FAKE NEWS CLASSIFIER

CS337 Artificial Intelligence and Machine Learning

Advaid Deepak 20d070006
Arpon Basu 200050013
Govind Saju 200070020
Sanchit Jindal 200020120

Contents

1 Abstract

Fake News can lead to a lot of misinformation and reduce credibility of the sources, As humans are not always able to differentiate the fake news from the genuine news we try to provide some order to the chaos by building a fake news classifier. Machine learning approaches to fake news detection typically include vectorization of text followed by classification. In this report, we compare and contrast different combinations of vectorizers and classifiers for the task of fake news detection.

2 Overview

We aim to build several models to detect fake news, and present our findings. The primary dataset used for this has been obtained from Kaggle. Our pipeline for detecting fake news has the following steps:

- **Preprocessing** - Preprocessing is done on the text. This includes removing punctuation, converting to lowercase, removing stopwords. We also choose to combine the title and the text into a single block of text here.
- **Vectorization** - We aim to convert a block of preprocessed text into a vector representation. The two approaches we analysed are the **TFIDF** Vectorizer(Term Frequency Inverse Document Frequency) and a vectorizer using **GloVe Embeddings** with an **LSTM** Network.
- **Classification** The output of the vectorizer is fed into a classifier which performs binary predictions on whether the news is fake or not. We tested a variety of commonly used classifiers such as Logistic Regression, Passive Aggressive Classifier and Random Forest Classifier.

3 Vectorizer Options

While choosing a vectorizer we have many choices, a naive vectorizer can use the simple approaches like One Hot Encoding but they have various drawbacks. In this project, we compare the following vectorizers, and contrast the performance of the two.

- **TFIDF** (Term Frequency-Inverse Document Frequency)

  This vectorizer vectorizes the document based on the statistical properties of the words in it. It is very fast and is purely deterministic, this helps as each data point is mapped to a constant vector. However, it fails to consider the correlation between the positions of words, and is purely based on the number of occurances. One major con of the Tfidf vectorizer is the dimensionality of the vectors generated. In unrestricted Tfidf, the resulting vectors have the same size as the vocabulary.

- **GloVe + LSTM**

  Here, we use a deep neural network to try to include the effect of the positions of words. The use of GloVe vectors also helps us infer meaning from words, and it is hence not biased by the different data meaning the same thing. It also is not swayed by noise such as unnecessary words. A good advantage of this vectorizer is the small dimensionality of resulting vectors.

4 Classifier Options

The vectorizer provides us with a numerical representation of text in the form of vectors, where we need to produce a classifier between the data vector and the cluster (Fake or Real News). For this purpose we have tested the following classifiers :-

- **Passive Aggressive Classifier**

  This classifier is an online learning algorithm and is based on the paradigm to heavily penalize the wrong predictions, while not changing the model if classification is correct. It is useful to use when dealing with sequential data.

- **Random Forest Classifier**

  It is an ensemble model basing its prediction on a large number of individual decision trees.

- **Logistic Regression**

  Employing the classic Logistic Regression model that is taught in the course.

- **Adaboost Classifier**

  Adaboost is also an ensemble algorithm, it combines a series of low performing or weak classifiers with the aim of creating a strong classifier.

5 Development of Models

5.1 GloVe + LSTM

GloVe embeddings refer to a set of word embeddings developed at Stanford in 2014. We utilised the 50 dimension Glove embeddings (the smallest available) due to constraints on GPU access.

After preprocessing text, each word is converted into its corresponding Glove vector in order to obtain a sequence of word vectors. This sequence of vectors is passed onto an LSTM network, from which we use the hidden state as the output of the LSTM. Hence, a sequence of word vectors is converted by the LSTM into an n-dimensional vector (we used n = 32). This completes the vectorizer part of this model.

In order to train this vectorizer, we used a classifier comprising of a Linear layer followed by a TanH activation function. The above vectorizer and the classifier were trained in conjunction using the Binary Cross Entropy loss function. This model was tested on the validation and test data as well. After training, we dropped the classifier developed above, and used the vectorizer alone as a function to map from a text sequence to a vector representation. The output from this vector representation was used to train a series of classifiers (as mentioned above) and their accuracies were compared.

5.2 TFIDF and TFIDF-32

We used the standard implementation of TFIDF vectorizer from scikit-learn to convert a body of text into a vector representation. This was used to train a model for vectorization, whose output was used on the different classifiers mentioned. In order to compare the effect of dimensionality - we tested two variants of tfidf, one where the size of resulting vectors were unrestricted, while

in the second we restricted the size of tfidf vectors to the same size as n=32, which is generated by our implementation of the LSTM vectorizer. We refer to this second variant of TFIDF as TFIDF-32 for the rest of the report.

6 Results and Analysis

The different models we built were tested in two ways:-

- The accuracies were tested on Kaggle's test data, where we do not have access to the true values. Submissions were made to Kaggle and the accuracy on the test dataset was determined.
- In order to check other metrics as well, we separated a portion of our training data into train and test data, and the f1_score, precision and recall were also calculated

6.1 Accuracy on Kaggle's test dataset

Classifier / **Vectorizer**	Passive Aggressive	Ada Boost	Logistic Regression	Random Forest
Glove + LSTM	96.56	96.50	96.61	96.71
TFIDF	97.38	95.48	95.55	93.56
TFIDF-32	77.19	78.54	79.00	80.79

6.2 Accuracy, Precision, Recall and F1-Score on test dataset

Here, TFIDF refers to unrestricted TFIDF where resultant vectors can be as large as possible, whereas TFIDF-32 represents the case where we restricted the size of TFIDF vectors to 32 dimensions. All values mentioned are in percentages.

Vectorizer	**Classifier**	**Accuracy**	**Precision**	**Recall**	**F1-Score**
Glove + LSTM	Passive Aggressive	96.49	96.72	96.35	96.53
Glove + LSTM	Ada Boost	96.22	96.43	96.12	96.27
Glove + LSTM	Logistic Regression	96.68	96.47	97.01	96.74
Glove + LSTM	Random Forest	96.68	96.64	96.82	96.73
TFIDF	Passive Aggressive	97.35	97.38	97.38	97.38
TFIDF	Ada Boost	95.48	95.56	95.47	95.52
TFIDF	Logistic Regression	95.55	95.40	95.80	95.60
TFIDF	Random Forest	93.26	94.87	91.61	93.21
TFIDF-32	Passive Aggressive	78.75	74.65	87.66	80.64
TFIDF-32	Ada Boost	80.10	79.74	81.19	80.46
TFIDF-32	Logistic Regression	79.95	79.14	81.86	80.47
TFIDF-32	Random Forest	82.78	82.12	84.23	83.16

6.3 Dimensionality of Vectorizer Output

Vectorizer	Dimension
TFIDF	161737
TFIDF32	32
Glove + LSTM	32

6.4 Analysis

The results summarised above show that Glove + LSTM work as an excellent vectorizer, providing a low dimensional vectorization of text that is consistent accross classifiers. It beats TFIDF on most classifiers, and is seen to be far more effective as the dimensionality of vectors is only 32, while TFIDF produces vectors the size of the vocabulary (in our case this was over 160000). If we restrict the number of features in TFIDF to 32 to compare the differences between our vectorization approach with that of TFIDF, then the differences are stark. Our method attains accuracies over 96%, while TFIDF-32 obtains only near 80%. Hence, it can be concluded that using Glove vectors + LSTM for vectorization provides a very effective and concise representation of large passages.

7 References

1. Kaggle - Dataset: https://www.kaggle.com/competitions/fake-news/data
2. Stanford - Glove embeddings: https://nlp.stanford.edu/projects/glove/
3. Loading glove embeddings in pytorch: Medium - mlearning-ai

A Some theory on Vectorizers

TFIDF

A reference to Tfidf can be found here

TFIDF stands for Term Frequency - Inverse Document Frequency TFIDF can be defined as

$$TFIDF(t,d) = TF(t,d) \times IDF(t)$$

where

$$\begin{aligned} TF(t,d) &= \text{Term Frequency that is number of times t appears in d} \\ IDF(t) &= \text{Inverse Document Frequency} \\ &= \log\left(\frac{1+(\text{number of documents})}{1+(\text{number of documents in which t appear})}\right)+1 \end{aligned}$$

TFIDF is great for processing large amount of data as it a purely Statistical Method, It is also deterministic so we get the same results for the same data.

But TFIDF does not associate any meaning to the words due to which it is easily swayed by noise and by words that are not required by the classifier

GloVe + LSTM

A reference to Glove can be found here
A reference to LSTM can be found here

GloVe stands for Global Vectors

LSTM stand for Long Short Term Memory, LSTM are a kind of RNN (recurrent Neural Network), that is the output from the first word is sent as the input to the next network and that output is sent to the third network and so on,

This helps in building a network linking all the words of the sentences and then this intermediate output is used as an input to the classifier which classifies the news as fake or not.

In our Implementation we get the GloVe Vectors of each word and then these vectors are input to the LSTM to get an Intermediate representation, Activation function used is **tanh** function, and the loss is **Binary Cross Entropy loss**. The Output is sent to the chosen Classifier to get the prediction.

B Some theory on Classifiers

Passive Aggressive Classifier

The Reference to Passive Aggressive Models Can be found here

It is an Online Learning Model, That means the data in inputted sequentially and the model is updated after each input, in contrast to the usual Batch Update Model.

These Models leave the correct predictions alone and penalize the wrong predictions, due to this reason it is called Passive Aggressive,

Random Forest Classifier

The Reference to Random Forest Classifier can be found here

First the training data is separated into different "Bags", We generate a model over all the different sets and then calculate the mean of the all the models,

This helps the Final model prevent over fitting as there is no correlation between the different models.

The number of trees should depend on the data available as with too many trees no tree will be able to generalize and the predictions will be skewed.

Logistic Regression

The Reference to Logistic Regression can be found here and in the class notes.

This is the basic Logistic Regression between a dataset of vectors and binary classes like we discussed in class.

Logistic regression is a iterative algorithm to minimize the Loss Function

$$\sum_{i=0}^{N} \frac{-1}{N}(y_i \log(y_i^{pred}) + (1 - y_i) \log(1 - y_i^{pred}))$$

AdaBoost Classifier

The Reference to AdaBoost Classifier can be found here.

AdaBoost improves upon the results of other classifiers, In our code it is implemented over Decision trees.

In AdaBoost Algorithm We run the underlying classifier multiple times to get various models, And then with each model we update the next Model to reduce the number of incorrect Predictions, In this way we start getting a better model than any originally provided