Natural Language Processing : CSE556

Aditya Jain 2020554
Akansha 2019011
Harsh Goyal 2020562
Vasu Yadav 2019344

1 Problem Definition

False or misleading information presented as news is known as fake news. Most of the time, the goal of fake news is to hurt someone or something's reputation or make money from advertising. Of late, fake news has been in the spotlight of mainstream journalism and the general public because it can affect a country's political scenario. In this project, we attempt to detect the authenticity of specifically political news. Social media is the primary channel for disseminating such content, though it occasionally makes its way into mainstream media as well. Because fake news can significantly impact an election's political outcome, it is becoming increasingly important to identify and classify it as such. The loose definition of "fake news" presents the primary obstacle to resolving the problem. For instance, fake news can be broken down into subcategories: a statement known to be completely false, a speech that presents statistics as facts that have not been thoroughly investigated, or satirical writing. Our main aim of the project is to detect and classify fake news.

2 Related Works

The fake news challenge dataset used by Hanselowski et al. (2018) divides the news into four categories: agree, disagree, discuss, and unrelated. For the embedded token sequence, the model is trained with two stacked LSTMs, and a three-layered neural network is used to estimate the likelihood that it belongs to a particular class. The test predictions in this paper are only obtained using an LSTM-based model. The concept of sentence classification using CNN and max-over-time-pooling is prominently discussed in Kim (2014). They used dropout on the final layer with a weight vector l2-norm constraint for regularization.
For the news statement, a solution proposed by Wang (2017) uses a convoluted neural network and bidirectional LSTM for other news features like the speaker, venue, etc. Convolutional layers, pooling layers, fully connected layers, and normalization layers are typically the hidden layers of a CNN. The statements' syntactic and temporal content is ignored by the aforementioned concepts. For instance, fake news sentences typically have a hard dependency parse due to the abundance of phrases and punctuation. As a result, we have attempted to incorporate these pieces of data into the models' training by employing sentence Dependency parsing and POS tags. We tested these ideas by fact-checking in the training dataset. We have used a subset of the LIAR dataset presented in Wang(2017) as provided in the project. The baseline model presented in the same paper uses LSTM and CNN for training text embeddings and metadata-dense features for training the model. LIAR dataset contains 12.8K manually labeled short statements in various contexts by diverse speakers from POLITIFACT.COM. Details about the dataset is mentioned in Section 3.1. Using LIAR dataset, the task of fake news detection is now framed as a 6-way clas- sification task into fine-grained labels for the truthfulness ratings: pants-fire, false, barely- true, half-true, mostly-true, and true.

3 Methodology

We have used several models of machine learning and deep learning concepts like Multinomial Naive Bayes , CNN and Bi-LSTM classifier.

3.1 Multinomial Naive Bayes

PreProcessing: We first preprocessed the statement column of our input dataset with lowercasing followed by tokenization. Then stopwords are removed from the text and these filtered statements are stemmed and lemmatised.

Implementation: Here we have used a

pipelined model with following configurations:
vectorizer: CountVectorizer
norm: Binalizer
classifier: Multinomial NB
We have used training input as statements of the preprocessed training data and training labels as training data labels. Finally we have trained the model with these training input and labels. The performance of the model is estimated on the basis of macro F1 score.

3.2 LSTM model

Preprocessing: Here, we have first encoded the training data columns which contains text values into integer labelling so that the data can be used in the deep learning models to be used in this project. These models work only on integer/float values. The encoded data is added as separate columns in the dataset. For eg. label encoding 'speaker' column into 'speaker-id' column. Then the training data is split into training and validation sets using train-test split of sklearn. Then we have removed the stopwords from the 'statement' column and finally used an additional feature of pos tagging in the 'statement' column.

POS Tagging: For each statement (without removing the stopwords), we generated the POS tags for each word. We selected top 9 frequently occuring POS tags (NOUN, VERB, ADP, PROPN,PUNCT, DET, ADJ, NUM, ADV) from the training set as individual labels and clubbed the remaining tags under a common label X. Further, we represented each POS tag as a 10-dimensional onehot vector. We created a POS embedding matrix of (10 x 10) dimensions which basically is an identity matrix wherein rowi corresponds to the embedding for the ith POS tag.

Training: For each statement, we pass it through the Embedding Layer with weights as embedding matrix. This is further passed as input to the Bi-LSTM. Similary, for each statement, we pass the POS tags through the Embedding Layer with weights as the POS embedding matrix. This is further passed as input to another Bi-LSTM . Both the Embedded layers are non-trainable. Both the LSTMs have output layer size of 100 and a dropout of 0.2. Relu is used as activation function in the dense layer. We finally concatenate the outputs of these 2 above mentioned Layers(with and without pos tags) and pass the output to a Dense Layer which servers as a classifier generating softmax probabilites over 6 output class

3.3 CNN model

The preprocessing for this model is same as LSTM with pos tagging. The only difference lies in the training of the model.
For each training sample, we created two Embedding Layers - one for the statement embedding, and the other for pos embeddings. These embedding Layers are non-trainable. Now, consider the Statement Embedding Layer's output. This is simultaneously fed into two Conv1d $\rightarrow$ MaxPool1d Layers. The output of two MaxPool layers are concatenated and passed through a Dropout Layer to reduce overfitting. Output from Dropout Layer is then fed to Dense Layer. Similar layers are constructed for POS embedding. So both of these embeddings are propogated simultaneously and independently. Finally, we concatenate the output and pass it our classifier Layer which generates softmax probabilities over 6 output classes.

3.4 Inputs and Parameters

• **Inputs** : In the training data, each statement is reducted to 15 words. Each word in each statement is represented by 100-dimensional vector representation as from GloVe. Each statement will have 15 POS tags (one for each of the 15 words) wherein each tag is a 10-dim one hot vector. The pos-embedding will have (None, 15, 10) shape.

• **LSTM :** – embedding-dim = 100
– hidden-size = 100
– lstm-size = 100
– num-steps = 15 : words in a stmt
– num-epochs = 30
– batch-size = 40
– dropout at bi-LSTM = 0.2
– optimizer = SGD(lr=0.025, clipvalue=0.3, nesterov=True)
– loss function = categorical crossentropy

• **CNN :** – kernel sizes for each of conv1d = 3
– filter-size for each of conv1d = 128
– num-steps = 15 : words in a stmt
– num-epochs = 30
– batch-size = 40
– dropout at CNN = 0.6
– optimizer = SGD(lr=0.025, clipvalue=0.3,

nesterov=True)
– loss function = categorical crossentropy

4 Experimental Results

We used the macro-F1 score as an evaluation measure. Since this dataset is a balanced one, it was observed that the f1-score results from different models were equivalent to respective accuracies.

4.1 Dataset Details

The training set contains 7168 rows and 8 columns which are : ’label’ , ’statement’, ’subject’ , ’speaker’ , ’speakers job title’ , ’state info’ , ’party affiliation’ , ’id’. The test data contains 3072 rows and 7 columns. Labels are missing in the test data. The fine-grained labels for the truthfulness ratings are : pants-fire, false, barely-true, half-true, mostly-true, and true.

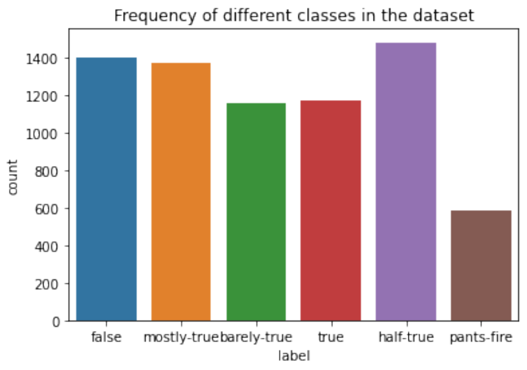

Figure 1: Distribution of different classes in dataset

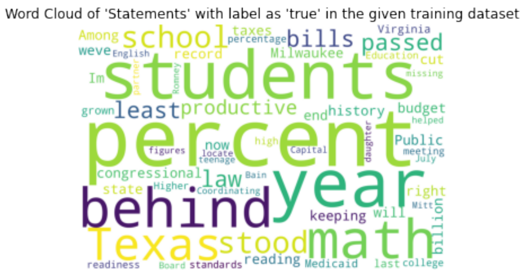
Word Cloud of 'Statements' with label as 'true' in the given training dataset
students
percent
year
behind
Texas
math
school
bills
passed
stood
law

Figure 2: Word Cloud of ’Statements’ with label as ’true’ in the given training dataset

4.2 F1-scores of our experiments

The F1-scores of various experiments that we carried out are listed below.

Model	Feature	F1-score
Naive Bayes	Multinomial	0.20742
CNN	stmt	0.2315
CNN	stmt + pos	0.18111
LSTM	stmt	0.18657
LSTM	stmt+pos	0.21422

5 Analysis

• For CNN, Statement + POS tagging decreased the performance. CNN are not able to use the temporal information from POS tagging accurately.
• For CNN, only Statement produced better results.
• For bi-LSTM, Passing only the statement embedding to bi-LSTM produces 0.18657 F1-score. With only statement, our model is not able to learn good representations. This shows we need richer features as input to our model.
• For bi-LSTM, Statement and information about the subject of the sentence produced good results than statement alone. i.e. Subject information is useful for the model to predict more accurately. Hence using POS tagging increases the model score to 0.21422.
• We concluded that LSTMs are better classifiers than CNNs for textual data

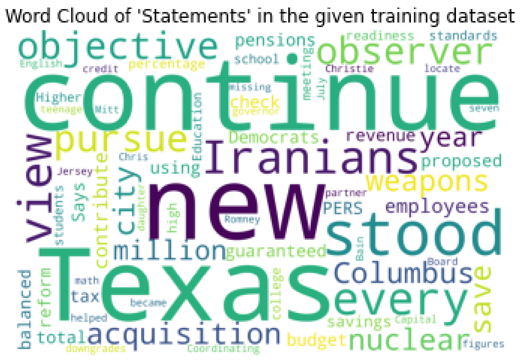

Figure 3: Word Cloud of ’Statements’ in the given training dataset

6 References

Fake News, Fake News Wikipedia
William Yang Wang, Liar, Liar Pants on Fire
Bidirectional LSTM ,implementation
Fake News Implementations, papers with code

7 Individual Contribution

Aditya Jain: Models Implementation, Report, Slides
Harsh Goyal: Models Implementation, Report, Slides
Vasu Yadav : Slides, Report
Akanksha: Slides