Natural Language Processing Technologies to Combat Fake News

UC Berkeley CS 298 Final Project

Bo Li	Jingran Zhou	Faye Yu
SID 3035322863	SID 3035326399	SID 3035326178
bo_li1996@berkeley.edu	jingran_zhou@berkeley.edu	fayeyu@berkeley.edu

Abstract

With the fast development of digital media, the authenticity of the information in news has become a longstanding concern influencing businesses and society. The goal of this project is to apply modern deep learning technologies to estimate the relative perspective of two pieces of text relative to a topic, and thus to combat fake news. Our GitHub repository records some combinations of natural language processing technologies we tried, including preprocessing, embedding, feature engineering, and classification. Our final implementation with Universal Sentence Encoder (USE), Term Frequency-inverse Document Frequency (TF-IDF) features, and the fine-tuning neural network achieved a best weighted test accuracy of 82.015%, which is accessible at this Colab notebook.

1 Introduction

According to a Pew Research poll, 64% of US adults said that a great deal of confusion has been caused by made-up news about the facts of current events.[1] A helpful step towards identifying the fake news is to understand its topic and stance, as well as their relationship. This method is called stance detection. This project aims to use machine learning, particularly natural language processing technologies to automate the process of stance detection and thus to identify fake news.

2 Dataset

2.1 Fake News Challenge

We used the dataset provided by the Fake News Challenge website.[2] The data provided is (headline, body, stance) instances, where the stance is one of the labels: *unrelated, discuss, agree, disagree*. There are two CSV files, one is 'bodies.csv', containing the body text of articles with corresponding IDs. Another one is 'stance.csv', containing labeled stances for pairs of article headlines and article bodies. The CSV files are split into training data and testing data. The 'train_bodies.csv' has 1683 body texts and the 'train_stances.csv' has 49972 rows in total because the same body texts are referenced under different headlines. Similarly, the 'test_bodies.csv' has 904 body texts and the 'test_stances.csv' has 25413 rows waiting to be labeled.

2.2 Exploratory Data Analysis

We made a statistic and plotted the stance distribution on the training data. There are 36545 stances for *'unrelated'*, 3678 for *'agree'*, 840 for *'disagree'*, and 8909 for *'discuss'*. We found that over 73.1% of data are labeled as unrelated.

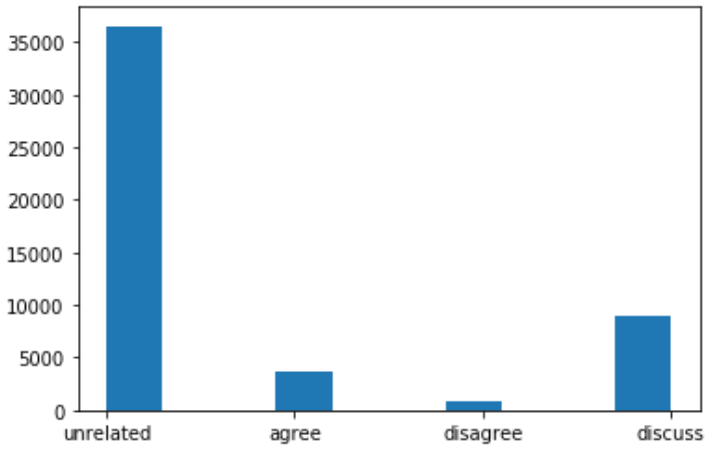

Figure 1 The distribution of stance classes in train_stances.csv

Therefore, we split the training data into the related data and unrelated data. To find some representative words in these two datasets, we created a word cloud for each of them. However, as we can see there are not too many differences in the most common words for unrelated and related data. Therefore, we need to find an efficient way to embed the words and extract some key features from the sentence.

Figure 2 Word cloud for training data

3 Methods

The crux of achieving state-of-the-art performances is finding the most appropriate feature encoding for the sentences in the Fake News dataset. We have decided to use the following methods as they are canonical in the field of NLP and compare their performances with the baseline.

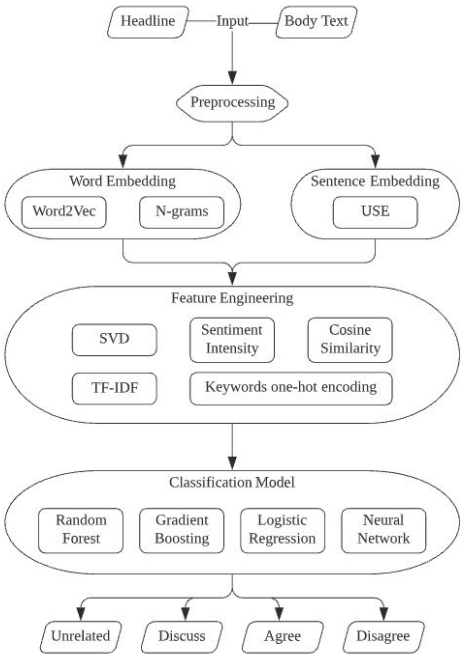

Figure 3 The proposed training pipeline

3.1 Data Preprocessing

We utilized the NLTK (Natural Language Toolkit) to conduct the data preprocessing. To remove the unmeaningful words, we used stopwords in NLTK. To normalize the words in the text, we used the NLTK Snowball stemmer. Besides the data, we also encoded the four stances into numeric target values as 0 for *agree*, 1 for *disagree*, 2 for *discuss,* and 3 for *unrelated*.

3.2 Embedding Methods

3.2.1 Word Embedding

For word embedding, we used pre-trained word vectors on Google News corpus to embed the headline and body text. After encoding, the bi-gram and trigram features were added to the dataset to capture the information provided by phrases or short terms.

3.2.2 Sentence Embedding

For the sentence embedding, we used a pre-trained Universal Sentence Encoder (USE) with a Deep Averaging Network(DAN) architecture to convert the text into high dimensional vectors. Since we were limited to the computational resources, we chose DAN as it will give us computationally less expensive encoding and with little lower accuracy.

3.3 Feature Engineering

Besides the embedding methods, we extracted some other key features to help us retrieve the information from the texts.

3.3.1 Key Words One-hot Encoding

This feature is adopted from the baseline model. We extract some words expressing attitudes and encode them as 1 if they occur in the body texts otherwise 0. These words include *fake*, *fraud*, *false*, *deny*, *despite*, *doubt*, etc.

3.3.2 Term Frequency-inverse Document Frequency (TF-IDF)

TF-IDF measures how relevant a word is to a document in a collection of documents. The vocabulary is constructed by concatenating the headline and body text. Then the term-frequency will be calculated using the same vocabulary and normalized by its inverse document frequency.

3.3.3 Singular Value Decomposition (SVD)

Because TF-IDF gives a very large and sparse matrix, we used the singular value decomposition to take the principal components of the TF-IDF matrix. We truncated the matrix to 50 dimensions, which means we took the first 50 components.

3.3.4 Sentiment Intensity

We conducted a sentiment analysis to measure the inclination of the article's opinion. We used the Sentiment Analyzer in NLTK to assign a sentiment polarity score to the headline and body. A positive score implies a positive opinion and a negative score implies a negative opinion. We can use it to detect whether the headline and body text express the same opinion.

3.3.5 Cosine Similarity

The cosine similarity is a measure of similarity between two vectors of an inner product space that measures the cosine of the angle between them. This could be used as an additional feature to measure the similarity between the headline and the text body.

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}},$$

Figure 4 Formula to calculate cosine similarity of two vectors

3.4 Classification Models

We used random forest, gradient boosting, logistic regression, and fine-tuning neural networks as the classification models to classify each pair of headline and body text into four classes of instance. Typically, we trained a small neural network with four hidden layers with ReLU activation, one linear layer, and one softmax layer.

3.5 Evaluation

We followed the evaluation schematic provided by the Fake News Challenge website[2] as below.

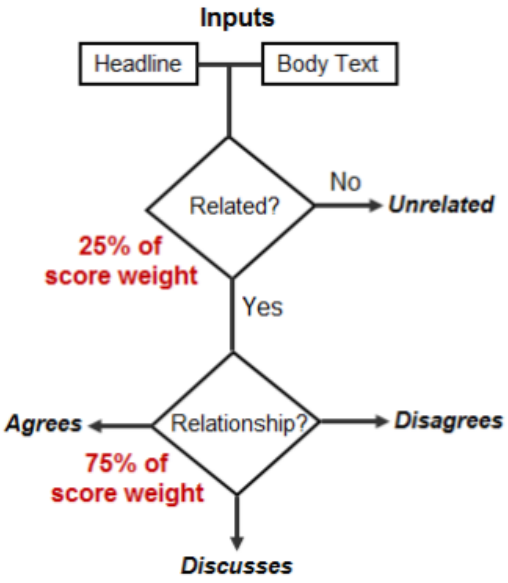

Figure 5 Scoring system to calculate weighted accuracy of labels

The baseline implementation including preprocessing, word/n-gram overlap features an indicator features for polarity and refutation, along with a gradient boosting classifier achieves a weighted score of 79.53% with 10-fold cross-validation. [3]

4 Experiments and Results

We tried different combinations of the above techniques to find out the best model with the highest weighted score. The validation and test scores for each model are listed below. To address the problem of data imbalance, we proposed a novel model with two classifiers where the first classifier will split the data into related and unrelated pairs and the second classifier will further classify the related pair. Therefore, the models with two classifiers gave us two validation scores and are shown in the first two rows of the table. The best model we found was implemented with USE, TF-IDF, and the neural network, and gave us the best test score of 82.015%.

Table 1 Weighted scores for proposed models

Preprocessing	**Embedding**	**Feature Engineering**	**Classification Model**	**Validation Score**	**Test Score**
Stopwords removal, stemming and lemmatization with NLTK	USE	TF-IDF, sentiment intensity, SVD	Gradient Boosting + Gradient Boosting	82.037% + 69.119%	56.455%
	Word2Vec N-gram	TF-IDF, sentiment intensity, SVD	Logistic Regression + Gradient Boosting	93.451% + 69.119%	75.202%
None	USE	None	Random Forest	87.075%	44.549%
			XGBoost	N/A	48.581%
		TF-IDF, cosine similarity	Neural Networks	90.294%	**82.015%**

5 Conclusion

With an accuracy of 82.015%, we prove that automation of stance detection is feasible with modern natural language processing approaches. For this particular imbalanced dataset, we find that both word-level information and sentence-level information needed to be extracted. We make a hypothesis that the boosting methods didn't work well with sentence-level embedding because some keywords information is lost during the weight adjustment. Therefore, USE and TF-IDF passed into Neural Networks give a better performance because word-level and sentence-level information play an relatively equivalent role in making predictions.

Reference

[1] M. Barthel, A. Mitchell, and J. Holcomb, “Many Americans Believe Fake News Is Sowing Confusion,” *Pew Research Center's Journalism Project*, 31-Dec-2019. [Online]. Available: https://www.journalism.org/2016/12/15/many-americans-believe-fake-news-is-sowing-confusion/. [Accessed: 09-May-2020].

[2] “Fake News Challenge Stage 1 (FNC-I): Stance Detection,” *Fake News Challenge*. [Online]. Available: http://www.fakenewschallenge.org/. [Accessed: 09-May-2020].

[3] FakeNewsChallenge, “FakeNewsChallenge/fnc-1-baseline,” *GitHub*, 25-Jul-2017. [Online]. Available: https://github.com/FakeNewsChallenge/fnc-1-baseline. [Accessed: 09-May-2020].