Natural Language Processing

Project Submission

Title: Fake News Detection

Group Number: 20

1. Akansha Gautam	(2016221)
2. Harshita Sahu	(MT20087)
3. Khushboo Bajaj	(MT20060)
4. Divya Pandey	(MT20128)

All the materials used in the project can be found using this link.

- Data Set
- Code Files
- Trained models

Background of the problem statement (What work has been done so far on the problem statement?)

Various works have been done on the problem statement listed as follows:

- In this notebook, the author has used the one-hot representation with LSTM neural network.
- In this notebook, the author has extracted features using TfidfTransformer and then used the ensemble model (`RandomForestClassifier, ExtraTreesClassifier, AdaBoostClassifier`) to train their features.
- In this notebook, the author has used the TF-IDF Vectorizer for feature extraction and (*`PassiveAggressiveClassifier)`* to train their features.
- In this notebook, the author has used the TF-IDF Vectorizer for feature extraction and RandomForestClassifier to train their features.
- In this notebook, the author has used the one-hot representation with LSTM.
- In this notebook, the author has used Wactors and Term Frequency–Inverse Document Frequency Matrices and XGBoost Classifier in this notebook.
- In this notebook, the author has used (*`PassiveAggressiveClassifier)`* with count vectorizer, Logistic Regression with TF-IDF Vectorizer.
- In this notebook, the author has used the one-hot representation with Bidirectional LSTM**.**
- In this notebook, the author has used a count vectorizer with Logistic Regressor with a special tokenizer.
- In this notebook, the author has used a count vectorizer for feature vector and multinomial Naive Bayes as a classifier.
- In this notebook, the author has used TF-IDF Vectorizer for feature extraction and Classifications, Logistic regression, Decision Tree, KNeighbours, Linear Discriminant to train their model.
- In this notebook, the author has used the TF-IDF Vectorizer for feature extraction and XGBoost classifier for training the model in this notebook.
- In this notebook, the author has used the TF-IDF Vectorizer for feature extraction and Logistic Regression Classifier to train their model.

Dataset:- The dataset that we have worked on has been taken from Kaggle. This is the link from where the dataset can be accessed:- https://www.kaggle.com/c/fake-news/data. This dataset contains news articles related to the politics of the United States of America. We have train.csv, test.csv, and submit.csv.The description of the dataset is as follows:-

train.csv: A full training dataset with the following attributes:

- id: unique id for a news article
- title: the title of a news article
- author: author of the news article
- text: the text of the article; could be incomplete
- label: a label that marks the article as potentially unreliable
    - **1: unreliable**
    - **0: reliable**

test.csv: A testing training dataset with all the same attributes at train.csv without the label.

submit.csv: A sample submission.

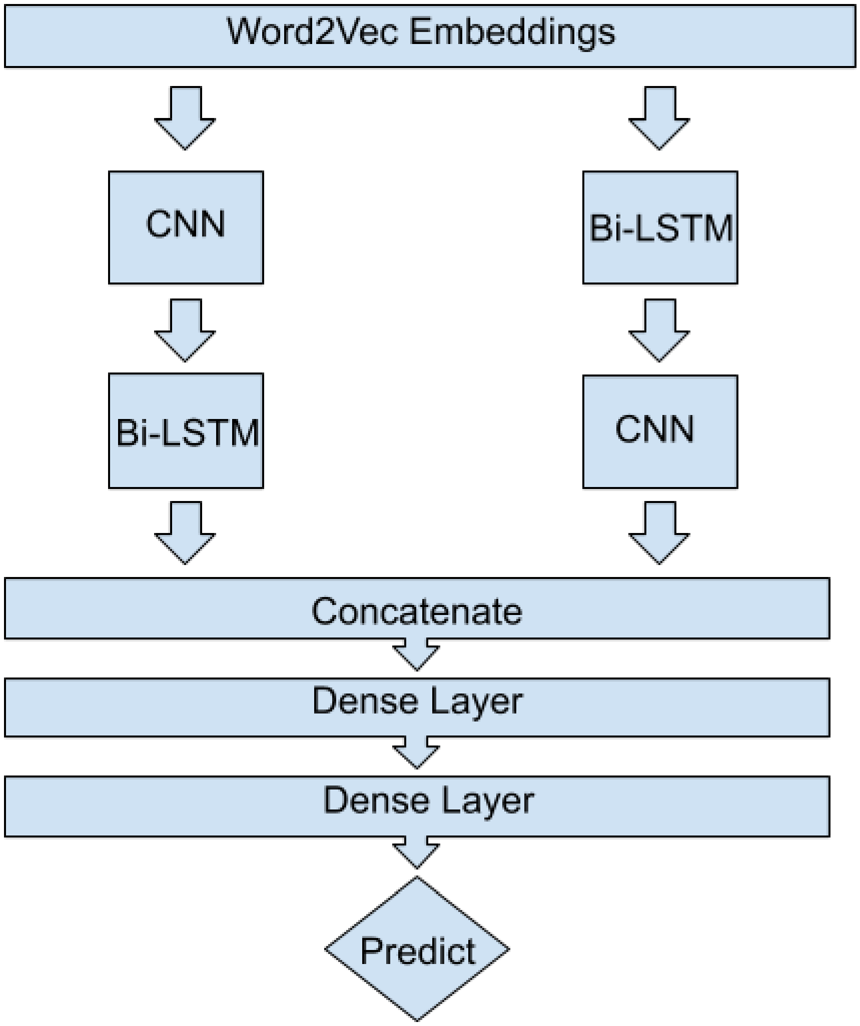
The number of rows in the **train.csv** dataset is 20800, and the number of columns is 5.

The number of rows in the **test.csv** dataset is 5200, and the number of columns is 4.

Methodology

- The experiments conducted in this project used title, author, and text data given in the dataset files.
- The preprocessing steps implemented on both the train and sets are as follows:
    - Fill the NULL entries in the given dataset using empty strings
    - Lowercase
    - Remove stopwords
    - Remove HTTPS links
    - Remove punctuation
    - Remove digits
    - Perform lemmatization
- The embedding vectors have been extracted using the Word2Vec model trained on the corpus. The parameters of the Word2Vec model are as follows:
    - size = 200
    - windows = 2
    - min_count = 1
    - maxlen = 1000

- The proposed method comprises a network that involves an ensemble approach of two deep learning models such as CNN + Bi-LSTM and Bi-LSTM + CNN.



  - The **intuition** behind using the combination of two models such as CNN + Bi-LSTM and Bi-LSTM + CNN is to incorporate two different variants:
    - In CNN + Bi-LSTM model, the convolution layer will extract local features from input word embeddings. Then the Bi-LSTM layer will be able to use the ordering of said features to learn about the input's text ordering.
    - In Bi-LSTM + CNN model, the Bi-LSTM layer generates a new encoding for the original input. The Bi-LSTM layer's output is then fed into a convolution layer to extract local features.
  - The intuition behind using CNN and Bi-LSTM models are as follows:
    - CNN is used to extract salient features (e.g., tokens or sequences of tokens) invariant to their position within the input sequences.

            - To set up CNN, we used a convolutional layer consisting of a set of filters. These filters only use a subset of the input text at a given time but are applied across the full input text by sweeping over it.
        - Bi-directional LSTM is used to capture the previous context and the future context of the input sequence.
- The other details related to this architecture are listed below:
    - The input layer is the first layer that accepts a list of words with a dimension of 200.
    - The pooling layer is used to downsample the output of the previous layer of the network that allows only the valid information to pass, resulting in fewer operations.
    - The purpose of adding a dense layer is to perform a linear operation in which every input is connected to every output layer by weight, followed by a non-linear activation function.
    - Activation functions like ‘relu’ and ‘sigmoid’ are used to add non-linearity to the network.
    - Loss function calculates the error for a single training example. We use `binary_crossentropy` as our problem belongs to binary classification.
    - We use Adaptive Moment Estimation (Adam) to update weight in the model.
    - Our model is trained on 30 epochs with batch size equals 32.
    - Dropout and BatchNormalization Layer added to the sequential model to avoid overfitting on the training data.

Results

Our proposed model achieves an accuracy score of 0.99537 on the 100% test set.

Comparison of methodology and results with previous works

Our proposed method outperformed the previously established works. We compared our results with the top-3 teams available on Kaggle public Leaderboard and top-3 teams available on Kaggle private Leaderboard.

Team name	**Score**
Matt Gallagher	0.98782
Leroy Todd	0.98589
Dylan Rainwater	0.98525
Ours	**0.99615**

Comparison with Public Leaderboard

Team name	**Score**
Leroy Todd	0.98598
Matt Gallagher	0.98379
Matthew LeGate	0.96923
Ours	**0.99478**

Comparison with Private Leaderboard