# Project Report : CS 5660
# Fake News Detection Using LSTM in TensorFlow and Python

Kavya Reddy Biradavolu

Harika Yarramalli

Shagun S Lokre

Varun Tiwari

## Abstract

*The use of neural networks based on LSTM for generating false news is investigated in this study, using TensorFlow and Python. The purpose of the study was to understand capabilities and potential risks associated with content generation based on AI powered technologies. An overview of the LSTM architecture is presented in an abstract, describing its suitability to sequence data modeling and how it can be used for long range dependencies. Discussions are being held on the implementation process, including data preprocessing and relevance of a diverse training dataset. For assessing the credibility and authenticity of produced articles, an evaluation is carried out with quantitative metrics as well as a user study. The findings of the study conclude that there is a need to improve detection tools, public awareness and ethical protection for combating artificial intelligence generated fake news. In general, the research shed light on how advanced language models affect society and emphasizes responsibility for developing artificial intelligence.*

## 1. Introduction

In the digital age, the prevalence of fake news has become a major issue, with potentially serious repercussions for individuals, communities, and democracy. The development of sophisticated machine learning algorithms has enabled the creation of powerful tools that can generate false news articles that appear to be genuine. In this paper, we explore the field of fake news generation using the LSTM architecture, which is implemented in the widely used deep learning framework TensorFlow and the Python programming language. In order to effectively detect fake news, one of the most important aspects of the detection process is the prior processing and representation of the textual data. In the past, the detection of fake news has been limited to the use of basic feature engineering techniques, or the use of a bag of words representation, which does not adequately capture the semantic and context-sensitive nature of the words. In order to overcome these shortcomings, we use vectorization techniques, such as the word2vec, which provides a more sophisticated and context-sensitive representation of the words.

Our goal in this paper is to show that vectorization with word2vec can be used for the detection of fake news using LSTM in TensorFlow and python. We look at preprocessing steps that clean and prepare the training dataset and emphasize the importance of using diverse and representative data sets to improve the detection performance of the model. We expect the model to be able to distinguish fake news articles from real ones more accurately and precisely by taking into account the semantic relationships that are captured by word2vec. This research helps to create more robust and efficient fake news detection solutions, helping to combat misinformation and promote information integrity in today's digital world.

## 2. Prerequisite

*Vectorization Word2Vec*: Word2Vec is one of the most widely used word embedding algorithms in NLP. Word embedding with Word2vec is the process of converting words into numerical vectors in continuous vector space and capturing semantic meaning and contextually related relationships between words. Word embeddings are word representations that are trained on a large set of text data using the Word2Vec algorithm. Word2Vec learns word representations based on a distributional hypothesis. The distributional hypothesis states that words that appear in the same context are likely to have the same meaning. Word2Vec predicts the likelihood of a word relative to its adjacent words in the given context window[1]. Word2Vec then assigns a vector to every word in the dictionary after it has been trained. Word vectors are high dimensional representations where the position of the vectors contains information about the relationship between the words.

Words that have similar meanings or appear in similar contexts usually have similar word representations. Therefore, vectorization with word2vec allows words with similar meanings to be spaced closer in the space of vectors.

Word2vec has the advantage of capturing semantic information and preserving semantic relationships between words, making it useful for various Natural Language Processing (NLP) tasks, such as the detection of fake news. By converting words into vectors, models can take advantage of semantic similarities between sentences and documents to gain a better understanding of their context and meaning. All in all, Word2 vectorization with vectorization is a tool that converts words into numbers, helping machine learning models effectively process and analyze text data. It helps to capture semantic relationships as well as contextual information, making NLP models more accurate and perform better.

*Long-Short Term Memory (LSTM)*: LSTM vectorization is the process of transforming sequential data into an appropriate numerical representation that may be fed into the LSTM for processing or analysis. Long-term memory (LSTM) is a subset of recurrent neural networks (RNNs) that is highly efficient at recognizing long-term relationships in sequential data[2]. When working with time-series or text-based data, LSTM necessitates the use of a vectorized representation. This vectorization involves transforming the sequence data into a fixed-length numerical vector, with each vector representing a specific element or time step in the sequence.

When it comes to LSTM, the main steps involved are tokenization, indexing, sequence conversion, and vectorization. Tokenization is when the text is divided into individual words, or subwords. Indexing is when each word is given its own unique index or an integer representation. Sequence conversion is when each word in the text is converted to a sequence of words, with each word being replaced with its own index from the dictionary. Padding is used to make sure the sequence length is always the same, with special tokens and zeros added to shorter sequences. Finally, vectorization is when the indexed sequences are converted to numerical vectors, which can be inputted into the model. This can be done using things like one-hot coding or word embedding.

The vector representation makes it possible for LSTM models to process the sequence data efficiently. At each step of the sequence, the layer LSTM receives a vector of the same length that represents the current element. The model can learn from the sequence patterns and relationships to make predictions, or to do things like classify, analyze sentiment, or detect fake news. LSTM vectorization is important for processing sequential data because it converts the raw data into a format that LSTM model can manipulate. It allows LSTM model to capture and understand long-term dependencies. This makes LSTM suitable for sequences with time or context information.

## 3. Related Work

In the past, people have suggested using text mining and machine learning to analyze news text to figure out how likely it is to be true. But now, with more computing power and the ability to handle huge datasets, more advanced deep learning models offer better performance than text mining or machine learning. CNN and RNN are popular Deep Neural Network architectures that can be used for a variety of Natural Language Processing (NLP) tasks, like text classifying, rumor and spam detection, sentiment analysis, and more. To figure out if a news article is fake, a simple approach is to use a Bayesian classifier and test it against a bunch of different Facebook news posts. This paper provides an analysis of the performance of four classification algorithms, Support Vector Machines (SVM), Gradient Boosting (GBB), Bounded Decisions Trees (BCDT), and Random Forests (RFT), across a dataset of approximately 11,000 published news articles.

The Neural Network architecture is based on TF-IDF (Term Frequency Inverse Document Frequency) and BOW (Bag-of-Words) representations as inputs to a Multi-Layer Perception (MLP). This architecture is used to detect the stance of a word in news articles. Word embedding is a neural representation of the word as a true-valued vector, which allows for word correlation to be measured. Neural networks display their highest performance in many Natural Language Processing (NLP) tasks when the word embedding has been pre-trained. This related work focuses on automatic fake news detection based on surface-level language patterns, a new neural network that integrates meta-data with text, and a recurrent neural network-based model for dialogue act classification. We'll also look at how context or sequence information can be used in this model. Plus, we'll explore a new recurrent neural network method that learns to recognize microblog events for rumor spotting. Finally, we'll explore how a deep learning approach can be used to analyze aspect-specific sentiment.

## 4. Approach

a) Data Importing and Exploration: Import the labeled dataset containing genuine and fake news articles. This can be in the form of a CSV or JSON file.
   Use libraries such as pandas to read and manipulate the dataset.
   Explore the dataset to understand its structure, check for missing values, and gain insights into the distribution of classes.

b) Text Preprocessing: Clean the text data by removing unwanted characters, punctuation, and special symbols. This can be done using regular expressions or string manipulation techniques.
   Tokenize the text by splitting it into individual words or subwords. This can be achieved using libraries like NLTK

or spaCy.
Remove stop words, which are common words with little semantic meaning, from the tokenized text. This helps reduce noise in the data.
Optionally, apply techniques like stemming or lemmatization to reduce words to their root forms, which can help in normalization.

c) Dataset Splitting: Split the preprocessed dataset into training, validation, and testing sets.
Ensured a balanced distribution of genuine and fake news articles in each set to avoid bias in the model.

d) Word Embedding with Word2Vec: Train a Word2Vec model on the preprocessed training data to learn word embeddings. This step captures the semantic relationships between words.
Convert the preprocessed tokenized text into Word2Vec embeddings, where each word is represented as a high-dimensional vector.
Store the Word2Vec model to be used later for embedding the testing and validation data.

e) Text Vectorization: Convert the tokenized text data into sequences of word indices using the Word2Vec vocabulary. Replace each word in the tokenized text with its corresponding index from the Word2Vec vocabulary.
Perform sequence padding or truncation to ensure a fixed length across all sequences. This is necessary to create consistent input sizes for the LSTM model.

f) LSTM Model Construction: Build an LSTM model using TensorFlow. TensorFlow provides an LSTM implementation through its Keras API.
Create an embedding layer that maps the word indices to their corresponding Word2Vec embeddings. This layer will be the input layer for the LSTM model[3].
Add one or more LSTM layers with a defined number of memory units or hidden dimensions. This captures the sequential information and long-term dependencies in the text data.
Optionally, include additional layers such as dense layers or dropout layers for regularization or to increase the model's capacity.

g) Model Training: Compile the LSTM model by specifying the loss function, optimizer, and evaluation metrics.
Train the model using the preprocessed and vectorized training data. This involves feeding the input sequences and their corresponding labels to the model.
Set the number of epochs (iterations over the training data) and the batch size (number of samples per gradient update).
Monitor the model's performance on the validation set during training and make adjustments as necessary. Early stopping can be implemented to stop training if the validation performance plateaus.

h) Model Evaluation: Evaluate the trained LSTM model on the preprocessed and vectorized testing data. This involves feeding the input sequences to the trained model and comparing the predicted labels with the actual labels.
Calculate metrics such as accuracy, precision, recall, and F1-score to assess the model's performance in detecting fake news.
Fine-tune the model by adjusting hyperparameters, architecture, or training strategies, if necessary, to further enhance the fake news detection performance.

## 5. Results

```
[ ] model.summary()

Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_1 (Embedding)     (None, 1000, 100)         23187300

 lstm_1 (LSTM)               (None, 128)               117248

 dense_1 (Dense)             (None, 1)                 129

=================================================================
Total params: 23,304,677
Trainable params: 117,377
Non-trainable params: 23,187,300
_________________________________________________________________
```

Fig1: Model Summary

```
[ ] model.fit(X_train, y_train, validation_split=0.3, epochs=6)

Epoch 1/6
737/737 [==============================] - 1350s 2s/step - loss: 0.1530 - acc: 0.9423 - val_loss: 0.0910 - val_acc: 0.9688
Epoch 2/6
737/737 [==============================] - 1370s 2s/step - loss: 0.0522 - acc: 0.9834 - val_loss: 0.1411 - val_acc: 0.9482
Epoch 3/6
737/737 [==============================] - 1372s 2s/step - loss: 0.0305 - acc: 0.9899 - val_loss: 0.0311 - val_acc: 0.9897
Epoch 4/6
737/737 [==============================] - 1376s 2s/step - loss: 0.0195 - acc: 0.9938 - val_loss: 0.0269 - val_acc: 0.9923
Epoch 5/6
737/737 [==============================] - 1375s 2s/step - loss: 0.0111 - acc: 0.9971 - val_loss: 0.0319 - val_acc: 0.9898
Epoch 6/6
737/737 [==============================] - 1435s 2s/step - loss: 0.0055 - acc: 0.9986 - val_loss: 0.0215 - val_acc: 0.9938
<keras.callbacks.History at 0x7f4d05d919f0>
```

Fig2: Training and Evaluation built model

```
[ ] print(classification_report(y_test,y_pred))

              precision    recall  f1-score   support

           0       1.00      0.99      0.99      5895
           1       0.99      1.00      0.99      5330

    accuracy                           0.99     11225
   macro avg       0.99      0.99      0.99     11225
weighted avg       0.99      0.99      0.99     11225
```

Fig 3: Metrics Calculation

```
[ ] #TEST USING YOUR OWN HEADLINE FROM ANY SOURCE
    x = ["Former Vice President Mike Pence testified on Thursday to a federal grand jury investigating the aftermath of the 2020 election and the actions
    x = tokenizer.texts_to_sequences(x)
    x = pad_sequences(x, maxlen=maxlen)
    (model.predict(x) >=0.5).astype(int)

    1/1 [==============================] - 0s 97ms/step
    array([[1]])
```

Fig 4: Detecting if the given news is fake or not. Array 1[1] represents that the news is true and array1[0] represents it's a fake news.

## 6. Conclusion

To sum up, LSTM in Python and TensorFlow for fake news detection shows promising results in the detection and classification of fake news articles. LSTM models use the sequential nature of the text data and capture long-term dependencies to capture contextual information to improve the accuracy of the fake news detection.

The implementation process includes Importing and exploring the data, Text preprocessing, Dataset splitting Word embedding, using Word2Vec Text vectorization, LSTM model construction, Training Evaluation Predictive analysis, we can effectively build a robust fake news detection system[4].

To sum up, fake news detection is difficult and there is no one-size-fits-all solution. The LSTM model's performance can be affected by training data quality, hyperparameter selection, and the intricacy of fake news articles. In order to further refine the accuracy of the Fake News detection system, it is suggested to investigate ensemble techniques, which may involve the combination of multiple Localized Stereotyped Machine Learning (LSTM) models or the integration of other machine learning techniques. Furthermore, continuous updates of the model with newly labeled data and the integration of external data sources, such as Credibility Scores or Fact-Checking Databases, can further enhance the performance and flexibility of the model.

| Student Name | Contributed Aspects | Details |
|---|---|---|
| Harika Yarramalli | Data Creation and Implementation | Responsible for data importing, exploration, and preprocessing. Cleans the dataset, performs text preprocessing tasks such as tokenization, removing stop words, and applying stemming or lemmatization techniques. |
| Varun Tiwari | LSTM Model | Builds and trains the LSTM model using TensorFlow and Python. Implements training strategies such as early stopping, regularization techniques, and batch size optimization. |
| Kavya Reddy Biradavolu | Vectorization | Implements Word2Vec for word embedding. Trains the Word2Vec model on the preprocessed data to learn word embeddings. Converts the preprocessed tokenized text into Word2Vec embeddings, mapping each word to a high-dimensional vector. |
| Shagun S Lokre | Analysis | Performs dataset splitting into training, validation, and testing sets. Ensures a balanced distribution of genuine and fake news articles in each set. Conducts exploratory data analysis to gain insights into the dataset and check for any data imbalances or biases. |
| Kavya and Shagun | Metrics Calculation | Evaluates the performance of the LSTM model on the testing set. Calculates evaluation metrics such as accuracy, precision, recall, and F1-score to assess the model's performance. |

Table 1. Contributions of team members.

## References


[1] Bahad, P., Saxena, P. and Kamal, R., 2019. Fake news detection using bi-directional LSTM-recurrent neural network. Procedia Computer Science, 165, pp.74-82.

[2] Kong, S.H., Tan, L.M., Gan, K.H. and Samsudin, N.H., 2020, April. Fake news detection using deep learning. In 2020 IEEE 10th symposium on computer applications & industrial electronics (ISCAIE) (pp. 102-107). IEEE.

[3] Keya, A.J., Afridi, S., Maria, A.S., Pinki, S.S., Ghosh, J. and Mridha, M.F., 2021, August. Fake news detection based on deep learning. In 2021 International Conference on Science & Contemporary Technologies (ICSCT) (pp. 1-6). IEEE.

[4] Fake news detection model using tensorflow in python (2022) GeeksforGeeks. Available at: https://www.geeksforgeeks.org/fake-news-detection-model-using-tensorflow-in-python/