Fake News Prediction

Final Project Report

Tanuj Sood
tanuj.sood_ug21@ashoka.edu.in

Neeraj Pandey
neeraj.pandey_ug21@ashoka.edu.in

Abstract

This project aims to identify sentiments of news data using stance prediction and classify them into four class labels. Through experimentation with different encodings and models, we observe the accuracy levels obtained while testing on the dataset as well as real time testing. We initially started by using Doc2Vec to achieve high accuracy scores but soon observed certain limitations of the the model which made us switch to our second model using the TF-IDF encoding. Testing the model with different classifiers, we were able to obtain a high accuracy on the testing data and analysed the limitations of our work. Finally, we discuss what progress we could make to improve our accuracy scores even further and what new methods or steps could be taken to achieve this.

1. Introduction

As a majority of the data online is unstructured, it is a necessary task to be able to classify all the data which is out there and manage it with ease. Natural language processing (NLP) algorithms have rapidly advanced to deploy machine learning and be able to structure this data expeditiously. However, another significant problem that rises apart from just segregating and structuring data which is of classifying false knowledge.

As social media platforms grow exponentially, a majority of the world, including us, receive most of the news from apps such as Facebook, Instagram etc. Since this social expansion has given everyone the freedom to post whatever they wish and like, these platforms have been plagued with fake news. With billions of posts and millions of people online each day, it is impossible to monitor everything that is posted on these platforms manually.

This is where fake news detectors come in. With the evolution of fake news detectors, media sites such as Twitter have been able to successfully reduce the number of tweets which contain false information. This is to secure the audience from consuming untrue facts, some of which could potentially harm another person's interests. Thus, in our project, we attempt to solve the fake news challenge with high testing accuracy and a decent level of real time prediction accuracy.

2. Problem Definition

Input: Article headline and body text of the article.

Output: Stance Prediction through classification into one of the given four class labels:

1. **AGREE:** Heading agrees with body text.
2. **DISAGREE:** Heading disagrees with body text.
3. **DISCUSS**: Body text discusses topic in heading.
4. **UNRELATED:** Body text and heading are unrelated.

3. Related Work

As Fake News detectors increase in demand, there have been several attempts to achieve well-performing efficient models which can detect stances in real time with high confidence levels. Looking at the Fake News Challenge [1], we analysed the steps that the winners, Talos Intelligence and runner-ups, Athene Systems took to achieve accuracy scores of approximately 82%.

Talos Intelligence's approach [2] was experimental and unique to this challenge where they attempted to combine two machine learning models in order to make predictions. In their model architecture, an n-gram model was encoded with different encoding methods such as TF-IDF, SVD and Word2Vec which was then trained with a gradient-boosted decision tree and a deep convolutional neural network. The results were combined and averages from both models were taken with a 50-50 weights to then predict the stances in a real-time implementation.

In Athene Systems [3] approach, they used a Bag of Words (BoW) model and then proceeded to optimise hyper-parameters, a random search is carried out using a multi-layer perceptron model. Performance is further optimised by implementing latent semantic indexing and other tools to achieve an accuracy level of 81.9%.

4. Dataset and Features

4.1 Dataset Analysis

The dataset used for this project is from the fake news challenge website, a competition which encourages teams to create a fake news detector using NLP techniques and obtain high real time accuracy.

We used the "FNC-1" [1] dataset which was segregated into training and testing data with article headlines, article bodies and the class label. The classes were labeled as follows.

Class 0: **AGREE**

Class 1: **DISAGREE**

Class 2: **DISCUSS**

Class 3: **UNRELATED**

In the given dataset, we performed composition analysis on the dataset to find out if there were any imbalances in the data populated with a particular class label. The composition of the training and test datasets have been shown through histograms in Fig 1 below.

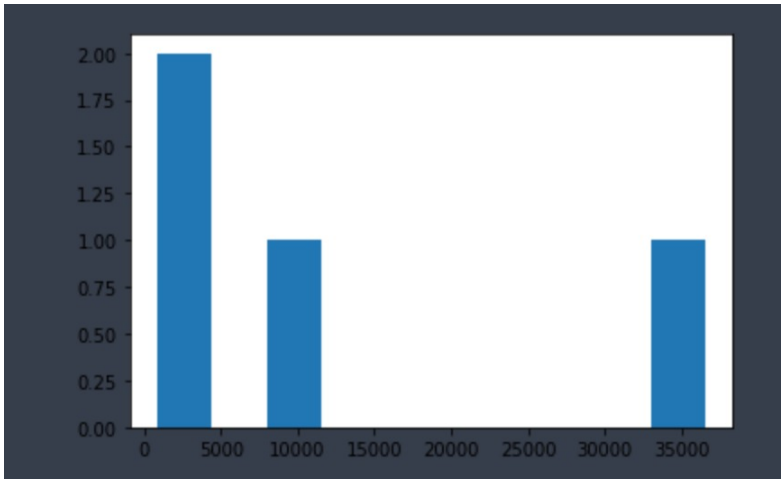

Fig 1. Data Composition for each class label

To handle the data easily and perform functions on it, we transform it into a "Pandas DataFrame." After this step, we go on to preprocess the data, including stemming and lemmatisation, which will be discussed in Section 5 of this report.

4.1 Data Pre-Processing

After appending all data to a dataframe, we pass it through regular expressions to remove all special characters and brackets to clean the data. This cleaned data is then tokenised into separate words and then pass the data through stemming and lemmatisation functions.

This process simplifies the data and remove any unnecessary synonyms while stemming them to their root words. When we encode this data using different encoding models, we will better results as there will be a higher frequency of root words which will be counted together to form a higher weightage for a particular word.

This whole preprocessing stage is done for both the training and test datasets and then they're passed on to different models.

5. Methods

5.1 TF-IDF Model

Our approach in solving this challenge is based on the NLP concepts learnt in class, where we encode our data based on the term frequencies and create TF-IDF indexes. After this process we performed experiments with different classification models and calculated accuracy levels on the test data.

5.1.1 Cosine Similarity

Cosine Similarity calculates the similarity between two non-zero vectors and checks their similarity based on the distance the vectors are pointing in. The formula to calculate cosine similarity is given in the figure below.

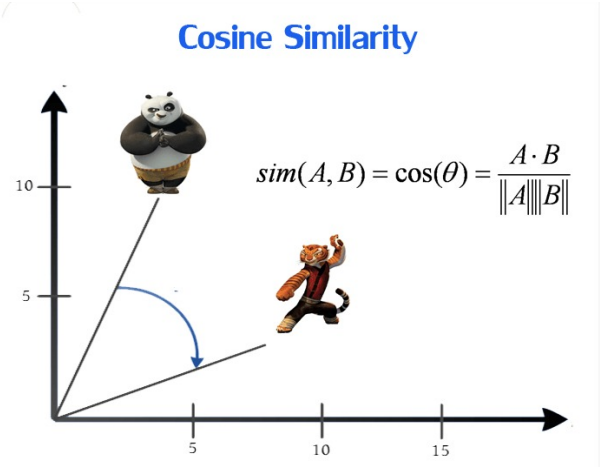

Fig 2. Cosine Similarity Formula

We use cosine similarity as a feature in our

5.1.2 Feature Extraction

To prepare the data for the TF-IDF model, we first one hot encode the class labels to avoid any unnatural ordering of the stance values. Then, TF index and an IDF index is created using the following formulae.

$$\mathrm{TF}_{ij} = \frac{\text{No. of times the word i appears in document j}}{\text{Total number of words in document j}}$$

$$\mathrm{IDF}_{ij} = log \frac{\text{Total number of documents}}{\text{Number of documents with word i in the specific document}}$$

Fig 3. TF-IDF Formulae

Calculating the tf-idf values by multiplying the tf values for each word with their respective idf value, we create a vector to create weights for each of the words based on their tf-idf values. The vocabulary is then generated by combining the headings and bodies of the articles in the training dataset. The vocabulary is then used to fit the data in both term frequency and inverse term frequency. Here, we then use the top 2500 features in our vocabulary to create a vector for each heading and body column for each row in our training data.

5.1.3 Model Architecture

To perform classification in the testing dataset, we use different models trained on our training dataset. In order to receive and compare accuracy levels, we feed our training dataset to random forests, logistic regression, Gaussian naive Bayes, Bernoulli naive Bayes, decision trees and multi layer perceptron classifiers and then compare the results with the test data.

5.1.4 Model Results

The results from the TF-IDF model using different classification techniques have been given in the figure below. On top of this, our model performs well on real time data when news inputs are fed into the model.

	accuracy
Random Forest	86.062252
Logistic Regression	86.113407
Gaussian NB	21.740841
Bernoulli NB	54.590170
Decision Tree	79.538032
MLP	76.496281

Fig 4. Testing Accuracy for TF/IDF

Since we get the highest accuracy from logistic regression, we will use it to create a confusion matrix. The confusion matrix acquired from the logistic regression classifier has been given below.

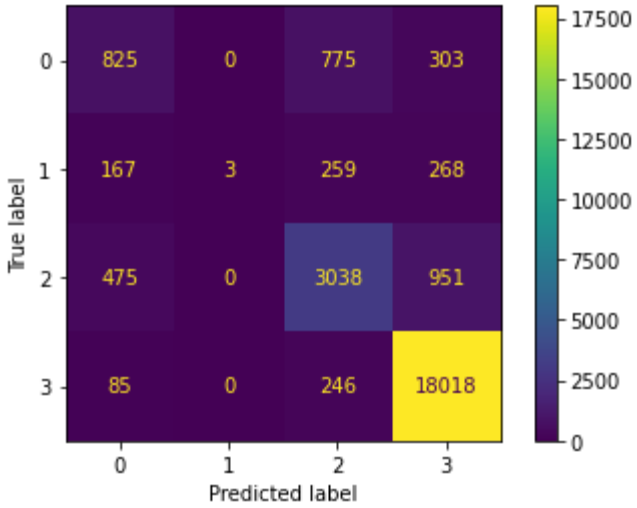

Fig 5. TF/IDF Confusion Matrix

As we can note, the “DISAGREE” class label doesn’t have any entries in the confusion matrix. This is due to the imbalance of the training data which created a bias towards the “UNRELATED” class label. We will improve our model by implementing the synthetic minority oversampling technique (SMOTE) to eliminate this imbalance.

5.2 Doc2Vec Model

Our second attempt at improving our accuracy levels obtained was by using the Doc2Vec model. In this method, we converted the documents into

a numeric representation of the heading and the article body, despite of its length. This and then passed on the data to different classification models.

5.2.1 Feature Extraction

We pass on our previously preprocessed dataset directly to the Doc2Vec model. There is only one modification required to run the model which is to change labels to integers in order to avoid “UNKNOWN” errors.

5.1.3 Model Architecture

To perform classification in the testing dataset, we use different models trained on our training dataset. In order to receive and compare accuracy levels, we feed our training dataset to random forests, logistic regression, Gaussian naive Bayes, Bernoulli naive Bayes, decision trees and multi layer perceptron classifiers and then compare the results with the test data. The figure below shows the model architecture we are using to create document vectors and then concatenating them before sending them to our classifiers.

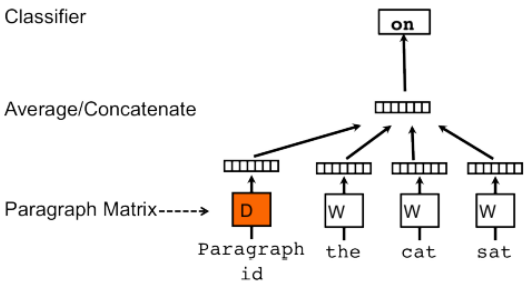

Fig 6. Model Architecture of Doc2Vec

5.1.4 Model Results

The results from the Doc2Vec model using different classification techniques have been given in the figure below. Our model doesn’t perform well on real time data when news inputs are fed into the model. The table

	accuracy
Random Forest	72.002518
Logistic Regression	72.057608
Gaussian NB	60.480856
Bernoulli NB	59.961437
Decision Tree	57.100696
MLP	62.865463

Fig 7. Testing Accuracy for Doc2Vec

below shows the results acquired on different classification models when compared to the testing dataset.

Since we get the highest accuracy from logistic regression, it is the best in Doc2Vec as well. However, the overall accuracies obtained are much lower than that of the TF-IDF model. Thus, this model is not effective enough to run on real time data and is not practical in real life.

6. Improvement of TF-IDF Model

Reading up on the data imbalance of our data, we experimented with **synthetic minority oversampling technique (SMOTE)** to see if the results would differ from the previously acquired results. Since the TF-IDF model had the best performance, we sent the balanced data to this model and ran it on the same model architecture.

6.1 Data Preparation

The data preparation was the same as last time, the only difference was the use of imbalanced-learn where we used SMOTE (Data Augmentation) to balance the dataset. Compared to the starting dataset, SMOTE balanced all points by oversampling the minority in the dataset. The data composition has been portrayed through the histogram below.

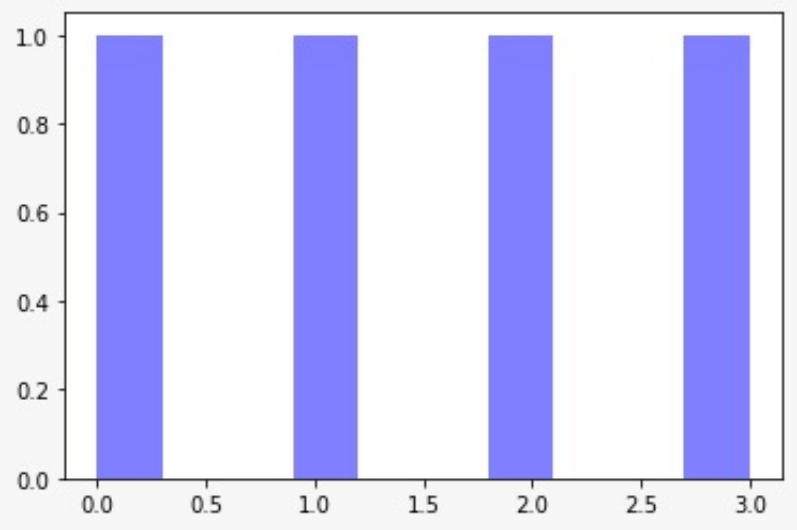

Fig 7. Data Composition for each class label using SMOTE

6.2 Model Results

The results from the TF-IDF + SMOTE model using different classification techniques have ben given in the figure below. On top of this, our model performs well on real time data when news inputs are fed into the model.

	accuracy
Random Forest	77.074726
Logistic Regression	84.322984
Gaussian NB	26.632039
Bernoulli NB	61.932869
Decision Tree	76.956676
MLP	81.281234

Fig 8. Testing Accuracy for TF/IDF with SMOTE

Since we get the highest accuracy from logistic regression, we will use it to create a confusion matrix. The confusion matrix acquired from the logistic regression classifier has been given below.

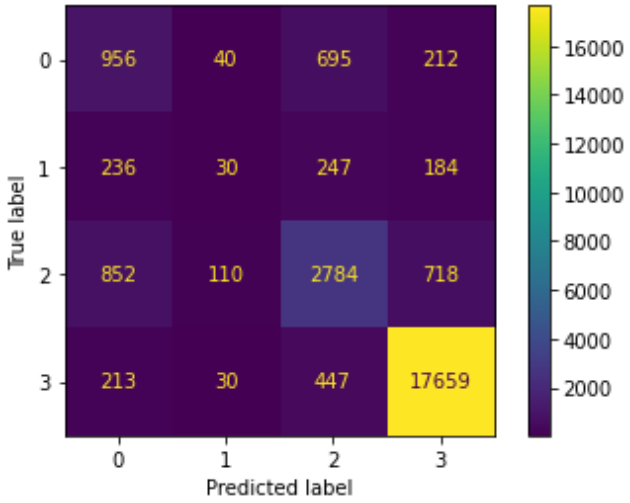

Fig 9. Confusion Matrix for TF/IDF with SMOTE

As we can note, the “DISAGREE” class label doesn’t have any entries in the confusion matrix. This is due to the imbalance of the training data which created a bias towards the “UNRELATED” class label. We will improve our

model by implementing the synthetic minority oversampling technique (SMOTE) to eliminate this imbalance.

7. Failed Models

7.1 Bag of Words

The observed accuracy was coming out to be 50% - 60% in most classifiers. The reason being, the BoW models are 0s and 1s vectors and don't show any relationship with each other. We concluded that this was one of the factors why the accuracy levels weren't comparable to other models. This is why we moved to other models such as Doc2Vec and TF-IDF models.

	accuracy
Random Forest	60.2
Logistic Regression	61.1
Gaussian NB	29.9
Bernoulli NB	54.2
Decision Tree	58.4
MLP	55.5

Fig 10. Testing Accuracy for BoW

7.2 Doc2Vec

We performed the model on the top 300 to 500 vector dimension features and were not able to surpass this ceiling due to computational restrictions. We tried running our model with higher dimensions but the accuracy levels from the top 300 to the top 500 remained consistent and we were set back by computational limits of our devices. We believe that either using a larger corpus or using more features would increase the accuracy.

7.3 Without Cosine Similarity

Without using cosine similarity in our TF-IDF model produced decent results with logistic regression accuracy at 75%. Later on, using cosine similarity and adding it directly to the features helped boost the accuracy level for the LR classifier by 11%. Once we included cosine similarity in our actual model, we discarded the previous model with the lack of this feature.

By adding the cosine similarity, we added an extra feature which measured the closeness between the article heading and article body producing a stronger correlation between the two features.

8. Results and Discussion

After performing different classifications on our TF-IDF, Doc2Vec and Class Filtering model, we experimented and found out which model yields the highest testing accuracy. Our quantitative metric to compare models is based on their testing accuracy over the dataset.

The obtained results and discussion with state-of-the-art models have been given below.

8.1 Model 1: TF/IDF

After a couple of tests with different classifiers and all three of our models, we received a testing accuracy level of 86% over the dataset with our TF-IDF model with the logistic regression classifier. We also created a real time model which proof-checked by the TA and worked with all the real time test cases we used in the demo.

This model had the best accuracy we had, the only problem though, was the data imbalance classification problem which was fixed with the inclusion of SMOTE method for imbalanced classification.

8.2 Model 2: Doc2Vec

The results received from the Doc2Vec model were decent when performed over a number of different classifiers. However, neither the testing accuracy nor the realtime prediction was effective and didn't compare to the results we obtained through either of our TF/IDF models. As mentioned above, we were mostly restricted by the corpus size and computational power, both of which could make the model perform with higher accuracy.

8.3 Model 3: TF/IDF with SMOTE

To go further in an attempt to improve our model accuracy, we tried to fix the problem of data imbalance by using the SMOTE method. Our initial hypothesis was that using SMOTE would increase our current model's performance. However, this hypothesis was refuted as our obtained test accuracy was lower than the previous accuracy.

Though the accuracy was slightly lower, the confusion matrix yielded better results by populating “DISAGREE” class variable sections compared to that of the previous model.

8.4 Comparison with State-of-the-Art Results

Though the accuracy we observed in our TF/IDF model was higher than that of state-of-the-art results, we believe Talos Intelligence’s model was a better model than ours if we run both programs with current technologies. Considering the model was created three years ago, it has the ability to outperform our model since it trains on a combination of decision trees and deep CNNs which would be much more powerful today than three years back.

7. Conclusion and Future Work

Working on this project, we realised the major demand of fake news detectors in today’s times and how it is extremely complex to make them highly efficient. We experimented with different encoding models and various classifiers to compute the accuracy levels when performed on both real time and testing data. Though we tried to perform better than our first model using SMOTE, we believe that we can outperform our current model with higher computational power used on the Doc2Vec model or by training a deep CNN model on a high number of epochs.

GitHub Link:

8. References

[1] https://github.com/FakeNewsChallenge/fnc-1

[2] https://blog.talosintelligence.com/2017/06/talos-fake-news-challenge.html

[3] https://github.com/hanselowski/athene_system/blob/master/system_description_athene.pdf

[4] https://www.sciencedirect.com/topics/computer-science/cosine-similarity

[5] https://www.sciencedirect.com/topics/computer-science/inverse-document-frequency