ELEC-E5550 — Statistical Natural Language Processing
Aalto University

FAKE NEWS CHALLENGE STAGE 1 (FNC-I): STANCE DETECTION

Jun Zhang 801940
⟨*jun*.1.*zhang*@*aalto*.*fi*⟩

Mingjie Ye 802169
⟨*mingjie*.*ye*@*aalto*.*fi*⟩

April 28, 2020

1 Introduction

Fake news detection is an interesting research topic yet challenging social problem. This is especially the case of at the moment. As the COVID-19 pandemic continues to simmer all around the world, we see a explosion of all kinds of rumors and fake news. This not only poses more difficulty for the public and the government to work together to fight against the pandemic, but also incites a lot of hatred between different countries and societies. How to tackle this problem with natural language processing has been an important research topic in the recent years. Although it is still very difficult to build a fake news detection system with high accuracy, we are able to break down the fake news detection into a few stages. Stance detection is commonly recognized as the "first" stage of fake news detection challenge.

Stance detection in this project refers to estimate the relative level (namely "stance") of one pair of text relative to a topic, claim or issue. There are different versions of stance detection. We choose the one proposed in an online challenge named Fake News Challenge(FNC-1)[1]. It is an extension of Ferraira Vlacho's work[2]. It estimates the stance of body text from a news article relative to a headline. Specifically, the body text may agree, disagree, discuss or be unrelated to the headline.

In this project, we explored different methods and features other than the baseline algorithm provided in the challenge. The features we explore include statistical NLP features such as ngram and handcraft features. We have tried out different random forest based algorithms such as GBDT, LightGBM, XGBoost, etc. In addition, we also experimented with end-to-end pretrained model such as RoBerta[3]. Pretrained models have been proved to be the best performing on most NLP tasks. Without extracting and modeling the features explicitly, models which are pretrained on large datasets with transformer components are able to learn the deep context embedding of text. We just need to fine-tune these models on small task-specific dataset with addition classification or inference layers when applying them to the downstream tasks. Without surprise, RoBERTa shows the best performance in our experiments. But the traditional statistical features extraction plus classifiers can still manage to detect the stance correctly at some level especially for the class unrelated and discuss.

2 Methods

In this project, we explore both statistical NLP methods and deep learning based methods. Subsection 2.1 describes the extracted features using the projects. Subsection 2.2 describes the classifiers we have tried. Subsection 2.3 describes one pretrained model we have used in the projects.

2.1 Feature Engineering

1) N-gram co-occurrence between the headlines and the article bodies

   We count how many times an n-gram[4] of the title appears in the entire body, and intro paragraph. Both word grams and char grams are taken into consideration. In

addition, we extract word grams and char grams with different lengths as part of the features.

2) Overlapping token ratio

For each text string including headlines and bodies, we first lowercase the string and remove any non-alphanumeric characters as well as the stop words. We further apply lemmatization to each token. Then we calculate the ratio of intersections to headlines token sets and bodies token sets to their union.

3) Sentiments of headlines and bodies

We tried different ways to get the sentiments of headlines and bodies. This includes detecting the existence of highly polarized words from a lexicon, using models trained on product/movie reviews and lexicon and rule-based method such as VADER[5]. Doing sentiment analysis on news data is more challenging than doing that on product reviews or twitter data because reporters tend to be objective when writing the articles, which greatly limits the use of emotional or highly polarized words. We trained transformer based model with movie review data due to the lack of labelled new data. However, this method does not seem to be better than detecting the existence of highly polarized words. In our experiments, VADER(Valence Aware Dictionary and sEntiment Reasoner) performs slightly better than the others. The advantages of using this method including not needing training data, good speed-performance tradeoff.

4) Keywords extraction

Keyword is one very important feature to classify if the headline and the body are talking related topics. We use TF-IDF algorithm to extract the top 20 n-grams. Tf-idf is one classical method to calculate the importance of the tokens to the documents. TF represents term frequency and IDF represents inverse document frequency. The formulas of the algorithm are as follow:

$$TF = \frac{occurrence\_of\_term}{total\_number\_of\_terms}$$

$$IDF = log(\frac{total\_number\_of\_documents}{number\_of\_documents\_containing\_the\_term + 1})$$

$$TF - IDF = TF \times IDF$$

2.2 Classifiers

In this section, we implemented seven classification algorithms to classify and compared the results based on a weighted scorer.

The classifiers we implemented in this case are random forest, Gradient Boosting Decision Tree (GBDT), Extra Trees, Descision Tree, AdaBoost, LightGBM aand XGBoost. The first five classifiers are based on sklearn.ensemble package.

After feature engineering, we had the features and labels. Therefore, we split 20% data as hold-out set (test set) and used k-fold cross validation (k = 10) to train.

2.3 End-to-end methods

Besides the above methods, we also experimented with some end-to-end methods. RoBERTa[3] performs the best in our case. RoBERTa is one popular variant of BERT[6] model. It uses the same bidirectional transformers models as BERT. The improvements are mainly done on the way of training. Frist, it uses dynamic masking instead of static masking. Second, it removes the next sentence prediction task existing in the BERT training process. Third, it uses bigger batch size and much larger dataset for pretraining. According to numerous experiments done by the original authors and other researchers, it outperforms BERT in most NLP tasks. We observed the same trend in our experiments. RoBERTa is also the best performing model in this stance detection task.

3 Experiments

3.1 Data

We use the data provided in this repo. The relation between headlines and stances is many-to-one. There are 49972 headlines and 1683 article bodies for training set and 25413 headlines and 904 bodies for test set.

3.2 FNC-1 score

In order to evaluate, we used a scorer provided in the FNC-1 dataset besides accuracy, precision, recall f1-score and confusion matrix. We can call the metric as FNC-1 score. The score system has two levels. Because the related/unrelated classification task is expected to be much easier and is less relevant for detecting fake news, it is given less weight in the evaluation metric. The Stance Detection task (classify as agrees, disagrees or discuss) is both more difficult and more relevant to fake news detection, so it is to be given much more weight in the evaluation metric.

3.3 Preprocessing

As mentioned in the section 2.1, the preprocessing contains the following steps.

- Lowercase
- Remove non-alphanumeric characters
- Remove stopwords
- Lemmatization

3.4 Feature extraction

- For N-gram char grams, we choose lengths of 2, 4, 8, 16. For N-gram word grams, we choose lengths of 2, 3, 4, 5, 6. For counting the co-occurrence, we aslo distinguish if the co-occurence happens in the first few paragraphs or the later paragraphs.

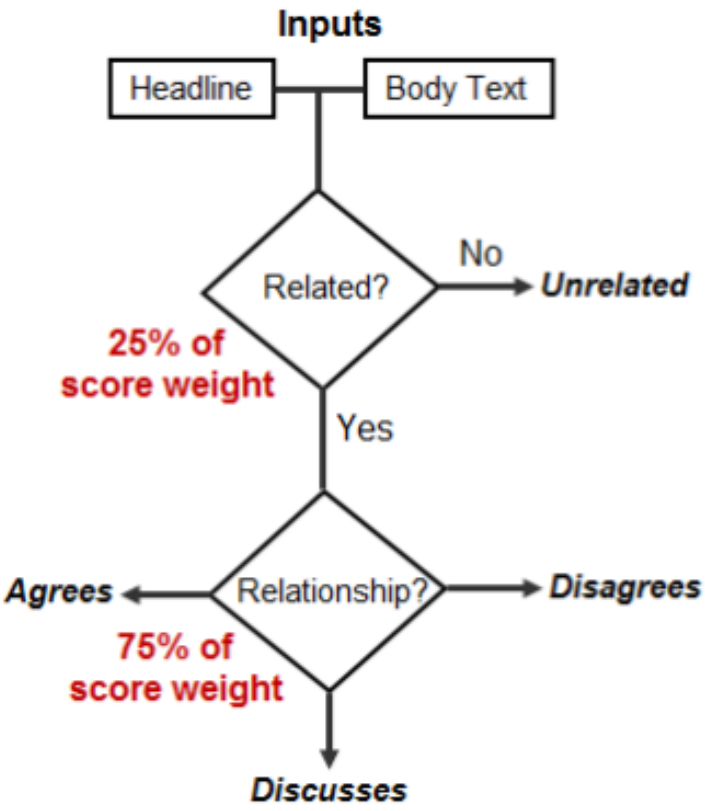

Figure 1: The Scorer

- For sentiment analysis of headlines and bodies, we use VADER provided by NLTK.
- To compute the keywords of bodies with TF-IDF, we first count frequencies of the unigrams and bigrams and output the top 20 grams according to the Tf-idf values.

3.5 Classifiers

After getting the features, we use 10-fold CV to train the models. The code below shows the parameters of these classifiers.

```
# 1. Random Forest
# clf = RandomForestClassifier(n_estimators=200)
# 2. GDBT
# clf = GradientBoostingClassifier(n_estimators=200, random_state=0,
    learning_rate=0.1, verbose=True)
# 3. ExtraTreesClassifier
# clf = ExtraTreesClassifier(n_estimators=200, max_depth=None,
    min_samples_split=2, random_state=0)
# 4. DecisionTreeClassifier
# clf = DecisionTreeClassifier(max_depth=6, min_samples_split=2, random_state
    =0)
# 5. AdaBoostClassifier
# clf = AdaBoostClassifier(n_estimators=200, learning_rate=0.1, random_state
    =0)
# clf.fit(X_train, y_train)

# 6. LightGBM
# train_data = lgb.Dataset(X_train, label=y_train)
# test_data = lgb.Dataset(X_test,label=y_test)
# param = {'num_leaves':63, 'learning_rate': 0.1, 'num_trees':600, 'max_depth
    ': 7, 'num_class':4, 'objective': 'multiclass'}
# num_round = 10
# bst = lgb.train(param, train_data, num_round, valid_sets=[test_data],
    early_stopping_rounds=5)
# clf=bst
# predicted=[]
# for a in clf.predict(X_test):
#     b=a.tolist()
#     predicted.append(LABELS[b.index(max(b))])

# 7. XGBoost
train_data = xgb.DMatrix(data=X_train, label=y_train)
test_data = xgb.DMatrix(data=X_test,label=y_test)
param = {'max_depth': 6, 'eta': 0.1, 'num_class': 4, 'objective': 'multi:
    softprob'}
watchlist = [(test_data, 'eval'), (train_data, 'train')]
num_round = 10
bst = xgb.train(param, train_data, num_round, watchlist)
```

For python implementation, please tend to *'fnc_kfold.py'* for further details. Take LightGBM model for example, we calculated its confusion matrix, precision, recall, f1-score, micro

average, macro average and weighted average. The training and testing logs were all saved in *'./results/'* path.

Eval_report

	precision	recall	f1-score	support
agree	0.58	0.13	0.21	762
disagree	0.50	0.01	0.02	162
discuss	0.65	0.86	0.74	1800
unrelated	0.96	0.98	0.97	6898
micro avg	0.88	0.88	0.88	9622
macro avg	0.67	0.50	0.49	9622
weighted avg	0.86	0.88	0.85	9622

Scores on the dev **set**

	agree	disagree	discuss	unrelated
agree	100	1	580	81
disagree	15	2	131	14
discuss	55	1	1544	200
unrelated	3	0	108	6787

Score: 3538.5 out of 4448.5 (79.54366640440598%)

Scores on the test **set**

	agree	disagree	discuss	unrelated
agree	139	6	1461	297
disagree	29	1	428	239
discuss	164	6	3629	665
unrelated	7	1	369	17972

Score: 8785.5 out of 11651.25 (75.40392661731573%)

3.6 RoBERTa

We use Simple Transformers to fine-tune the pretrained RoBERTa-base on our dataset. The initial learning rate is 3e-5. Batch size is 8. MAX_SEQUENCE_LENGTH is 512. Warmup and early stop are applied. We trained the model for 8 epoches and achieved a FNC-1 score of 89.257% and 76.987% of Macro F1.

4 Results

4.1 Effects of different features

We explored how different features affect the performances. The following table 1 shows the differences on FNC-1 score. Results show option 4 performs the best. All the experiments are done with the same GBDT classifier.

Table 1: Effects of different features

Feature Extraction	1	2	3	4
N-gram	✓	✓	✓	✓
Overlapping ratio	✓	✓	✓	✓
Sentiment(keywords)	✓		✓	
Sentiment(VADER)		✓		✓
Keywords			✓	✓
Dev	79.544%	80.291%	80.847%	80.819%
Test	75.089%	75.286%	76.500%	76.633%

4.2 Effects of different classifiers

The table 2 shows the scores of different classifiers on development set and test set. From the results, we can find the scores are pretty similar and LightGBM attains the highest score. By combining the best features option 4 with LightGBM classifier, we get the results with 76.003% of FNC-1 socre. It is slight worse than features option 4 + GBDT, which shows that feature engineering has bigger influence on the performance.

4.3 Best performance

The best performance is attained by fine-tuned RoBERTa model. It is a end-2-end solution so there is no need to extract the handcrafted features explicitly. This is expected as we have seen the power of BERT family models on different NLP tasks in the recent years. The reasons are mainly two-fold: (1) Two directional transformer building block enables the model to capture long distance relation and in-depth context meaning, (2) The model is pretrained on over 100G data.

The result of this model is shown below.

```
F1 macro: 76.987%
F1 agree: 69.469%
F1 disagree: 54.074%
F1 discuss: 85.181%
F1 unrelated: 99.224%
```

Table 2: Effects of different classifiers

Classifier	Dev	Test
Random Forest	79.043%	74.219%
GDBT	79.515%	75.202%%
Extra Trees	78.082%	73.704%
Decision Tree	78.813%	74.809%
AdaBoost	78.296%	74.968%
LightGBM	79.543%	75.403%
XGBoost	79.009%	74.824%

Scores on the test **set**

	agree	disagree	discuss	unrelated
agree	1281	119	472	31
disagree	142	375	143	37
discuss	339	191	3834	100
unrelated	23	5	89	18232

Score: 10399.5 out of 11651.25 (89.25651754103637%)

Test report

	precision	recall	f1-score	support
0	0.72	0.67	0.69	1903
1	0.54	0.54	0.54	697
2	0.84	0.86	0.85	4464
3	0.99	0.99	0.99	18349
accuracy			0.93	25413
macro avg	0.77	0.77	0.77	25413
weighted avg	0.93	0.93	0.93	25413

5 Conclusion

Fake news is still an unsolved yet very important topic. In this project, we focus on stance detection task, which can be regarded as the first step of the fake news detection. A simple fake news detection system can be built by simply aggregating the predictions the stances of the news to a couple of news from highly credible media firms. In our experiments, the most recent pretrained model RoBERTa performs much better than the others. There is no

doubt this is the way to go if we want further improve the performance. Yet the traditional statistical features extraction plus classifiers can still manage to detect the stance correctly at some level especially for the class unrelated and discuss. We observe the same trend in the experiments of RoBERTa. While class agree and disagree are much more difficult to be classified correctly. Still, there are still much room for improvement by looking at the F1 macro. More research needs to be done.

6 Division of labor

We worked together on this project. The below shows our contribution.

Task	Jun Zhang	Mingjie Ye
Literature Review	40%	60%
Algorithm (Feature Engineering)	70%	30%
Algorithm (Classifier)	30%	70%
Algorithm (End-to-End)	55%	45%
Report	60%	40%

Figure 2: Contribution to the project

References

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

[2] Ferreira, William, and Andreas Vlachos. "Emergent: a novel data-set for stance classification." Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: Human language technologies. 2016.

[3] Liu, Yinhan, et al. "Roberta: A robustly optimized bert pretraining approach." arXiv preprint arXiv:1907.11692 (2019).

[4] Shannon, Claude E. "A mathematical theory of communication." Bell system technical journal 27.3 (1948): 379-423.

[5] Hutto, C.J. Gilbert, E.E. (2014). VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. Eighth International Conference on Weblogs and Social Media (ICWSM-14). Ann Arbor, MI, June 2014.

[6] Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018).

Appendices

The sourcecode of the projects can be checked via this repos.