into individual words.

```

# Import necessary libraries for text cleaning
import re
import nltk
from nltk.corpus import stopwords

# Download stopwords
nltk.download('stopwords')
stop_words = set(stopwords.words('english'))

# Define the cleaning function
def clean_text(text):
    text = text.lower() # Lowercase text
    text = re.sub(r'\W+', ' ', text) # Remove special characters
    text = ' '.join([word for word in text.split() if word not in stop_words]) # Remove stop words
    return text

# Apply cleaning to text
data['clean_text'] = data['text'].fillna('').apply(clean_text)

```

Importing libraries such as re and nltk (Natural Language Toolkit) and using NLTK stopwords library. We can make a function that remove stopwords

Data Splitting

```
# Split data into training and testing sets
X = data['clean_text']
y = data['label']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Using sklearn's library we can split the data for training and testing purposes. As we have quite a lot of data, I decided to use an 80-20 split. This ensures we have a big enough volume for data to train and to test with.

Text Vectorization

To make the model able to read the text and compare them, we needed to convert the cleaned text into numerical representations.

Tokenization and Padding

The text was tokenized into sequences of integers using Keras's **Tokenizer**, and the sequences were padded to ensure uniform input size. Each word is assigned a unique integer index, and sequences shorter than the maximum length were padded with zeros.

```
# Import the tokenizer and pad_sequences functions from Keras
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

# Tokenize the text data
MAX_WORDS = 10000 # Maximum number of words in the vocabulary
MAX_SEQ_LEN = 300 # Maximum length of input sequences
tokenizer = Tokenizer(num_words=MAX_WORDS)
tokenizer.fit_on_texts(X_train)

# Convert text to sequences and pad them to ensure equal length
X_train_seq = tokenizer.texts_to_sequences(X_train)
X_test_seq = tokenizer.texts_to_sequences(X_test)
X_train_pad = pad_sequences(X_train_seq, maxlen=MAX_SEQ_LEN)
X_test_pad = pad_sequences(X_test_seq, maxlen=MAX_SEQ_LEN)
```

Word2Vec Embedding

We used a Word2Vec model to train word embeddings on the training data. Word embeddings represent words as dense vectors in a continuous space, capturing semantic relationships between words. These embeddings were used to initialize the embedding layer in the LSTM model.

```

# Import gensim for Word2Vec model training
from gensim.models import Word2Vec

# Tokenize sentences into lists of words
sentences = [text.split() for text in X_train]

# Train Word2Vec model
word2vec_model = Word2Vec(sentences, vector_size=100, window=5, min_count=1, workers=4)
word2vec_model.train(sentences, total_examples=len(sentences), epochs=10)

import numpy as np

# Create an embedding matrix for the words in the tokenizer's vocabulary
EMBEDDING_DIM = 100 # Must match the Word2Vec vector size
word_index = tokenizer.word_index
embedding_matrix = np.zeros((len(word_index) + 1, EMBEDDING_DIM))

for word, i in word_index.items():
    if i < MAX_WORDS:
        try:
            embedding_vector = word2vec_model.wv[word]
            embedding_matrix[i] = embedding_vector
        except KeyError:
            embedding_matrix[i] = np.zeros(EMBEDDING_DIM)

```

Model Architecture

Introduction to the model

For the sake of fake news detection, Long Short-Term memory is used, a type of recurrent neural network. LSTMs are particularly useful for analyzing sequential data, such as text, where the order of words can carry important context.

In this project, we aim to classify news articles into one of two classes:

1. Fake News (label: 1)
2. Real News (label: 0)

We chose an LSTM because it can:

- Capture dependencies between words in a sequence (e.g., how certain words influence the meaning of the sentence).
- Maintain information over long sequences (useful for analyzing long news articles).

LSTM Model Overview

The model architecture includes:

- Embedding Layer: Initialized with the pre-trained Word2Vec embeddings.
- Bidirectional LSTM Layer: Captures relationships between words in both forward and backward directions.

- Dropout Layer: Used to prevent overfitting by randomly dropping units during training.
- Dense Output Layer: Uses a sigmoid activation function for binary classification (real or fake).

Model Architecture

This is the architecture of the model based on the overview above

```
# Import necessary layers from Keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense, Dropout, Bidirectional

# Build the LSTM model
model = Sequential()

model.add(Embedding(input_dim=embedding_matrix.shape[0], # Match input_dim to embedding matrix size
                    output_dim=EMBEDDING_DIM,
                    input_length=MAX_SEQ_LEN,
                    weights=[embedding_matrix],
                    trainable=False)) # Set to False to not modify the Word2Vec weights

# 2. Bidirectional LSTM layer
model.add(Bidirectional(LSTM(128, return_sequences=False)))

# 3. Dropout layer to prevent overfitting
model.add(Dropout(0.5))

# 4. Dense output layer with sigmoid activation for binary classification
model.add(Dense(1, activation='sigmoid'))

# Compile the model
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```

1. Embedding Layer

- Purpose: The embedding layer converts each word in the input sequence (represented by an integer) into dense vectors of fixed size. This allows the model to represent words as vectors in a continuous space, capturing relationships between words based on context and meaning.
- Input: The input to this layer is a sequence of word indices, where each word in a sentence is mapped to a unique integer. The shape of the input is (batch_size, MAX_SEQ_LEN), where MAX_SEQ_LEN is the maximum length of the padded sequences.
- Output: The output is a 2D tensor with shape (batch_size, MAX_SEQ_LEN, EMBEDDING_DIM), where EMBEDDING_DIM is the dimension of the embedding vector (in this case, 100).
- Why it's used: Instead of feeding raw word indices into the model, word embeddings provide a more compact, semantically meaningful representation of words. We use pre-trained Word2Vec embeddings to leverage relationships between words learned from a large corpus.

2. Bidirectional LSTM Layer

- Purpose: The LSTM layer is designed to process sequences of data and capture dependencies between elements in the sequence. The bidirectional LSTM processes

the input in both forward and backward directions, allowing the model to capture context from both past and future words in the sequence.

- Input: The input to this layer is the sequence of word embeddings, with shape (batch_size, MAX_SEQ_LEN, EMBEDDING_DIM).
- Output: The output is a 2D tensor with shape (batch_size, 128), where 128 is the number of LSTM units. Since the LSTM is bidirectional, it combines the information from both directions (forward and backward).
- Why it's used: LSTMs are effective at learning long-term dependencies in sequences, which is important for understanding the meaning of sentences. The bidirectional LSTM improves performance by considering the sequence in both directions, helping the model learn context from both preceding and succeeding words.

3. Dropout Layer

- Purpose: The dropout layer is a regularization technique used to prevent overfitting. It randomly sets a fraction of the input units to 0 during training to prevent the model from becoming too reliant on specific features.
- Input: The input to this layer is the output from the LSTM layer, with shape (batch_size, 128).
- Output: The output has the same shape as the input (batch_size, 128), but with randomly set values dropped (i.e., set to 0) during training.
- Why it's used: Dropout reduces overfitting by preventing the model from becoming too specialized on the training data. It helps the model generalize better to unseen data.

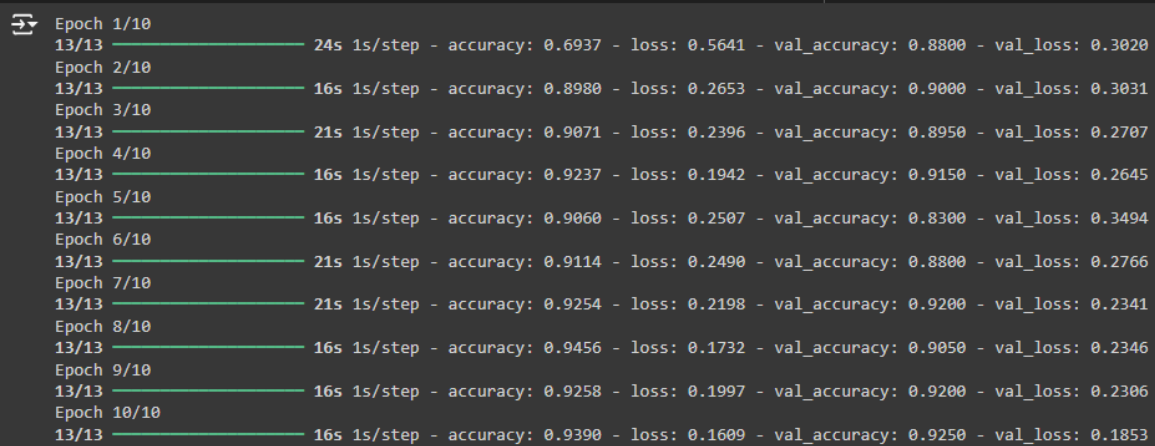
4. Dense Output Layer

- Purpose: The dense layer is a fully connected layer that takes the output of the LSTM and produces a single probability value, which represents the likelihood that the news article is fake. The sigmoid activation function is used to map the output to a value between 0 and 1 for binary classification.
- Input: The input to this layer is the output from the dropout layer, with shape (batch_size, 128).
- Output: The output is a 1D tensor with shape (batch_size, 1), representing the probability of the news article being classified as fake (1) or real (0).
- Why it's used: The dense layer is responsible for making the final prediction. The sigmoid activation function is used in binary classification tasks to produce a probability score between 0 and 1.

Model Training and Evaluation

Model Training

```
[ ] history = model.fit(X_train_pad, y_train, epochs=10, batch_size=64, validation_data=(X_test_pad, y_test))
```



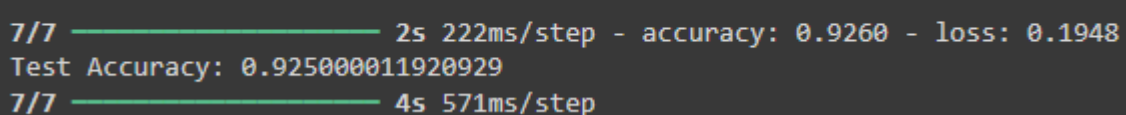
Epoch	Progress	Time	Accuracy	Loss	Val Accuracy	Val Loss
1/10	13/13	24s	0.6937	0.5641	0.8800	0.3020
2/10	13/13	16s	0.8980	0.2653	0.9000	0.3031
3/10	13/13	21s	0.9071	0.2396	0.8950	0.2707
4/10	13/13	16s	0.9237	0.1942	0.9150	0.2645
5/10	13/13	16s	0.9060	0.2507	0.8300	0.3494
6/10	13/13	21s	0.9114	0.2490	0.8800	0.2766
7/10	13/13	21s	0.9254	0.2198	0.9200	0.2341
8/10	13/13	16s	0.9456	0.1732	0.9050	0.2346
9/10	13/13	16s	0.9258	0.1997	0.9200	0.2306
10/10	13/13	16s	0.9390	0.1609	0.9250	0.1853

The model was trained for 10 epochs using the training data. The function calculates the training loss and accuracy and the validation loss and accuracy.

Model Evaluation

```
# Evaluate the model
test_loss, test_accuracy = model.evaluate(X_test_pad, y_test)
print(f'Test Accuracy: {test_accuracy}')
```

```
# Generate predictions
y_pred = (model.predict(X_test_pad) > 0.5).astype("int32")
```

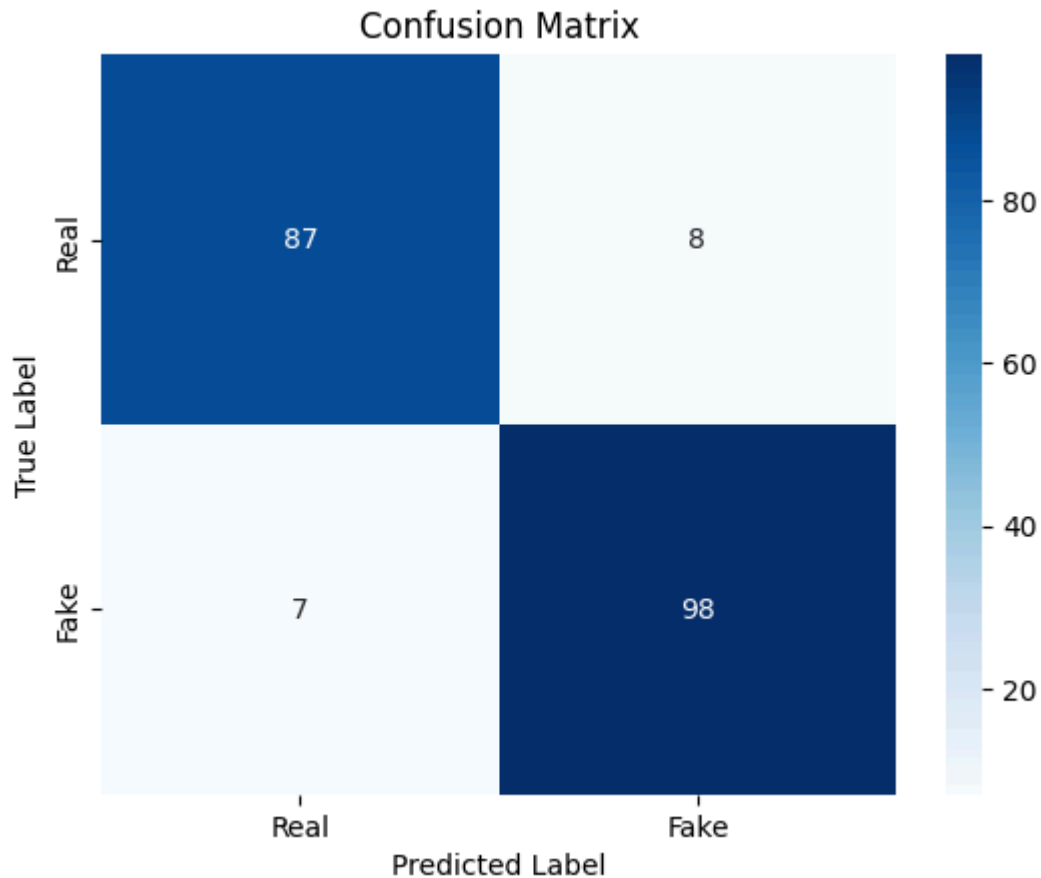


Progress	Time	Accuracy	Loss
7/7	2s	0.9260	0.1948
Test Accuracy: 0.925000011920929			
7/7	4s		

The model's performance was evaluated using the test set. The accuracy and loss were reported, along with a confusion matrix and classification report.

Confusion Matrix

A confusion matrix was used to visualize the model's performance in terms of true positives, true negatives, false positives, and false negatives.



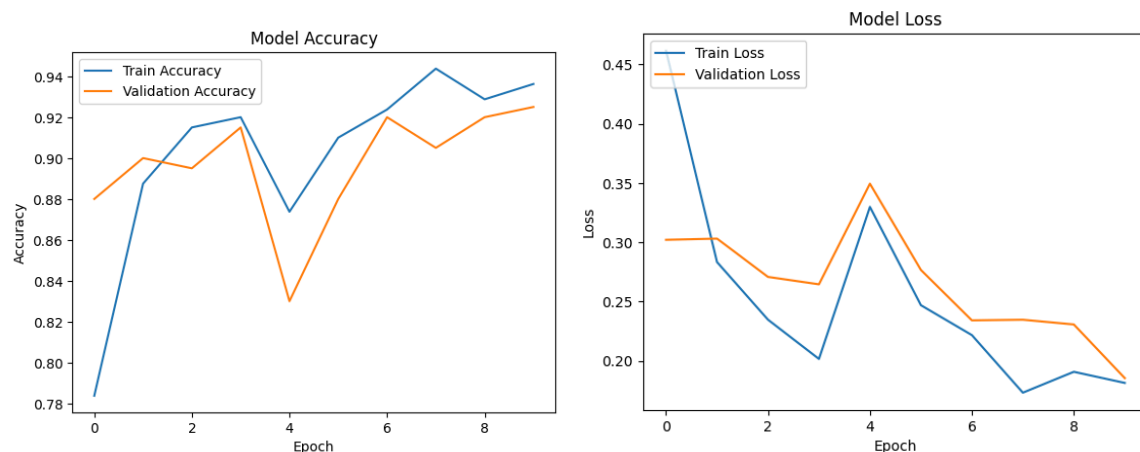
Based on the image, the model correctly predicted 87 real news and incorrectly predicted 8 fake news to be real. The model also correctly predicted 98 fake news and incorrectly predicted 7 fake news to be real.

Classification

	precision	recall	f1-score	support
Real	0.93	0.92	0.92	95
Fake	0.92	0.93	0.93	105
accuracy			0.93	200
macro avg	0.93	0.92	0.92	200
weighted avg	0.93	0.93	0.92	200

Out of all the news the model predicted as real, 93% of them were real, while out of all the news predicted as fake, 92% of them were fake. F1-Score is good for both classes, indicating the model is balanced between precision and recall. There is no significant imbalance in the dataset, as the support values for real news (95) and fake news (105) are quite close.

Model Accuracy over multiple epochs



The two graphs shown above are the model accuracy and loss over 10 epochs. The model accuracy and validation accuracy seem to be learning well as both accuracy at the end was extremely high. Noticeable dip during epoch 4, however it is of no issue as the model seemed to recover after. The model loss seems to also be going downwards in the graph with a noticeable increase in epoch 4.

Conclusion and Improvements

Conclusion

In this report, LSTM (Long Short-Term Memory) is successfully used to create a working model to detect fake news using pre-trained Word2Vec embeddings for word processing. The dataset was preprocessed to clean and tokenize the text, followed by converting it into numerical representations for the model to actually recognize it.

The LSTM model demonstrated strong performance in classifying fake news, achieving an accuracy of around 93%. Additionally, the precision and recall scores for both real and fake news were also high, indicating that the model effectively identifies both fake news and real news with balanced performance. The F1-scores of 0.92 for real news and 0.93 for fake news show a good balance between precision and recall, meaning that the model does not sacrifice one metric for the other.

While there were some fluctuations in learning during epoch 4, the model is able to recover and stabilize, suggesting that the model can be able to predict unseen data. The overall alignment between training and validation accuracy indicates that the model is not overfitting and is capable of handling the data efficiently.

In summary, the combination of Word2Vec embeddings and LSTM architecture proved to be a powerful approach for fake news detection, achieving high accuracy and consistent performance across key evaluation metrics. With further tuning, such as adjusting the learning rate or implementing early stopping, the model's performance could be further optimized. This model serves as a strong foundation for identifying fake news and could be extended to broader applications with larger and more diverse datasets.

Improvements

While this machine learning model can classify with a high degree of accuracy if a piece of news is explicitly fake or not, a lot of news articles aren't fully black and white. These news are filled with half-truths or real and truthful statistics, but it is skewed to make people interpret the news in a certain way. This is more prominent in politics as what a political figure would say or do can be interpreted in multiple ways.

For this purpose, it is important to make improvements to categorize the news articles into different categories. As an example, PolitiFact splits their categories into: Pants On Fire (a lie so blatant it baffles them), False (Fake News/Statements), Mostly false (While there is some truth in it, it is still mostly misleading), Half true (The statement is technically true, but requires context), Mostly true (Some small lies mixed in, but mostly true), and True (Real news). This puts fake news into different categories, which makes it easier for people to differentiate them.

References

Rocha, Y. M., de Moura, G. A., Desidério, G. A., de Oliveira, C. H., Lourenço, F. D., & de Figueiredo Nicolete, L. D. (2021). The impact of fake news on social media and its influence on health during the COVID-19 pandemic: a systematic review. *Journal of Public Health*, 1(10). <https://doi.org/10.1007/s10389-021-01658-z>

Emine Bozkuş. (2022). Fake News Detection Datasets.
<https://www.kaggle.com/datasets/emineyetm/fake-news-detection-datasets>