Fake-News Classifier

add a short subtitle

Yocheved Ofstein, Shay Gali, Shalom Ofstein

January 2025

Abstract

This project aims to develop a machine learning model that classifies news articles as either "true" or "fake," focusing on identifying misleading or false information. Framed as a binary classification task, the model relies on textual content for the classification. A variety of models, from simple baseline classifiers to advanced deep learning techniques, were implemented and evaluated. Performance was assessed using classification metrics such as accuracy, precision, recall, and F1-score. The results show that complex models significantly outperform the baseline, emphasizing the value of advanced architectures in tackling the challenge of fake news detection.

1 Introduction

1.1 Motivation

The goal of this project is to develop a machine learning model that classifies news articles as either true or fake. By analyzing textual features, the model aims to predict the veracity of news content, framing the problem as a binary classification task. The objective is to differentiate between accurate news and misleading or false information.

1.2 Dataset Description

The dataset used in this project consists of labeled news articles categorized as "true news" and "fake news." It includes two sources, one for fake news articles and one for true news. Although the dataset contains additional features such as title, subject category, and publication date, the model primarily uses the cleaned and processed text content for classification. The text data is preprocessed through tokenization and stop word removal. For numerical representation, word2vec embeddings are applied, transforming the text into dense vectors that capture semantic relationships between words.
The dataset is split into 70% for training and 30% for testing, with 20% of the training data reserved for validation to ensure effective model tuning.

1.3 Model Frameworks

The models employed in this study range from simple approaches, such as a majority class classifier (serving as the baseline), to more sophisticated techniques like logistic regression, fully connected neural networks, and advanced deep learning models. Each model's performance is evaluated using standard classification metrics, including accuracy, precision, recall, and F1-score, to compare their effectiveness in distinguishing between true and fake news.

2 Baseline Model

A baseline model serves as a simple starting point for analysis, providing a reference for evaluating the performance of more complex models. For this project, the baseline was implemented using the Majority Class Classifier approach. This method predicts the majority class in the dataset for all test samples, offering a straightforward benchmark for model performance.

2.1 Majority Class Classifier Approach

The Majority Class Classifier was implemented by calculating the distribution of "true news" and "fake news" articles in the training set. The class with the higher frequency, "true news" in this case, was selected as the majority class. As a result, the baseline model predicted "true news" for all test samples.

2.2 Evaluation Metrics

The baseline model was evaluated using three key classification metrics: accuracy, precision, and recall. These metrics help provide a comprehensive understanding of the model's performance:

- **Accuracy**: The baseline model achieved an accuracy of approximately 51.46%, indicating that slightly more than half of the test samples belonged to the majority class.
- **Precision**: The precision for the "true news" class was 51.46%, as all predictions were for this class.
- **Recall**: The recall for the "true news" class was 100%, as all "true news" samples were correctly identified. However, the recall for the "fake news" class was 0%, as the baseline model made no predictions for this class.

The performance of the Majority Class Classifier highlights its limitations, particularly in addressing class imbalance and its inability to correctly identify instances of the minority class.

3 Logistic Regression Model

To build a more sophisticated classifier, logistic regression was implemented to classify news articles as "true" (1) or "fake" (0). The following subsections outline the preprocessing steps, model training, and evaluation.

3.1 Model Training

The logistic regression model was trained on the preprocessed training data using 1,000 maximum iterations to ensure convergence. The model learns to assign weights to features, enabling it to predict the likelihood of a news article belonging to either the "true" or "fake" category.

3.2 Initial Attempts and Corrections

The initial implementation of the Logistic Regression model resulted in an accuracy metric of 100%, which was deemed highly improbable given the complexity of the classification task. A review of the data and preprocessing steps revealed that all the "true news" articles contained the term "Reuters" at the beginning or early in the text.
This unintended feature created a significant bias in the dataset, enabling the model to classify news articles based on the presence of the term "Reuters" rather than learning meaningful patterns from the text content.
To resolve this issue, the preprocessing step was updated to remove all occurrences of "Reuters" from the text data. After this adjustment, the model was re-trained, and the accuracy dropped to a more realistic value. This correction underscored the importance of robust preprocessing techniques to eliminate biases and ensure that models rely on meaningful features for classification.

3.3 Validation Results

The model was evaluated on the validation set, achieving a high accuracy of 97.83%. Detailed performance metrics are presented below:

- **Accuracy**: 0.98 for both "true" and "fake" classes.
- **Precision**: 0.98 for both "true" and "fake" classes.
- **Recall**: 0.98 for both "true" and "fake" classes.

3.4 Comparison with Baseline

Unlike the baseline model, which only predicted the majority class (”true news”), the logistic regression model effectively captured patterns in the text data and provided accurate predictions for both ”true” and ”fake” news articles.

Metric	Baseline Model	Logistic Regression
Accuracy	51.46%	97.83%
Precision (True News)	51.46%	97.83%
Recall (True News)	100%	97.83%
Precision (Fake News)	N/A	97.83%
Recall (Fake News)	0%	97.83%

Table 1: Comparison of Baseline Model and Logistic Regression Model.

4 Fully Connected Neural Network

4.1 Comparison with Previous Models

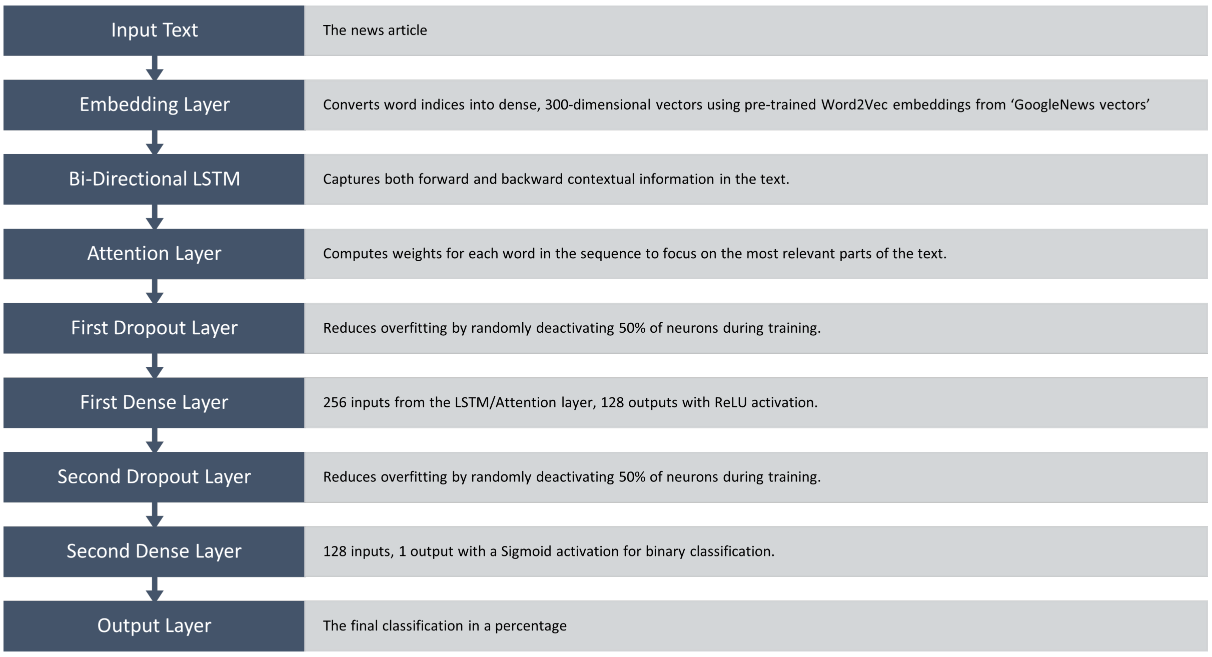
5 Advanced Model

5.1 Introduction

Here we presents an advanced Long Short-Term Memory (LSTM) model Leveraging bidirectional LSTM layers, attention mechanisms, and dropout regularization. we will present the model architecture, training methodology, and evaluation results.

5.2 Advanced Model Architecture

The proposed model consists of the following key components:



Input Text	The news article
Embedding Layer	Converts word indices into dense, 300-dimensional vectors using pre-trained Word2Vec embeddings from ‘GoogleNews vectors’
Bi-Directional LSTM	Captures both forward and backward contextual information in the text.
Attention Layer	Computes weights for each word in the sequence to focus on the most relevant parts of the text.
First Dropout Layer	Reduces overfitting by randomly deactivating 50% of neurons during training.
First Dense Layer	256 inputs from the LSTM/Attention layer, 128 outputs with ReLU activation.
Second Dropout Layer	Reduces overfitting by randomly deactivating 50% of neurons during training.
Second Dense Layer	128 inputs, 1 output with a Sigmoid activation for binary classification.
Output Layer	The final classification in a percentage

Figure 1: The Advanced Model Architecture

5.2.1 Embedding Layer

Purpose: Converts word indices into dense, 300-dimensional vectors using pre-trained Word2Vec embeddings.

Why Needed: Raw text cannot be directly used as input to machine learning models because they require numerical data. Embeddings provide a way to represent words as dense numerical vectors, capturing semantic relationships (e.g., "king" and "queen" have similar vectors).

Trainable: Yes, allowing fine-tuning of embeddings during training.

Input Shape: [Batch Size, Sequence Length].

Output Shape: [Batch Size, Sequence Length, Embedding Dimension (300)].

5.2.2 Bidirectional LSTM Layer

Purpose: Captures both forward and backward contextual information in the text.

Why Needed: Language is inherently sequential, and understanding a word often depends on its context (both before and after). Bidirectional LSTMs process the sequence in both directions, making it possible to consider future and past words simultaneously.

Hidden Units: 128 units in each direction, resulting in a total of 256 outputs.

Input Shape: [Batch Size, Sequence Length, Embedding Dimension (300)]

Output Shape: [Batch Size,256]

The following diagram presents the functionality of the Bi-Directional LSTM Layer:

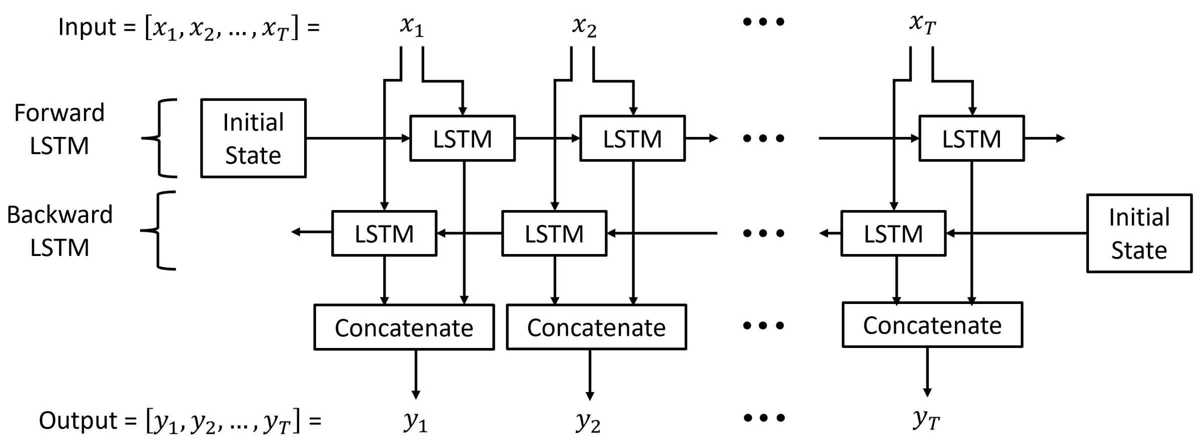


Figure 2: Bi-LSTM

5.2.3 Attention Mechanism

Purpose: Computes weights for each word in the sequence to focus on the most relevant parts of the text.

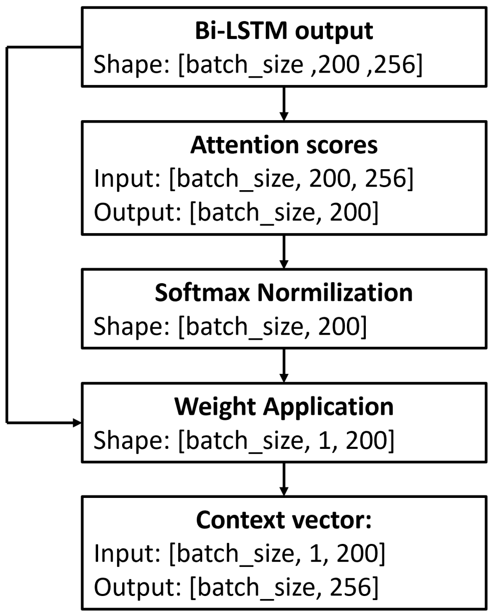
Why Needed: Not all words in a sentence contribute equally to the meaning. For example, in the sentence "The weather is beautiful, but it might rain later," the word "rain" is more important for predicting the sentiment. Attention allows the model to assign higher importance to relevant words.

Implementation: A linear layer computes attention scores, normalized using the softmax function.

Input Shape: [Batch Size,256]

Output Shape: [Batch Size,256]

The following diagram presents the Attention Layer Mechanism :



Bi-LSTM output
Shape: [batch_size ,200 ,256]
Attention scores
Input: [batch_size, 200, 256]
Output: [batch_size, 200]
Softmax Normilization
Shape: [batch_size, 200]
Weight Application
Shape: [batch_size, 1, 200]
Context vector:
Input: [batch_size, 1, 200]
Output: [batch_size, 256]

Figure 3: Attention Layer Mechanism

5.2.4 Dropout Regularization

Purpose: Reduces overfitting by randomly deactivating 50% of neurons during training.

Why Needed: Without dropout, the model may become too specialized to the training data and fail to generalize to unseen data. Dropout forces the network to learn more robust features by introducing randomness.

Input Shape: Matches the input shape of the layer where it is applied.

Output Shape: Matches the output shape of the layer where it is applied.

5.2.5 Fully Connected Layers

Structure:

- First Layer: 256 inputs from the LSTM/Attention layer, 128 outputs with ReLU activation.
- Second Layer: 128 inputs, 1 output with a Sigmoid activation for binary classification.

Purpose: Maps the high-dimensional features from the LSTM/Attention layers into a single output

Why Needed: After processing the sequential data, fully connected layers act as a classifier.

The following diagram presents the Dense Layers and the Dropout Layers as implemented in the Model:

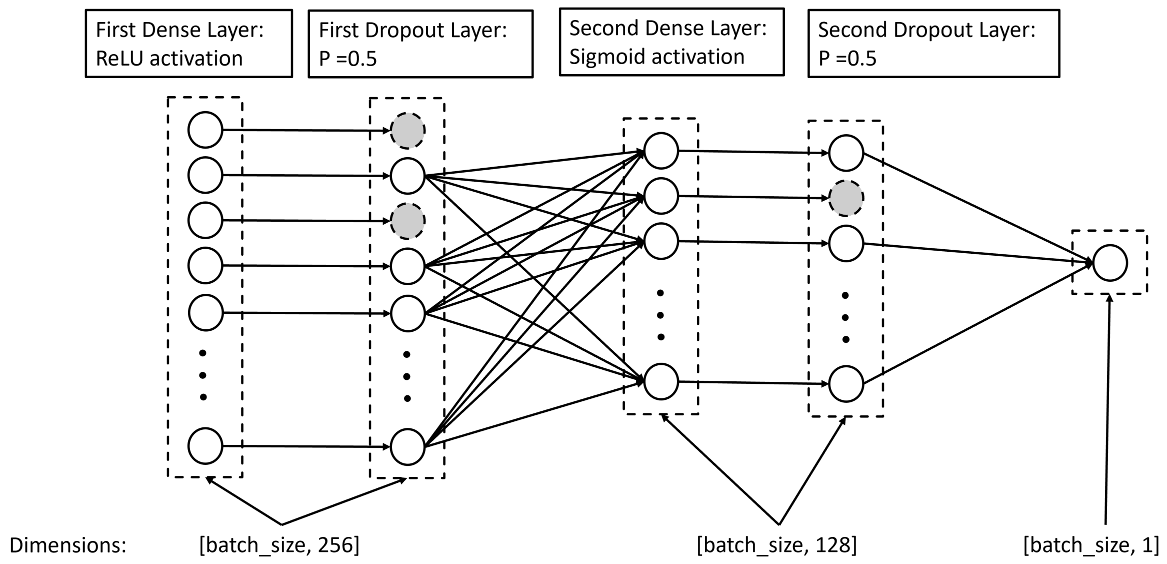

First Dense Layer: ReLU activation
First Dropout Layer: P =0.5
Second Dense Layer: Sigmoid activation
Second Dropout Layer: P =0.5
Dimensions:
[batch_size, 256]
[batch_size, 128]
[batch_size, 1]

Figure 4: The Dense and Dropout Layers

5.3 Training Methodology

5.3.1 Loss Function

To evaluate the Loss we use Binary Cross-Entropy Loss (BCELoss) as the Loss Function. BCELoss Measures the divergence between predicted probabilities and true binary labels. Formula:

$$Loss = -\frac{1}{N}\sum_{(i=1)}^{N}[(y_i)\cdot\log(p_i) + (1 - y_i\ )\cdot\log(1 - p_i)]$$

This loss function is especially suitable for binary classification tasks.

5.3.2 Optimizer

Adam Optimizer: Adaptive optimization algorithm with a learning rate of 0.001.

5.3.3 Training Process

Batch Size: 32 samples per batch.

Epochs: 10 iterations over the entire training dataset.

Validation: The model was evaluated in the validation set after each epoch.

5.4 Results

Accuracy: 0.9578
Precision: 0.9621
Recall: 0.9457
F1-Score: 0.9538