Fake News Classification based on Stance Detection

Anchit Bhattacharya, Ganesh Rakate
School of Computing, Informatics, and Decision Systems Engineering
Arizona State University
{abhatt22, grakate}@asu.edu

November 26, 2018

Abstract

The Kaggle competition WSDM - Fake News Classification[1] aims to address the fake news classification problem by using machine learning and natural language processing techniques. The task involves building a classification model whose input will be a pair of a fake news article A and the title of a coming news article B. The classification model will classify the article B into one of three classes as *unrelated*, *agreed* or *disagreed* by comparing it with the corresponding article A. We propose a stance detection based approach to solve the fake news classification problem. Stance detection means finding the attitude expressed in a text towards a target text. We have presented a feature engineering approach which extracts multiple textual features to use for classification. We plan to improve our approach by incorporating semantic understanding features and also Bidirectional-LSTM with encoder-decoder embedding to find a total combined representation of article A and article B.

1 Introduction

Social media has made consumption of information low cost and very easy to access. However, with the ease of consumption of news, it has also brought about wide propagation of fake news i.e news which intentionally spreads false information (Shu et al. [2017a], Weedon et al. [2017]). Claims made by newsworthy individuals are not considered fake news. Fake news also explicitly excludes humorous or satirical content, which is meant to entertain and not deceive. Fake News has negative impacts on an individual as well as on society. Therefore it has become increasingly important to build fake news detection models, but it's a technically challenging problem. Challenges in the fake news detection problem ranges from collecting reliable data with ground truth, extracting important features, and using these features to build effective models. Most of the data used by researchers for fake news detection has been based on the news content, but recently new research has emerged where people are trying to use social context features which acts as an auxiliary information for fake

[1] https://www.kaggle.com/c/fake-news-pair-classification-challenge

news determination. For a data set collected from news articles or social media networks, it is important to understand which users or articles are more likely to be fake. However, automating the task of identifying or classifying fake news without human intervention is very difficult due to context and semantics of the human language.

The WSDM - Fake News Classification competition hosted by Kaggle aims to address the fake news classification problem by using machine learning and natural language processing techniques. Some of similar competitions previously organized were SemEval-2016 Task 6: Detecting Stance in Tweets[2] and the Fake News Challenge[3].The WSDM - Fake News Classification competition involves building a classification model whose input will be a pair of a fake news article A and the title of a coming news article B. The classification model will classify the article B into one of three classes as *unrelated*, *agreed* or *disagreed* by comparing it with the corresponding article A.

The fundamental approach for classification is to find the stance of article B towards article A. Stance detection means finding the attitude expressed in a text towards a target text. We present a feature engineering approach which extracts multiple textual features to identify the stance of a coming news article B towards a fake news article A. We plan to improve our approach by incorporating semantic understanding features and also Bidirectional-LSTM with encoder-decoder embedding to find a total combined representation of article A and article B. We also plan to create stance based networks between different articles.

2 Related Work

Stance detection is a very crucial stage in any fake news detection and classification problem. Stance detection has been previously deployed for fake news classification or identification at the SemEval-2016 Task 6: Detecting Stance in Tweets and the Fake News Challenge. A detailed analysis of various approaches for the Fake News Challenge Stance Detection Task is provided by Hanselowski et al. [2018]. Similarly, the detailed analysis of SemEval-2016 Task 6: Detecting Stance in Tweets is provided by Mohammad et al. [2016].

Our approach is primarily based on feature engineering for text data (Li et al. [2017]) which extracts multiple textual features to use for classification. Baird et al. [2017] developed an emsemble of feature engineering approach and CNN based deep learning approach for solving the Fake News Challenge. The feature engineering approach involved extracting features using techniques such as TF-IDF, SVD, Word2Vec, sentiment analysis and count vectorization which were then combined using XGBoost model for classification. Ghanem et al. [2018] developed a feature engineering approach involved extracting features using cue words and Google News word2vec embedding along with n-gram co-occurrence between two articles and word overlapping.

LSTM based approaches are most commonly implemented for solving the problem of stance detection. Augenstein et al. [2016] developed a stance detection model for the SemEval 2016 Task 6 Twitter Stance Detection using LSTM Bidirectional Conditional Encoding. It involved building a representation of

[2] `http://alt.qcri.org/semeval2016/task6/`
[3] `http://www.fakenewschallenge.org/`

the text content that is dependent on the target text which outperformed encoding the two text contents independently. Dey et al. [2018] implemented a topical stance detection model which used two-phase LSTM based deep neural network and embedded attention at each of the phases. In the two-phase attention model, the first phase classified subjectivity whereas second phase classified sentiment.

3 Problem Statement

In this section we provide more details on the structure of the data set and the classification task.

3.1 Data set

The data set contains pairs of a fake news article and a new article whose stance towards the corresponding fake news article is to be determined. The data set contains 320,552 train samples and 80,126 test samples. Each data sample has seven columns and one additional column for class label for train samples.

A few sample data points from the train data set are described below.

Data Sample 1:

Article A: "If you do not come to Shenzhen, sooner or later your son will also come." In less than 10 years, Shenzhen per capita GDP will exceed Hong Kong.

Article B: Shenzhen's GDP overtakes Hong Kong? Bureau of Statistics refutes rumor: Unsurpass but the gap shrinks again

Class Label: *unrelated*

Data Sample 2:

Article A: "How to discriminate oil from gutter oil by means of garlic.

Article B: It's very practical to use a single piece of garlic to distinguish oil from oil! "

Class label: *agreed*

Data Sample 3:

Article A: There are over 10,000 people in the field who dissect children and steal their organs.

Article B: it is rumoured that people from outside the city 's quiet sea area stole children' s organs from stealing children 's organs.

Class label: *disagreed*

The column 'id' contains the id number for each data sample. Column 'tid1' contains the id number of 'title1_en' which is the fake news article. Column 'tid2' contains the id number of 'title2_en' which is the new news article whose stance towards the corresponding 'title1_en' is to be determined. Column 'title1_zh' contains the Chinese language text of the fake news article. Column 'title2_zh' contains the Chinese language text of the new news article. Column 'title1_en' contains the English language text of the fake news article. Column 'title2_en' contains the English language text of the new news article. Finally, the column

'label' contains the class label which is one of the three as 'agreed', 'disagreed' or 'unrelated',

Among the train data samples, the approximate ratio of class labels is: 68% *unrelated*, 29% *agree*, and 2.5% *disagree*. It shows that the class distribution in the data set is highly unbalanced.

3.2 Task

The data set can be organized as pairs of text articles, where each pair contains title of a fake news article A and the title of a coming news article B. As described in the introduction, the primary task is to build a classifier to identify the stance of B towards A. If article A and article B are discussing about different topics, then the class label is 'unrelated'. If they are discussing the same topic, there are two possibilities. If B talks about the same fake news as A, the class label is 'agreed'. If B refutes the fake news in A, the class label is 'disagreed'.

3.3 Evaluation Metrics

Weighted categorization accuracy is used for the evaluation of the classifier. The weights of the three categories, agreed, disagreed and unrelated are 1/15, 1/5, and 1/16, respectively.

$$WeightedAccuracy(y, \hat{y}, \omega) = \frac{1}{n}\sum_{i=1}^{n}\frac{\omega_i(y_i = \hat{y}_i)}{\sum \omega_i}$$

4 Proposed Approach

We have developed our model considering only the English language text for both the fake news article A(title1_en) and new news article B(title2_en). The general framework of our implementation is as follows:-

1. We start with basic NLP(Natural Language Processing) tasks to clean the data which involves splitting the sentences into words(tokenization), followed by removing stopwords, changing words in third person to first person and verbs in past and future tenses into present(lemmatization) and reducing words into their root form(stemming).

2. Then we create a vector representation of the tokenized articles, using the following four approaches.

   (a) Count Vectors(Bag of Words) representation using the entire document as dictionary

   (b) TF-IDF representation using the entire document as dictionary.

   (c) Count Vector(BOW) representation using words of a single row as a dictionary

   (d) TF-IDF Vectors using words of a single row as a dictionary

3. After we have title1_en and title2_en in a vectorized form, we try to create different features which would best represent the relation between title1 and title2, and will generate measures which best separates the 3 classes.

4. We create a vector of features for each pair of titles and feed it into Machine Learning Algorithms, along with the labels for training. Then we create the same set of feature on the test data, and pass it to our trained model to generate the final results.

After we create a set of different features, we create a scatter plot of the features, tagged with the labels to get an idea of how effective the features would be. Once the features are finalized we give them as an input to the Machine Learning Algorithms, along with the labels, to train out ML model. Then we do similar transformation on the test data and create similar features and input it to the machine learning model to get the classification result. In the next section we elaborate at each part described above.

4.1 Generation of Features

The major component of our proposed approach is to generate different features from the given text data which will be fed to the classification algorithm. We worked on implementing various techniques for generating features which are best separated for different classes.

4.1.1 TFIDF Features

In one of our approaches, we used TF-IDF scores as a representation of each individual title, and provide a combination of the representations of title1 and title2 as an input to the Machine Learning Model.

4.1.2 SVD Features

We used the TF-IDF representation of each title, and performed Singular Value Decomposition to reduce the dimensionality. We did this for title1 and title2, and combined both of them as to form a feature for our Machine Learning Model.

4.1.3 Topic Modelling

We constructed a dictionary containing text data from both article A and article B. We then implemented Topic Modelling using the Latent Dirichlet Allocation algorithm. Based on the value of model perplexity and coherence score, we constructed the final model with the number of topics to be 100. We analyzed the topics generated and the the weights of words belonging to that topic. We found the top two most relevant topics for each of article A and article B and created features which describe whether article A and article B are talking about the same topic, i.e. whether they are *related* or *unrelated*.

4.1.4 Sentiment Features

We performed sentiment analysis on the the corpus and calculated polarity and subjectivity features for article A and article B in each data sample. Polarity score of a given text describes whether the expressed opinion is positive, negative, or neutral. The subjectivity score describes whether the text content is stating a fact (objective) or expressing an opinion (subjective). We also calculated other features such as number of characters, number of words in each

article. The sentiment features are crucial in classifying the article B into *agree* or *disagree* category. Figure 1 shows a sample of sentiment features.

	title1	title1_len	title1_polarity	title1_subjectivity	title2	title2_len	title2_polarity	title2_subjectivity
0	There are two new old-age insurance benefits f...	94	0.078788	0.218182	Police disprove "bird's nest congress each per...	111	0.100000	0.200000
1	"If you do not come to Shenzhen, sooner or lat...	144	-0.083333	0.033333	Shenzhen's GDP outstrips Hong Kong? Shenzhen S...	106	0.000000	1.000000
2	"If you do not come to Shenzhen, sooner or lat...	144	-0.083333	0.033333	The GDP overtopped Hong Kong? Shenzhen clarifi...	73	0.156250	0.500000
3	"If you do not come to Shenzhen, sooner or lat...	144	-0.083333	0.033333	Shenzhen's GDP topped Hong Kong last year? She...	101	0.000000	0.066667
4	"How to discriminate oil from gutter oil by me...	60	0.000000	0.000000	It took 30 years of cooking oil to know that o...	81	0.433333	0.833333
5	"If you do not come to Shenzhen, sooner or lat...	144	-0.083333	0.033333	Shenzhen's GDP overtakes Hong Kong? Bureau of ...	107	0.000000	0.000000
6	"if you eat durian, you will kill yourself if ...	66	-0.625000	0.900000	Durian can't eat with anything, it's the same ...	78	0.000000	0.125000
7	"If you do not come to Shenzhen, sooner or lat...	144	-0.083333	0.033333	Shenzhen's GDP outpaces Hong Kong? Defending R...	83	0.000000	0.000000
8	"Frog frog? It's a fertility test! Let's play"...	58	0.000000	0.000000	A store in xianning contains "cotton"? A multi...	89	0.000000	0.000000
9	"How to discriminate oil from gutter oil by me...	60	0.000000	0.000000	A single piece of garlic can spot gutter oil? ...	106	-0.044643	0.157143

Figure 1: Sentiment Features

4.1.5 Similarity Features

We create the following similarity features

1. Cosine similarity between two titles using row level dictionary, and count vectors. This creates a score indicating presence of common words in the pair of articles. Our idea is that articles which are unrelated, will have fewer similar words that articles which agree or disagree with each other.(Sim). In our report we denote this feature as sim.

2. We observe that some recurring words such as rumour, fake, myth etc. occurs in title2 of articles with class label disagreed. Cosine similarity of each title with manually created set of words(which represents words such as rumor, myth and similar words), and calculating the difference between the two articles (indicating words are present in one of the articles but not both), would be an useful feature for separating articles which disagree with each other.We denote this as sim_m1.

4.1.6 Word2Vec/Doc2Vec Features

The Word2Vec method is a vector representation of words which takes into account the context in which a word appears. The Doc2Vec approach creates sentence level vectors using the Word2Vec internally. Our intuition is if we can, generate a Word2Vec/Doc2Vec representation of our titles, and generate various features based on this vector, it would have more information than a BOW/ TF-IDF representation, and would improve our model significantly. We plan to implement this model in future submissions.

4.2 Classification

We implemented various classifiers such as Random Forest, Naive Bayes, AdaBoost, and Logistic Regression. The generated features are fed to the classifier which assigns a class label to each data sample. We observed that generative classifiers such as Naive Bayes performs better than discriminative classifiers

like Support Vector Machine, Logistic Regression due to the nature of the feature we are generating. Based on the feature we have generated, getting a good separation is a challenging task and ensemble models such as the Random Forest classifier and AdaBoost performs best. Out of these two, Random Forest provided the best accuracy.

We also found the following insights on the classification approach

1. Oversampling :- As 68 percent(219313 out of 320552) of the training data belong to the unrelated class, 29 percent(92973 out of 320552) and 2.5 percent(8266 out of 320552) we faced the class imbalance problem, while training our model. Which basically means, our model learned features corresponding to unrelated class better than the other classes. To solve this problem, we copied instances of the agreed and disagreed classes multiple times to have a more balanced data set.

2. Confusion Matrix :- For evaluating the machine learning models, we split the training data into a 80-20 distribution, were we train it on 80 percent of the data and then evaluate our result on 20 percent of the data set. We use the classification report provided by sci-kit learn to evaluate the precision, recall, and F-1 score of our ML model, and use the confusion matrix to understand the classification patterns of our model.

	precision	recall	f1-score	support
0	0.62	0.68	0.65	43782
1	0.59	0.74	0.65	37145
2	0.65	0.33	0.44	28138
avg / total	0.62	0.61	0.60	109065

Figure 2: Confusion Matrix 1

```
cm
```

```
array([[29730,  9511,  4541],
       [ 9382, 27317,   446],
       [ 9210,  9574,  9354]])
```

Figure 3: Confusion Matrix 2

5 Results and Conclusion

5.1 Evaluation of proposed approach

The Kaggle competition provides a platform to evaluate the accuracy of the classifier. Our current model achieves an accuracy of 72.63% on test data set.

5.2 Feature Analysis

Our best result is when we use the similarity features based on the cosine similarity and Count Vectors approach. The visualization of the two features, and it's class wise distribution, is shown in Figure4.

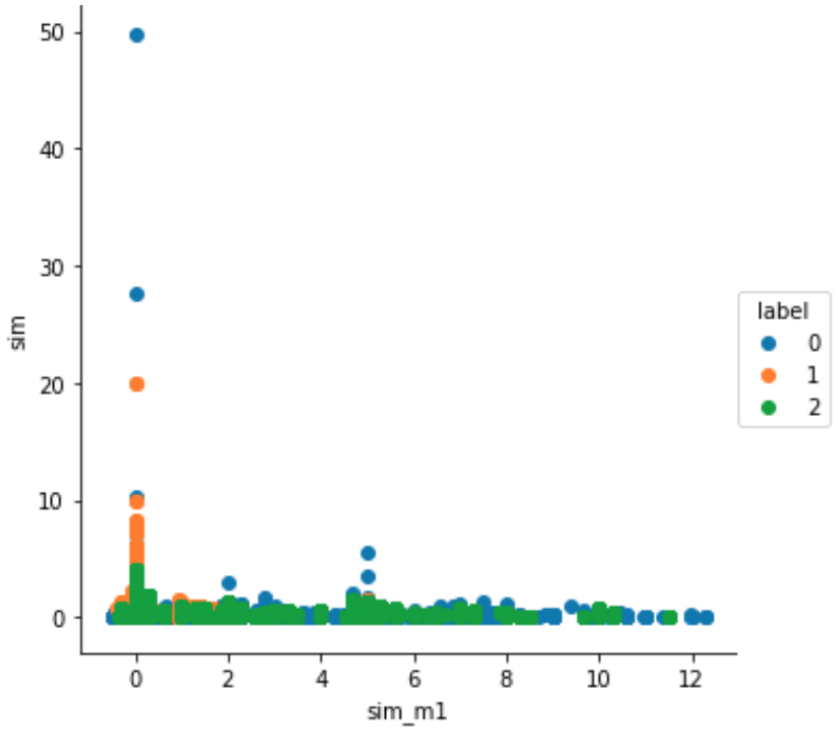

Figure 4: Similarity Features Scatter Plot

5.3 Conclusion

The class distribution in this data set is unbalanced. About 68% of the data samples have the label *unrelated*, about 29% have the label *agree*, and only 2.5% have the label *disagree*. Also, most of the features we generated are overlapping.

The current set of generated features is not enough for more accurate classification. Finding the meaning of the text content as a whole is an important issue in this project. For example, article B may have negative sentiment and contain words like refute or oppose, but the class label may be 'unrelated'. Also, differentiation between the 'agree' and 'disagree' classes requires more detailed approach as an agreeing view may also contain words like fake or rumour. For example, the article B may just say that article A is fake or rumour.

6 Discussion and Future Work

In this section, we describe our conclusions and the future approaches to improve the fake news classification models.

6.1 Discussion

The features for fake news detection can be classified into news content features and social context features. The news content features can be divided into Linguistic-based and Visual-based. There are three types of social context features, namely, user-based, post-based and network-based (Shu et al. [2017b]). User-based features are characteristics of the users having interactions with news on social media. Post-based features are unique features of posts based on the social response of users, such as stance, topic and credibility. Network-based features involve constructing specific networks between users involved in the social media posts.

Stance detection problem is a challenging problem, since the best performing features are not yet able to correctly classify the difficult data samples. Thus, more sophisticated machine learning techniques combined with deeper semantic understanding need to be developed. The stance should be determined on the basis of propositional content and social context as well, instead of relying on just textual features. Social context is also an important factor in determining whether a news is fake or just an opinion.

Additionally, if information related to user profiles or news posting platforms is obtained, the relationship between follower and followee for each fake news can be used to find the users or news posting platforms which are more likely to post fake news. Such information can be used for analyzing and extracting new features, visualize them and use them for creating novel models for identifying and classifying fake news.

6.2 Future Work

We believe that incorporating semantic understanding features will significantly improve the performance of the classifier. Extracting semantic understanding will help in finding the meaning of the title as a whole which will help in identifying the stance.

We also plan to implement Bidirectional-LSTM with encoder-decoder embedding to find a total combined representation of article A and article B. We will also perform stacking with combination of feature engineering approach and deep learning based approach.

We are also interested in constructing a network based approach for stance detection which will involve considering a set of articles talking about the same topic. The network measure would provide additional insights for the fake news classification.

References

Isabelle Augenstein, Tim Rocktäschel, Andreas Vlachos, and Kalina Bontcheva. Stance detection with bidirectional conditional encoding. *arXiv preprint arXiv:1606.05464*, 2016.

Sean Baird, Doug Sibley, and Yuxi Pan. Talos targets disinformation with fake news challenge victory, 2017.

Kuntal Dey, Ritvik Shrivastava, and Saroj Kaushik. Topical stance detection for twitter: A two-phase lstm model using attention. In *European Conference on Information Retrieval*, pages 529–536. Springer, 2018.

Bilal Ghanem, Paolo Rosso, and Francisco Rangel. Stance detection in fake news a combined feature representation. In *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*, pages 66–71, 2018.

Andreas Hanselowski, Avinesh PVS, Benjamin Schiller, Felix Caspelherr, Debanjan Chaudhuri, Christian M Meyer, and Iryna Gurevych. A retrospective analysis of the fake news challenge stance detection task. *arXiv preprint arXiv:1806.05180*, 2018.

Jundong Li, Kewei Cheng, Suhang Wang, Fred Morstatter, Robert P Trevino, Jiliang Tang, and Huan Liu. Feature selection: A data perspective. *ACM Computing Surveys (CSUR)*, 50(6):94, 2017.

Saif Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiaodan Zhu, and Colin Cherry. Semeval-2016 task 6: Detecting stance in tweets. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 31–41, 2016.

Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang, and Huan Liu. Fake news detection on social media: A data mining perspective. *ACM SIGKDD Explorations Newsletter*, 19(1):22–36, 2017a.

Kai Shu, Suhang Wang, and Huan Liu. Exploiting tri-relationship for fake news detection. *arXiv preprint arXiv:1712.07709*, 2017b.

Jen Weedon, William Nuland, and Alex Stamos. Information operations and facebook. *version*, 1:27, 2017.