The Fake News Challenge: Stance Detection Using Traditional Machine Learning Approaches

Article · September 2018

CITATION
1

READS
1,234

2 authors:

Razan Masood
University of Duisburg-Essen
3 PUBLICATIONS **5** CITATIONS
SEE PROFILE

Ahmet Aker
The University of Sheffield
90 PUBLICATIONS **723** CITATIONS
SEE PROFILE

The Fake News Challenge
Stance Detection Using Traditional Machine Learning Approaches

Razan Masood and Ahmet Aker
Department of Information Engineering, University of Duisburg-Essen, Duisburg, Germany
{*masood, aker*}*@is.inf.uni-due.de*

Keywords: Fake News, Stance Detection, Traditional Machine Learning, Feature Engineering

Abstract: Fake news has caused sensation lately, and this term is the Collins Dictionary Word of the Year 2017. As the news are disseminated very fast in the era of social networks, an automated fact checking tool becomes a requirement. However, a fully automated tool that judges a claim to be true or false is always limited in functionality, accuracy and understandability. Thus, an alternative suggestion is to collaborate a number of analysis tools in one platform which help human fact checkers and normal users produce better judging based on many aspects. A stance detection tool is a first stage of an online challenge that aims to detect fake news. The goal is to determine the relative perspective of a news article towards its title. In this paper, we tackle the challenge of stance detection by utilizing traditional machine learning algorithms along with problem specific feature engineering. Our results show that these models outperform the best outcomes of the participating solutions which mainly use deep learning models.

1 INTRODUCTION

Fake news is one of the controversially discussed issues lately. New York Times defines it as "a made-up story with an intention to deceive"[1]. Moreover, propaganda, conspiracy theories and other false stories have always been used in the media for a second gain like monetizing, political goals and opinion manipulation. Online services such as *factcheck.org* and *PolitiFact.com* perform manual fact checking to filter fake news.

The current online environments like social media create powerful tools to spread false stories extensively. As a result, journalists and fact checkers with their current strategies cannot label fake stories in real time before they are out of control. Automating those strategies is one solution to speed up the procedure. This kind of issues is considered to fit a machine learning task (Markowitz and Hancock, 2014; Hardalov et al., 2016; Jin et al., 2017).

Until lately, the work on fighting fake news is handled in many separate projects and studies. However, organizations like *FullFact.org* suggests to open collaborations between these projects to build a platform that provides a collection of tools to handle the different aspects of fact checking routines[2]. Similarly *Fake News Challenge* (*FNC-1*), which is an on-line competition, also suggests a solution for fake news detection to be composed by a collection of automated tools to support human fact checkers and speed up their processes. Stance detection is among the collection of these tools [3].

Stance detection has been proven to be useful in disinformation detection. (Jin et al., 2016) applied the stance to analyze the credibility propagation for news verification through building connections between micro-blogs (tweets) as supporting or denying each others' viewpoints. (Qazvinian et al., 2011) use the stance observed in tweets in a *Belief Classification* to classify false and true rumors, even though rumors checking is found to be different from news checking. Stance detection for fact checking in the emerging news has mostly been investigated in micro-blogs.

The stance detection task presented by *FNC-1* is about predicting the stance of one piece of text (news body) towards another (news headline). Particularly, it should predict whether the news body has the stances *Unrelated, Discuss, Agree* or *Disagree* to a news headline. Most of the teams participated in the

[1] https://www.nytimes.com/2016/12/06/us/fake-news-partisan-republican-democrat.html

[2] https://fullfact.org/media/uploads/full_fact-the_state_of_automated_factchecking_aug_2016.pdf

[3] http://www.fakenewschallenge.org/

FNC-1 including the winner team used deep learning approaches to solve the task. Although deep learning is a powerful technique and it has shown a great success in various tasks, the deep architectures is said to be short on providing more understandable results in terms of features extracted by the deep architecture and their performances. This drawback is the motivation in our work. Thus, instead of deep learning we use traditional machine learning approaches along with the appropriate feature selection/engineering and show that we can beat the deep learning approaches in an attempt to provide useful information about engaged features and their performances.

Thus our contributions are as follows:

- We provide a solution for the *FNC-1* task using traditional machine learning algorithms.
- We extract a range of different features that are useful for the stance detection task.
- We perform feature analysis and discuss different experimental training and testing settings which were important to obtain state-of-the-art results.
- We achieve a score of 82.1%[4] which is currently the best score achieved for the fake news stance detection task.

We first discuss related work, and then in Section 3 we describe the data and the scoring system. In Section 4 we introduce our method including the features and the machine learning approaches used to learn the models. Then, in Section 5, we discuss experimental settings followed by the results discussion in Section 6 and conclusion in Section 7.

2 RELATED WORK

The problem that is introduced in this paper was published first as a pure stance detection tool within a plan to employ it in a wider fake news detection platform. Many researches proposed methods which employ stance in disinformation detection. The veracity of claims were also predicted using the stance of articles and the reliability of their sources (Popat et al., 2017). Stance features were also employed in detecting the credibility of rumours, which are also defined to be unverified claims (Enayet and El-Beltagy, 2017). Moreover, Using Tweets publishing time and stances as the only features to model the veracity of tweets using Hidden Markov Models achieved high accuracy (Dungs et al., 2018). Some other cases that targeted rumours used also stance features to identify their veracity (Zubiaga et al., 2018).

Detecting stance of news articles is the most related work to our task. On this line the work of Ferreira & Vlachos addresses rumor debunking based on stance. The aim is to estimate the stance of a news headline towards its paired claim as *Observing, For* or *Against*. Linguistic features are extracted from each claim and headline pair (Ferreira and Vlachos, 2016).

The *FNC-1* task extends the work of Ferreira & Vlachos to predict the stance of a complete news article (body of an article) towards a title or headline paired with that article. For this task results of first three top systems have been announced. The first ranked team[5] approach is based on a 50/50 weighted average ensemble combining a gradient-boosted decision tree model fed with text based features from the headline and the body pair, and a deep learning model based on one dimensional Convolutional Neural Network (CNN) with Google News pre-trained vectors.

Unlike most of the approaches used by the participating teams described above we employ traditional machine learning techniques along with feature engineering. We investigate several features and determine their contribution towards the task. We also experiment with different training settings. Overall we show that our approach leads to slightly better results than those reported by deep learning strategies.

3 THE FAKE NEWS CHALLENGE

The fake news challenge (*FNC-1*) is a machine learning task which is a contribution between AI community, journalists and fact-checkers. It forms a basis for fighting fake news and aims to develop tools towards fake news detection. One of the tools is a stance detection tool which is the first interest of the challenge. The challenge is about predicting the stance of a news article (body of the article) towards its paired title or headline. In the following sections we describe the data and the scoring mechanism of *FNC-1*.

3.1 Data

The data used in the competition was extracted from Craig Silverman's Emergent dataset [6] which is part of a research project that employs rumor tracking in detecting misinformation. The dataset consists of 2595 articles that relates to 300 claims (headline) so that for each claim there are between 5 to 20 articles. These

[4]This score is calculated using the *FNC-1* scoring system

[5]https://github.com/Cisco-Talos/fnc-1

[6]http://www.emergent.info/

articles are labeled manually by journalists as agree, disagree or discuss the claims they are paired with.

The *FNC-1* organizers mixed and matched the article bodies and their headlines, and used the labels relative to the claims. They got 75,119 labeled pairs as the following:

- *Unrelated*: The topic of the headline is different from the topic of the article body.
- *Discuss*: The body observes the headline's claim neutrally without taking a position.
- *Agree*: The body confirms the headline's claim.
- *Disagree*: The body refutes the headline's claim.

The resulted pairs were divided by *FNC-1* organizers into 49,972 pairs as training data and 25,147 pairs for testing. The training dataset was a match between 1648 unique headlines and 1683 unique article bodies, whereas the test dataset is a match between 880 unique headlines and 904 unique article bodies with no overlaps between the splits. In addition, the test data used to finally evaluate the competitors was supplied with an additional 266 pairs that the organizers derived and labeled using Google News articles. The headline's length ranged between 1 - 40 words with an average of 11 words. While the article body length ranged between 3 - 4800 words with an average of 350 words. The training dataset is highly unbalanced with class distribution as the following: 73.13% unrelated, 17.83% discuss, 7.36% agree and 1.68% disagree.

3.2 FNC-1 Scoring System and Baseline Classifier

FNC-1 scoring system adds 0.25% score for each pair classified correctly as *Unrelated*. The score is increased by 0.25% if the pair was related and was classified as any of *Discuss, Agree* or *Disagree* classes. If the pair was correctly classified as *Discuss, Agree* or *Disagree*, the score is increased to 0.75%. We consider the approach that won the *FNC-1* as our baseline system. This system scored *82.02%* according to the *FNC-1* scoring system.

4 METHOD

In our methodology we apply traditional machine learning approaches, specifically, L1-Regularized Logistic Regression provided by LibLINEAR (Fan et al., 2008) using WEKA (Hall et al., 2009) and Random Forest classifier from the same WEKA toolkit. Both approaches rely on feature engineering. Our feature engineering focuses on the article content and tries to find parts of it that would best describe the stance the article has towards the headline.

The data provided for the *FNC-1* stance detection task is limited to articles' text with no reference to sources, writers or any explicit meta data. Given this, the features we relied on are only linguistic features. In the following sections we describe our features in detail.

4.1 Headline Features

- **Headline length (H-Len)** This is equal to the number of words in the headline.
- **Headline contains question mark (H-Q)** A feature indicating whether a headline contains a question mark or not (0 or 1).

4.2 Article Content Features

We split each article content into a heading, middle, and tail parts based on the sentences[7]. The motivation behind this splitting is that most news articles are written in a specific style in which the article beginning (heading) introduces the main argument(s) that the entire article wants to convey to the users, the body part (middle) provides more detailed information about the argument(s) made earlier and a conclusion towards the end (tail) summarizing what is detailed in the body. We have experimented with different splitting strategies however, dividing the entire article into first 5 sentences (heading), 4 sentences from the tail and 10% of the middle sentences (min. 2 sentences)[8] gave us best performance. In the following we explain the features extracted from these parts.

- **Bag of words (BoW)**: We extract uni-grams and bi-grams from the heading and tail parts of the article. However, we retain only the 500 most occurring n-grams and delete all the remaining ones.
- **Root distance (Root-Dist)**: This feature is calculated similar to the study of (Ferreira and Vlachos, 2016). However, for our case we compute 3 different features (feature vectors), i.e. one for the heading part, one for the middle and one for the tail part. For each sentence in each part we parse it using Standard CoreNLP parser and compute its root distance to pre-collected words list obtained from related work (*Discuss* or *Refute* words).

[7] Sentence splitting has been performed using The Stanford CoreNLP tools (Manning et al., 2014)

[8] We start taking from the median, then left and right of it until we have reached our threshold.

- **Sentiments**: For each sentence in each article part we compute its sentiment score. The tool used is Stanford Sentiment (Socher et al., 2013) which gives each sentence a score between 0 (high positivity) to 4 (high negativity).
- **Sentence length (Sentence-Len)**: This feature indicates the maximum and the average length of the sentences in the respective article parts.
- **Punctuation count (Punct)**: We use several punctuation such as dot, comma, etc. and for each of them we compute how many times it appears in the entire article (not only in the three parts).
- **Lemma count**: We remove all stop words from the headline and lemmatize the remaining words. For each sentence in each article part we count the occurrences of each lemma that also appears in the headline and take the sum of all lemma occurrence counts as a lemma count feature. We also do this for the entire article regardless of the mentioned article split boundaries.
- **Character grams (Ch-Grams)**: We build sets of character sequences of lengths 4, 8 and 16 from the headline. For each character sequence set we count how many times the sequences appear in each sentence of each article part. Each sentence is assigned three count values each indicating how many times the sentence includes any sequence from the respective length category. We use lemmatized text before building the character sequences and also remove all the stop words.
- **Word2Vec Similarity (W2Vec-Sim)**: For this feature, a vector space representation of both the headline and each article part is computed using word embedding (Mikolov et al., 2013). For the embedding, we used Google's Word2Vec pretrained words and phrases from Google News [9]. Once the embedding vectors are obtained we compute the cosine similarity between the given vectors.
- **Word grams (N-Grams)**: This is similar to the *Ch-Grams* feature however, instead we take word sequences of lengths 2, 4, 8 and 16.
- **Hypernyms similarity (Hyp-Sim)**: We use WordNet 3.1 (Miller, 1995) and collect hypernyms from the first synset of nouns and verbs. The nouns and verbs are taken from the headline, article heading and article tail. For the collected hypernyms we build word embedding vectors and compute similarities between title-article heading and title-article tail using cosine.
- **Cosine similarity (Cos-Sim)**: This feature computes the cosine similarity of the headline to each sentence in each article part. The vector values are word counts. Before computing we take lemmas of the words and remove stop words.
- **Paraphrase alignment (ppdb)**: This feature captures an alignment score calculated between two texts depending on the Paraphrase Database (Pavlick et al., 2015) and the Kuhn-Munkres algorithm (Kuhn, 1955; Munkres, 1957). It is computed between words from the headline and words from a sentence in each article part. This feature is calculated similar to (Ferreira and Vlachos, 2016).
- **Subject, Verb and Object triples entailment (SVO)**: This feature indicates the entailment relations between the subject, verb and object triples of the headline and each sentence in each article part. The entailment is again found using the paraphrase database (Pavlick et al., 2015). This feature is computed as in (Ferreira and Vlachos, 2016) work but instead of indicating the entailment with 0 or 1 we count how many sentence in each article part have this entailment relationship.
- **Negation (Neg)**: We use the Hungarian algorithm (Kuhn, 1955) to align words between the headline and words from each sentence from the article. Then we check for each aligned word pairs whether one of them is the negation of the other according to (Ferreira and Vlachos, 2016). Each sentence is assigned a counter indicating how many negated pairs it contains. We compute this feature for each sentence in the entire article.
- **Word Overlap score (W-overlap)**: For this feature we compute an overlap score between the headline and the body's heading as well as between headline and tail. The method is based on extracting all possible sub-strings from these parts and then finding the longest matching sub-strings. The score is calculated by summing up the square lengths of these matches.
- **Bias count (Bias)**: Based on a bias lexicon as in (Recasens et al., 2013) and (Allen et al., 2014), we compute how many bias lexicon entries appear in the entire article as well as in the headline.

5 EXPERIMENTAL SETTING

As noted in section 3.1, the data has four different class labels: *Unrelated*, *Discuss, Agree* and *Disagree*. We trained our classifiers so that they predict one of these four labels. However, the performance of the

[9] https://code.google.com/archive/p/word2vec/

resulting models were below the baselines[10]. After manual inspection of the data we realized that the articles labeled differently were similar in tone towards the headline and thus difficult for a multi-class labeler to predict the right class. Furthermore, the data is unbalanced and contains mostly *Unrelated* pairs and relatively few pairs from the other classes. To overcome these issues we experimented with different training strategies without modifying the training and testing settings defined by *FNC-1*:

- **2-Steps Classifier**: We first train the classifier to distinguish only between *Unrelated and Related* pairs, where *Related* represents all the categories *Discuss, Agree, Disagree*. Next, we train a second classifier on the pairs labeled with *Discuss, Agree, Disagree*. For testing, we first run the first classifier on the entire testing data. Any article-headline pair classified as *Related* is further analyzed with the second classifier to further classify it as *Discuss, Agree* or *Disagree*.
- **3-Steps Classifier, setting 1**: We further split the classification of the *Related* classes and create three classifiers. We keep the first step as it is in the previous 2-step classifier setting (classification for *Unrelated* and *Related*. Then we train a classifier to predict the classes *Discuss* and *Non-Discuss*, where *Non-Discuss* category stands for the original categories *Disagree, Agree*. For the third step we use a 2-way classification for the remaining categories *Disagree, Agree*.

  For testing we again run first the first classifier to split the data into *Unrelated* and *Related* categories. After, the *Related* data pairs are further classified to obtain *Discuss* and *Non-Discuss* article-headline pairs. Finally, for the Non-Discuss pairs we further detail their actual classes using the third classifier and obtain the *Disagree Agree* classes.
- **3-Steps Classifier, setting 2**: We keep the first step as it is, but we used a 2-way classification in the second step for the categories *Disagree, Non-Disagree*. The *Non-Disagree* category represents the original categories *Discuss, Agree*. For the third step we use a 2-way classification for the remaining categories *Discuss, Agree*.

In all settings for the first two steps we use an L1-regularized logistic regression classifier (Fan et al., 2008). For the third step we use a Random Forest classifier with 100 trees (Breiman, 2001). In each step we used different sets of features. Table 1 shows the features used in each step.

[10]Classifier predicting all classes led to 77.04% *FNC-1* score.

Table 1: Classifier steps and the features used in each step.

1st step	2nd step	3rd step
W-Overlap	H-Q	H-Q
Lemma Count	BoW	BoW
Ch-Grams	Root-Dist	Root-Dist
N-Grams	Neg	Neg
Cos-Sim	SVO	SVO
Hyp-Sim	Sentiments	Sentiments
PPDB	PPDB	PPDB
W2Vec-Sim	W2Vec-Sim	Sentence-Len
	Bias	Bias
		Punct

6 RESULTS AND DISCUSSION

Our overall results are shown in Table 2. We report, as in *FNC-1* challenge, the results using the *FNC-1* score. We also compute accuracy. From the table we see that the best results are obtained with the 3-step classifiers and setting 2. However, we found no difference in terms of significance to our other settings.[11] From the table we also see that the performance of our classifier (3-steps classifier setting 2) is better than the one of the best system participated in the *FNC-1* task. Both *FNC-1* as well as accuracy figures are better than those of the best performing baseline.[12]

Tables 3 and 4 show the confusion matrices of the best baseline and our 3-steps classifier with setting 2. According to the matrices, the 3-steps classifier in setting 2 predicts more correct *Discuss*, *Disagree*, *Unrelated* pairs. The baseline, on the other hand, performs better on the *Agree* class.

6.1 Features Analysis

As shown in Table 5 best results are obtained when all features are used. We aimed to understand the contribution of each feature to the overall results. Thus we removed a feature at a time, trained the classifiers with the remaining features and tested on the testing data. The difference in results are captured using paired t-test and a p-value of $p < 0.0028$[13] In the results we see only a significance drop when we remove the *BoW* feature, in all other settings the results are not significantly different from when there is no feature omission. However, in all removal cases there is a moder-

[11]Significance test is performed using student t-test.

[12]Again in the results we did not find any indication for significance.

[13]When conducting multiple analyses on the same dependent variable, the chance of achieving a significant result by pure chance increases. To correct for this we did a Bonferroni correction on the p-value. Results are reported after this correction.

Table 2: N-Steps classifiers and Baseline.

	Winning Baseline	2-Step Classifier	3-Steps Classifier setting 1	3-Steps Classifier setting 2
Unrelated	0.98	0.98	0.98	0.98
Discuss	**0.76**	**0.76**	0.75	**0.76**
Agree	**0.54**	0.52	0.49	0.52
Disagree	0.04	0.05	0.07	**0.1**
Accuracy	89.1	89.1	88.8	**89.18**
FNC- Score	82.02	82.0	81.53	**82.10**

Table 3: Best baseline Confusion Matrix. A, DA, DC and U stands for *Agree, Disagree, Discuss* and *Unrelated* respectively.

	A	DA	DC	U
A	1114	17	588	184
DA	275	13	294	115
DC	823	6	3401	234
U	35	0	203	18111

Table 4: 3-Steps Classifier with setting 2 Confusion Matrix. A, DA, DC and U stands for *Agree, Disagree, Discuss* and *Unrelated* respectively.

	A	DA	DC	U
A	947	29	799	128
DA	181	39	343	134
DC	589	28	3558	289
U	10	2	219	18118

ate drop in the results indicating that every feature has some contribution to the final results.

We also removed combinations of features from the entire set of features used in our final model to show the effect of more than one feature removed at once. The selection of different combinations is chosen according to the relatedness of features. We list them in groups:

- *Group A*: Is a group of features used in the first step for distinguishing *Related* and *Unrelated* classes, namely *Ch-grams*, *N-grams*, *Lemma Count* and *W-overlap* (see Table 1 for the set of features used in the first step). When removing *Group A* features, the number of correctly classified instances as *Unrelated* reduces the most (from 18118 to 18034), hence reducing the correctly classified instances as *Discuss*. See confusion matrices in Tables 6 and 4 for comparison.
- *Group B*: This group holds features related to similarity and entailment, namely *PPDB*, *Hyp-Sim*, *W2Vec-Sim* and *Cos-Sim*. They have lower effect on *Related* and *Unrelated* but greater effects on the *Agree* (reduction from 947 to 913) and *Disagree* (reduction from 39 to 26). See confusion matrices 7 and 4.

Table 5: Accuracy, and FNC-1 score when using all features compared to results when removing features one by one accordingly.

Features	Accuracy	FNC-1 Score
All Features	**0.891**	**82.1**
- BoW*	0.870	78.53
- Lemma Count	0.888	82.07
- Ch-Grams	0.888	81.42
- N-Grams	0.890	82.00
- Hyp-Sim	0.891	81.93
- Cos-Sim	0.890	81.86
- W2Vec-Sim	0.891	82.01
- ppdb	0.890	81.70
- w-overlap	0.891	82.02
- H-Len	0.891	81.97
- Root-Dist	0.888	81.57
- SVO	0.889	81.59
- Neg	0.891	81.85
- Sentiments	0.887	81.37
- Bias	0.891	82.00
- Punct	0.889	81.66
- Sentence-Len	0.887	81.41
- Tittle-Q	0.891	81.86
- group A	0.885	81.19
- group B	0.886	80.87
- group C	0.888	81.38
- group D	0.890	81.89

Table 6: Group A: Features without *Ch-grams*, *N-grams*, *Lemma Count* and *W-overlap*.

	A	DA	DC	U
A	934	27	797	145
DA	180	42	342	133
DC	561	25	3493	385
U	16	1	298	**18034**

- *Group C*: This group contains *SVO*, *Neg*, *Root-Dist* and *PPDB* features. Confusion matrix 8 shows that by removing these features, the categories *Disagree* and *Discuss* are mostly affected.
- *Group D*: This group contains *Punct*, *Bias*, *Sentence-Len* and *T-Quest* features. Removing this combination has a greater effect on the *Agree* category. See Table 9.

Figures for the accuracy and *FNC-1* metrics after re-

Table 7: Group B: Features without *PPDB*, *Hyp-Sim*, *W2Vec-Sim* and *Cos-Sim*.

	A	DA	DC	U
A	**913**	30	783	177
DA	165	**26**	328	178
DC	563	30	3485	386
U	9	1	243	18096

Table 8: Group C: Features without SVO, Neg, Root-Dist and PPDB.

	A	DA	DC	U
A	939	26	793	145
DA	212	**29**	310	146
DC	626	35	**3486**	317
U	24	1	212	18112

moving these group features are shown in Table 5. Overall the removal of all group features lead to decrease in performance. However, similar to the single features cases the decreases are only moderate without significance relevance.

6.2 Discussion

Overall we have seen that our 3-step classifier in setting 2 outperforms the state-of-the-art system that participated in the *FNC-1* challenge. Although the differences in the results are only moderate, nevertheless, they show that it is possible to beat state-of-the-art results with feature engineering as well as traditional machine learning approaches. Furthermore, tackling the problem in hand with such an approach has the advantage that, unlike deep learning approaches, enables feature extraction and later feature analysis. In our case, we carefully picked our features and investigated settings including finding article parts and classification steps where they shine best.

Feature analysis shows that removing any single feature leads to some drop in performance compared to the results when all features are used. The significant drop happens when we remove the *BoW* feature. The *BoW* feature includes uni-grams and bi-grams extracted from the article heading as well as from the article tail. Thus, it aims to capture what is in those article parts in terms of vocabulary. Those areas of the article introduce and summarize arguments. The chance is very high that they capture the claim introduced in the headline. Indeed the results confirm this phenomenon with a significant drop when removing this feature.

We also grouped features and removed them altogether from the complete feature set. The overall drop in terms of performance was moderate. However, in the confusion matrices we have seen that each feature group has its strength in a specific category or class. Group A features help in the relatedness task (step one of the classification) whereas the other groups find their shining points at later steps and address *Agree*, *Disagree* and *Discuss* classes.

Table 9: Group D: Features without *Punct, Bias, Sentence-Len* and *T-Quest.*

	A	DA	DC	U
A	**886**	26	863	128
DA	173	34	356	134
DC	556	27	3592	289
U	14	2	215	18118

Finally we performed error analysis on the final classifier results. We observed the following points:

1. There is ambiguity in *Disagree* definition. Example: pair: {headline: "Justin Bieber Helps Defend Russian Fisherman...", body ID: 2373}. This pair's correct class is "Disagree", but it is classified as *Unrelated* by our classifier. In this example there is no mention of *Justin Bieber*. The article itself is about a Fisherman being attacked by a bear. However, there is no disagreement about the topic that is introduced in the headline. Thus according to the definition for the category *Disagree*, this pair should be classified as *Unrelated*.
2. Detecting disagreement is hard in some cases because it depends on the implicit meaning of the article. As an example, the pair:{Headline: "People Actually Believed Argentina's President Adopted A Jewish Boy...", body ID: "2382",} This pairs correct classification is *Disagree*, but it is classified as *Discuss* by our classifier. The article talks about passing a law to stop some act of Argentina's people and it does not refute explicitly what it is in the headline.
3. Detecting unrelated titles to their paired articles is critical when the article uses most of the words mentioned in the title.
4. In most cases there is no clear indications for differentiating between the classes *Agree* and *Discuss* which makes them hard to judge by our classifier. Most of the classifier errors are due to this phenomenon. See Table 4.

7 CONCLUSIONS

In this paper we re-investigated the Fake News first challenge of stance detection using traditional machine learning and feature engineering approach. Us-

ing this method we scored better than the first winner's deep learning model.

We performed feature analysis by removing a feature at a time but also groups of features. Any removal led to moderate performance drop. The significance drop happened when the BoW feature was removed. This feature contains uni-grams and bi-grams extracted from the article heading and article tail. As discussed both parts either introduce or summarize arguments and are likely to capture what is said in the headline. Overall every feature plays a role in the classification. We showed that some features play role in the first step (distinguishing between related and unrelated pairs) and others play at discriminating between agree, disagree and discuss classes.

Our immediate future work will be to use stance to perform judgments about fake news. We will investigate how stance can be integrated for the fake news classification.

REFERENCES

Allen, K., Carenini, G., and Ng, R. (2014). Detecting disagreement in conversations using pseudo-monologic rhetorical structure. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1169–1180.

Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

Dungs, S., Aker, A., Fuhr, N., and Bontcheva, K. (2018). (in press). can rumour stance alone predict veracity? In *Proceedings of COLING 2018, The 27th International Conference on Computational Linguistics*.

Enayet, O. and El-Beltagy, S. R. (2017). Niletmrg at semeval-2017 task 8: Determining rumour and veracity support for rumours on twitter. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 470–474.

Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., and Lin, C.-J. (2008). Liblinear - a library for large linear classification. The Weka classifier works with version 1.33 of LIBLINEAR.

Ferreira, W. and Vlachos, A. (2016). Emergent: a novel data-set for stance classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. ACL.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2009). The WEKA data mining software: an update. *SIGKDD Explorations*, 11(1):10–18.

Hardalov, M., Koychev, I., and Nakov, P. (2016). In search of credible news. In *International Conference on Artificial Intelligence: Methodology, Systems, and Applications*, pages 172–180. Springer.

Jin, Z., Cao, J., Zhang, Y., and Luo, J. (2016). News verification by exploiting conflicting social viewpoints in microblogs. In *AAAI*, pages 2972–2978.

Jin, Z., Cao, J., Zhang, Y., Zhou, J., and Tian, Q. (2017). Novel visual and statistical image features for microblogs news verification. *IEEE Transactions on Multimedia*, 19(3):598–608.

Kuhn, H. W. (1955). The hungarian method for the assignment problem. *Naval Research Logistics (NRL)*, 2(1-2):83–97.

Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J. R., Bethard, S., and McClosky, D. (2014). The stanford corenlp natural language processing toolkit. In *ACL (System Demonstrations)*, pages 55–60.

Markowitz, D. M. and Hancock, J. T. (2014). Linguistic traces of a scientific fraud: The case of diederik stapel. *PloS one*, 9(8):e105937.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Miller, G. A. (1995). Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41.

Munkres, J. (1957). Algorithms for the assignment and transportation problems. *Journal of the society for industrial and applied mathematics*, 5(1):32–38.

Pavlick, E., Rastogi, P., Ganitkevitch, J., Van Durme, B., and Callison-Burch, C. (2015). Ppdb 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 425–430.

Popat, K., Mukherjee, S., Strötgen, J., and Weikum, G. (2017). Where the truth lies: Explaining the credibility of emerging claims on the web and social media. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 1003–1012. International World Wide Web Conferences Steering Committee.

Qazvinian, V., Rosengren, E., Radev, D. R., and Mei, Q. (2011). Rumor has it: Identifying misinformation in microblogs. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1589–1599. Association for Computational Linguistics.

Recasens, M., Danescu-Niculescu-Mizil, C., and Jurafsky, D. (2013). Linguistic models for analyzing and detecting biased language. In *Proceedings of ACL*.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., and Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Zubiaga, A., Aker, A., Bontcheva, K., Liakata, M., and Procter, R. (2018). Detection and resolution of rumours in social media: A survey. *ACM Computing Surveys (CSUR)*, 51(2):32.