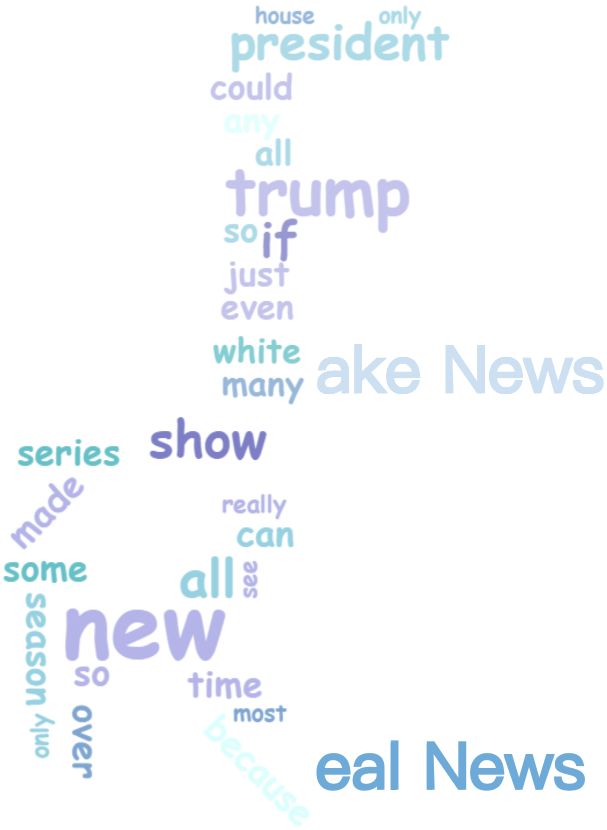
MM-DET : Study on Fake News Detection Based on Multiple Methods

2020/5

Watch it on GitHub and

Hit the star button!

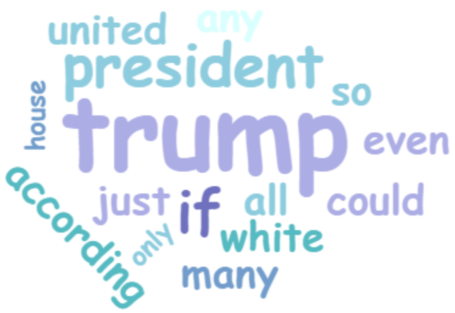


CONTENTS

- Background
- Data Preparation
- Dataset Analysis
- Word Embedding : TF-IDF
- Methods Overview
- Experiments
- Conclusion

Background

Social media has become a popular means for people to consume and share the news. At the same time, however, it has also enabled the wide dissemination of fake news, i.e.,news with intentionally false information, causing significant negative effects on society. To mitigate this problem, some researches of fake news detection have recently drawn a lot of attention. In this report, we try to resolve fake news detection by multiple methods.



Data Preparation

Dataset : Subset of Fake News Net dataset

4739 fake & 11193 real (gossip)

349 fake & 466 real (political)

Train : 75%   Val : 25%

Data Format : data.csv

Selected Keys : title, text, author and label (0/1)

Metric :

Accuracy = Correct classified samples / total samples

AUC : Area under ROC curve

- shows how well a binary classifier performs
- able to avoid problems caused by unbalanced positive/negative samples

{{ title : Michael Buble ...,
text : Michael Buble has announced his first studio album for two years after he put his career on hold when…,
author : ["Katie O'Malley"]…,
label : 0},
…}

Dataset Analysis : Word Frequency

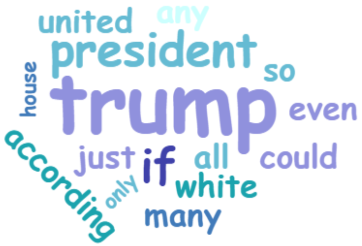
gossip fakenews

gossip realnews

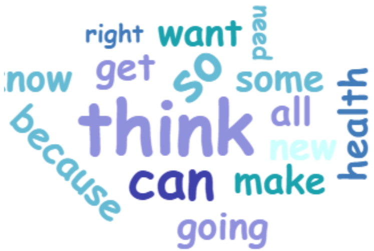
News Samples

1000 gossip

300 political

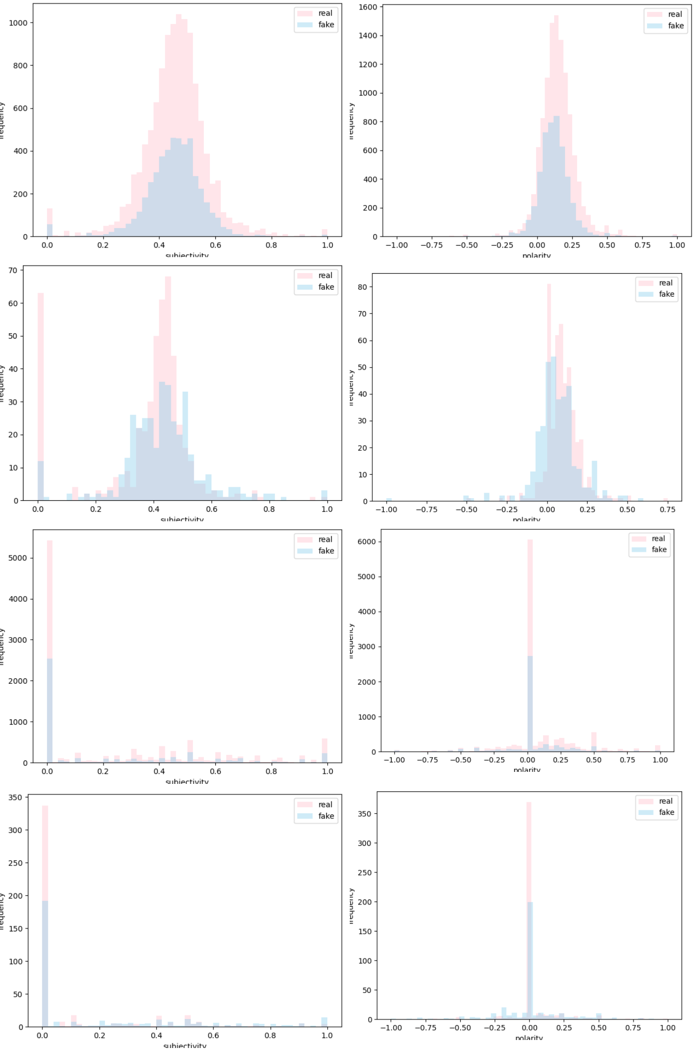


political fakenews



political realnews

- to be more valid , function words such as a , and , of , the , it , in … are left out (stop words)



Dataset Analysis : Sentiment

Text

Title

Polarity & Subjectivity of Text & Title

Package : TextBlob

Result : Similar Gaussian distribution

```
text = content['text']
title = content['title']
label = content['label']
real_pol = []
real_sub = []
fake_pol = []
fake_sub = []
for i, _ in tqdm(enumerate(text)):
    if label[i] == 0:
        blob = TextBlob(text[i])
        real_pol.append(blob.sentiment.polarity)
        real_sub.append(blob.sentiment.subjectivity)
    else:
        blob = TextBlob(text[i])
        fake_pol.append(blob.sentiment.polarity)
        fake_sub.append(blob.sentiment.subjectivity)
```

Word Embedding : TF–IDF

Assign tf-idf weight for each term t in a document d,

increases with number of occurrences of term in a doc

with rarity of term across entire corpus.

TF = Term Frequency — Frequency of a word in a certain piece of news

IDF = Inverse Document Frequency — \`Frequency\` of a word in the whole corpus

$$idf(t) = \log \frac{n_d}{df(d,t)} + 1$$

$$idf(t) = \log \frac{1 + n_d}{1 + df(d,t)} + 1$$ Smoothed

$$TF - IDF = tf(t) \times idf(t)$$

Package :

sklearn.feature_extraction.text.TfidfTransformer

```
data=pd.read_csv('../dataset/politi_data.csv')

data=data.fillna(' ')
data['total']=data['title']+' '+data['author']+data['text']

transformer = TfidfTransformer(smooth_idf=False)
count_vectorizer = CountVectorizer(ngram_range=(1,2))
counts = count_vectorizer.fit_transform(data['total'].values)
tfidf = transformer.fit_transform(counts)
```

Methods Overview

- sklearn
- Logistic Regression

  Regularization is applied by default

  C – regularization strength, a smaller value suggests a stronger regularization
- SVM **SVM** & **Nu–SVM**

  Nu–SVM uses a parameter to control the number of support vectors

  Adjust their kernels
- Decision Trees, Extra Trees and Random Forest

  How performance is related to depth

  Compare these three similar classifiers
- Adaboost

Experiments : Ablation study — AdaBoost

Table 1: Performance of Different Methods on Dataset Political Without Adaboost

	Train Acc	Test Acc	AUC
DecisionTreeClassifier	1.00	0.84	0.84
ExtraTreesClassifier	1.00	0.83	0.82
RandomForestClassifier	0.99	0.81	0.80

Table 2: Performance of Different Methods on Dataset Political With Adaboost

	Train Acc	Test Acc	AUC
DecisionTreeClassifier	1.00	0.83	0.83
ExtraTreesClassifier	1.00	0.91	0.90
RandomForestClassifier	1.00	0.83	0.83

Table 3: Performance of Different Methods on Dataset Gossip Without Adaboost

	Train Acc	Test Acc	AUC
DecisionTreeClassifier	1.00	0.81	0.80
ExtraTreesClassifier	1.00	0.89	0.89
RandomForestClassifier	0.97	0.77	0.76

Table 4: Performance of Different Methods on Dataset Gossip With Adaboost

	Train Acc	Test Acc	AUC
DecisionTreeClassifier	1.00	0.81	0.81
ExtraTreesClassifier	1.00	0.90	0.90
RandomForestClassifier	1.00	0.81	0.81

Experiments : Ablation study — SVM-Kernel

Table 6: Performance of SVM with Different Kernel(political)

kernel	Train Acc	Test Acc	AUC
linear	**1.00**	**0.83**	**0.82**
poly	0.58	0.52	0.50
sigmoid	0.44	0.44	0.43
rbf	0.59	0.53	0.51

Table 8: Performance of SVM with Different Kernel(gossip)

kernel	Train Acc	Test Acc	AUC
linear	**1.00**	**0.84**	**0.84**
poly	0.57	0.56	0.50
sigmoid	0.47	0.49	0.48
rbf	0.63	061	0.65

Table 5: Performance of NuSVM with Different Kernel(political)

kernel	Train Acc	Test Acc	AUC
linear	0.94	0.83	0.83
poly	0.41	0.46	0.47
sigmoid	0.58	0.58	0.59
rbf	**0.92**	**0.84**	**0.84**

Table 7: Performance of NuSVM with Different Kernel(gossip)

kernel	Train Acc	Test Acc	AUC
linear	**0.94**	**0.85**	**0.85**
poly	0.69	0.64	0.68
sigmoid	0.58	0.55	0.57
rbf	0.92	0.84	0.84

Experiments : Fine-tuning with Grid Search

Table 11: Performance of Different Methods with Dataset Political Without Fine Tuning

	Train Acc	Test Acc	AUC
DecisionTreeClassifier	1.00	0.81	0.79
ExtraTreesClassifier	1.00	0.78	0.77
AdaBoostClassifier	0.92	0.81	0.80
RandomForestClassifier	0.98	0.74	0.73
MultinomialNB	0.96	0.77	0.75
LogisticRegression	**1.00**	**0.84**	**0.83**
LinearSVM	1.00	0.83	0.82
Nu SVM	0.92	0.80	0.79

Table 12: Performance of Different Methods with Dataset Gossip Without Fine Tuning

	Train Acc	Test Acc	AUC
DecisionTreeClassifier	1.00	0.81	0.80
ExtraTreesClassifier	1.00	0.76	0.75
AdaBoostClassifier	0.88	0.83	0.83
RandomForestClassifier	0.97	0.77	0.76
MultinomialNB	0.96	0.75	0.74
LogisticRegression	**1.00**	**0.86**	**0.85**
LinearSVM	1.00	0.80	0.79
Nu SVM	0.92	0.82	0.80

Table 13: Performance of Different Methods with Dataset Political with Fine Tuning

	Train Acc	Test Acc	AUC
DecisionTreeClassifier	1.00	0.83	0.81
ExtraTreesClassifier	1.00	0.79	0.78
AdaBoostClassifier	0.91	0.81	0.80
RandomForestClassifier	0.98	0.75	0.74
MultinomialNB	0.96	0.77	0.75
LogisticRegression	**1.00**	**0.88**	**0.86**
LinearSVM	1.00	0.83	0.82
Nu SVM	0.92	0.84	0.84

Table 14: Performance of Different Methods with Dataset Gossip With Fine Tuning

	Train Acc	Test Acc	AUC
DecisionTreeClassifier	1.00	0.82	0.80
ExtraTreesClassifier	1.00	0.78	0.76
AdaBoostClassifier	0.87	0.84	0.85
RandomForestClassifier	0.97	0.82	0.78
MultinomialNB	0.96	0.75	0.74
LogisticRegression	**1.00**	**0.88**	**0.85**
LinearSVM	1.00	0.84	0.84
Nu SVM	0.94	0.85	0.85

Conclusion

- TF–IDF is a good way to embed text content
- Logistic Regression is the most suitable method for this fake news detection task
- Adaboost trick can improve the performance of some classifiers

  With adaboost, ExtraTrees performs best at an AUC of about 0.90 on political and gossip dataset

Q&A

Scan to checkout details !