Detecting Fake News Through Natural Language Processing

Group: JSMCJ

York University

January 18, 2019

Outline

Introduction

- Fake news, defined by the New York Times as

  ”a made-up story with an intention to deceive”,
- Fake news is arguably one of the most serious challenges facing the news industry today.
- We want to explore how we can use maching learning techniques and NLP to to combat the fake news problem.
- **Goal:** Determine if a story is real or not.
- To reduce or stop the destructive consequences of the proliferation of fake news

Existing Work:

- FNC: Fake News Challenge.
  $http : //www.fakenewschallenge.org/$
- Roy, A., Basak, K., Ekbal, A., and Bhattacharyya, P. (2018). A Deep Ensemble Framework for Fake News Detection and Classification. CoRR abs/1811.04670
- Ruchansky, N., Seo, S., and Liu, Y. (2017). CSI: A Hybrid Deep Model for Fake News Detection. CIKM.
- Yang, W. (2017). ”Liar, Liar Pants on Fire”: A New Benchmark Dataset for Fake News Detection. ACL 2017. arXiv:1705.00648

Data collection and Dataset preparation:

- We used the Kaggle fake news and megred with "All the news" in the Kaggle data sets.
- A first step in the data preparation was to merge the two data sources, of real and fake news. The aim was to obtain a balanced mix of real and fake news items, suitable for modeling purposes. The datasets were first matched by month of publication: In the real news dataset, the data falls between the years 2015 and 2017, and for our project, in order to match the time period in which the fake news articles were published (November 2016), we only used news articles published in October to December 2016.

Data collection and Dataset preparation:

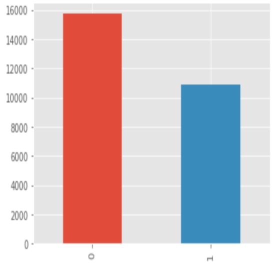
- In the real dataset, we aimed to select several reputable publications, as well as several other 'less-reputable' publications, in order to have a balanced representation of news published by a real news publication company. We used a media bias/fact check website (source: https://mediabiasfactcheck.com/) to assess each news publication company.
- We only use the features: Title, text and the target value is the column named **'fake'** in our data set, 1 for being fake and 1 for being a real news.

Data exploration:

- The 'Title' column in the fake news dataset had missing values; but ‘Title’ and ‘Thread-title’ variables were the same we replaced the missing values in ‘Title’ with the values from ‘Thread-title’.
- Only English-language news records were kept, filtered by ‘Language’ in the fake news dataset.
- Created a ‘Month’ variable extracted from the publication date from the real and fake news datasets

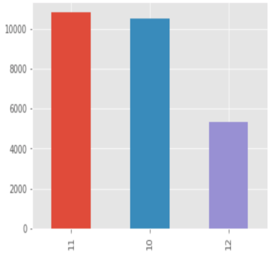
Data Visualization:

- Distribution of our Target variable, ‘fake’



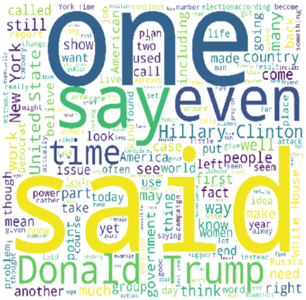
Data Visualization:

- Distribution of the ‘month’ variable



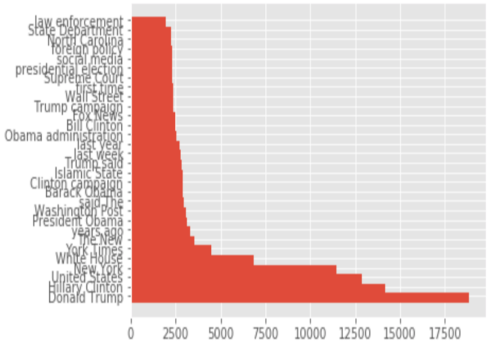
Data Visualization:

- We also performed exploration of the concatenated news articles' title and text, using Python's NLTK library.
- After tokenization and stopword removal, we created a WordCloud



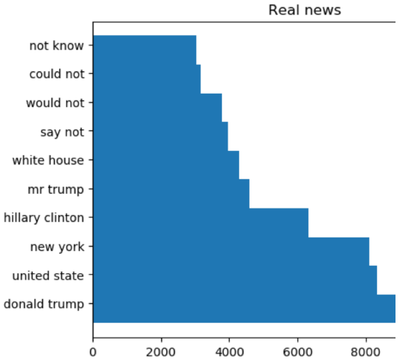
Data Visualization:

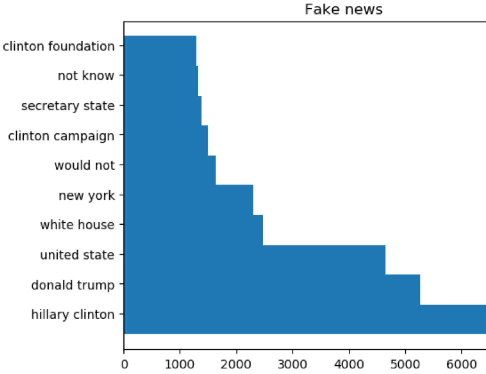
- Word pairs with the highest frequency include “Donald Trump", “Hillary Clinton", “New York", “White House", “United States", “President Obama", “Washington Post", “Trump said", “Wall Street", “Islamic State", and “last week".



Data Visualization:

Interestingly, “Donald Trump" dominates the real news and “hillary clinton" dominates the fake news.





Text Preprocessing and Normalization

- We build classification models on the news body.

Normalization Process:

Normalization is done following these steps:

1. Strip HTML tags.
2. Remove accented character
3. Contraction expansion
4. Convert all letter to lowercase
5. Remove redundant newlines
6. Insert spaces between special characters to isolate them
7. Lemmatize text
8. Remove special characters
9. Remove redundant white spaces
10. Remove stop words

Feature Engineering and Classifiers:

- Our experiments have explored 6 kinds of different features 6 kinds of classifiers.

Classifiers / Features	Logistic Regression	Multinomial Naïve-Bayes	Random Forrest	SVM Linear Kernel	Logistic Regression	CNN
Count-Vectors	✓	✓	✓	✓	✓	
TF-IDF	✓		✓	✓	✓	
Naïve Bayesian Count-Vectors	✓		✓	✓	✓	
Naïve Bayesian TF-IDF	✓		✓	✓	✓	
Word embedding-Fasttext 300d						✓
Word embedding-Glove 50d						✓

Feature Engineering:

- The vectorizers and their parameters are summarizes in the following table

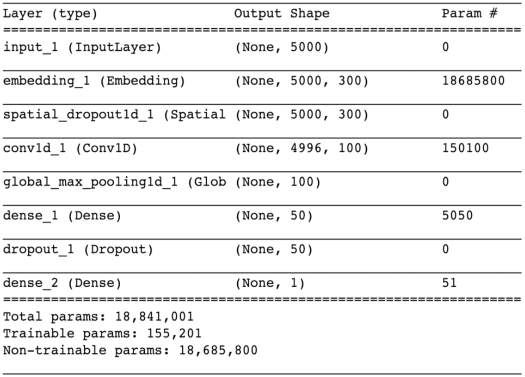
Vectorizers	**Parameters**
Count-Vectors	2-gram
TF-IDF	2-gram
Naïve Bayesian Count-Vectors	2-gram, smoothing factor=1
Naïve Bayesian TF-IDF	2-gram, smoothing factor=1
Word embedding fastText 300d	Word vector size=300 Unique word in file= 400K Max document length=5000
Word embedding GloVe 50d	Word vector size=50 Unique word in file= 1M Max document length=5000

Classifiers:

- Our projects have used models from sklearn and Keras packages.

Classifiers	Parameters
Logistic Regression	C=4
Multinomial NB	sklearn Default
Random Forrest	50 estimators max-features=0.8
Bagging of Logistic Regression(C=1)	50 estimators Bootstrapped features Boostrapped samples
SVM	Kernel = “linear"
CNN	Spatial dropout likelihood=0.3 Final dropout layer likelihood=0.2

CNN Network Structure:



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 5000)              0
_________________________________________________________________
embedding_1 (Embedding)      (None, 5000, 300)         18685800
_________________________________________________________________
spatial_dropout1d_1 (Spatial (None, 5000, 300)         0
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 4996, 100)         150100
_________________________________________________________________
global_max_pooling1d_1 (Glob (None, 100)               0
_________________________________________________________________
dense_1 (Dense)              (None, 50)                5050
_________________________________________________________________
dropout_1 (Dropout)          (None, 50)                0
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 51
=================================================================
Total params: 18,841,001
Trainable params: 155,201
Non-trainable params: 18,685,800
_________________________________________________________________
```

Classifiers:

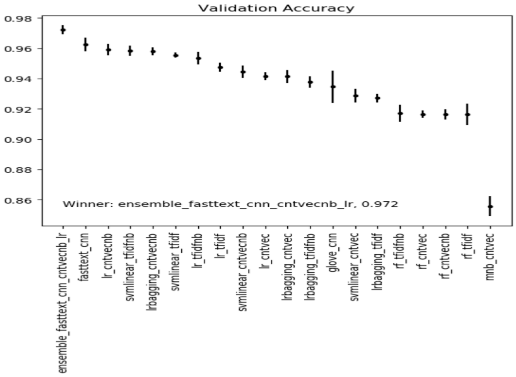
- In the final stage of our projects, we ensemble the following two models by averaging their probability outputs:
  - fastText + CNN
  - Naïve Bayesian Count-Vectors + Logistic Regression
- As documented in the modeling results, the ensemble methods have offered the best cross-validation accuracies.

Cross-validation:

- Following data preparation, the news dataset was split into train and validation sets.
  - Training set: 80%
  - Test set: 20%
- Throughout the entire projects, we have used 5-fold cross validations to evaluate different feature-classifier combinations, including the CNN classifiers.
- The mean accuracies from the 5-fold cross validations are used in order to determine which models are the best for the dataset.

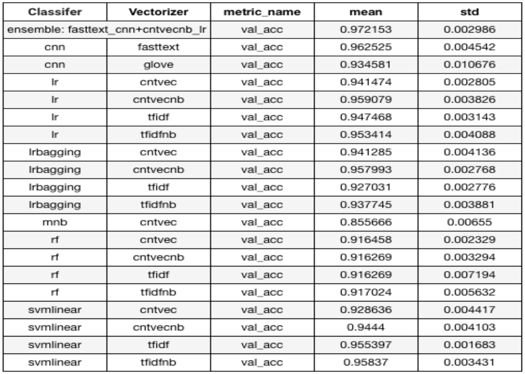
Results:

- The mean and standard deviation of cross validation accuracies of all feature-classifier pair is plotted.



Results:

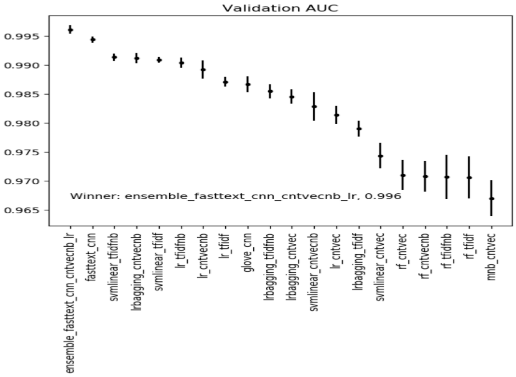
- The mean and standard deviation of cross validation accuracies of all feature-classifier pair is plotted.



Classifer	Vectorizer	metric_name	mean	std
ensemble: fasttext_cnn+cntvecnb_lr		val_acc	0.972153	0.002986
cnn	fasttext	val_acc	0.962525	0.004542
cnn	glove	val_acc	0.934581	0.010676
lr	cntvec	val_acc	0.941474	0.002805
lr	cntvecnb	val_acc	0.959079	0.003826
lr	tfidf	val_acc	0.947468	0.003143
lr	tfidfnb	val_acc	0.953414	0.004088
lrbagging	cntvec	val_acc	0.941285	0.004136
lrbagging	cntvecnb	val_acc	0.957993	0.002768
lrbagging	tfidf	val_acc	0.927031	0.002776
lrbagging	tfidfnb	val_acc	0.937745	0.003881
mnb	cntvec	val_acc	0.855666	0.00655
rf	cntvec	val_acc	0.916458	0.002329
rf	cntvecnb	val_acc	0.916269	0.003294
rf	tfidf	val_acc	0.916269	0.007194
rf	tfidfnb	val_acc	0.917024	0.005632
svmlinear	cntvec	val_acc	0.928636	0.004417
svmlinear	cntvecnb	val_acc	0.9444	0.004103
svmlinear	tfidf	val_acc	0.955397	0.001683
svmlinear	tfidfnb	val_acc	0.95837	0.003431

Results:

- The mean and standard deviation of cross validation AUC score is plotted.

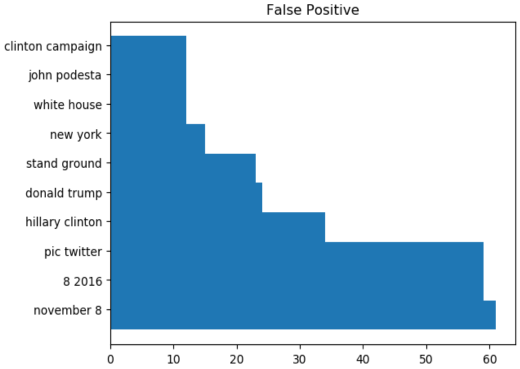


The Winner:

- Based on these results, we have a winner. That is the ensemble of the following two models by averaging their probability outputs:
  - fastText + CNN
  - Naïve Bayesian Count-Vectors + Logistic Regression
- It is easy to see that we ensemble the 2nd best and 3rd best feature-classifier pair and produce an ensemble model that is better than both of them.
- The **accuracy and AUC score of the best model** on the test set is 97.3% and 0.996% respectively.

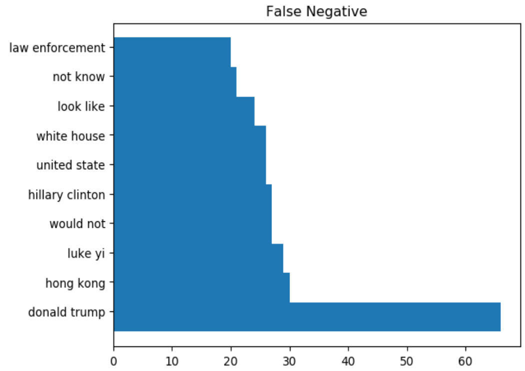
The Winner:

- Top 10 word-pair distributions on the test set divided by the following predicted labels using the best model(that is, the ensemble of fastText-CNN + cntvecnb-LR)



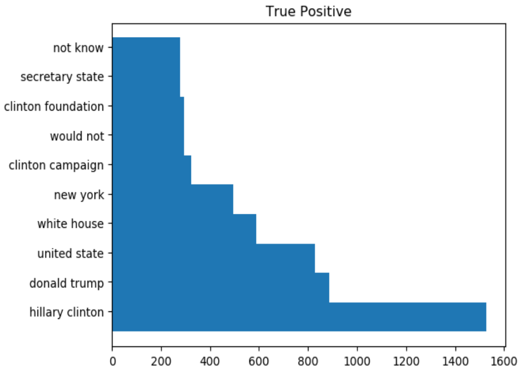
The Winner:

- Top 10 word-pair distributions on the test set divided by the following predicted labels using the best model(that is, the ensemble of fastText-CNN + cntvecnb-LR)



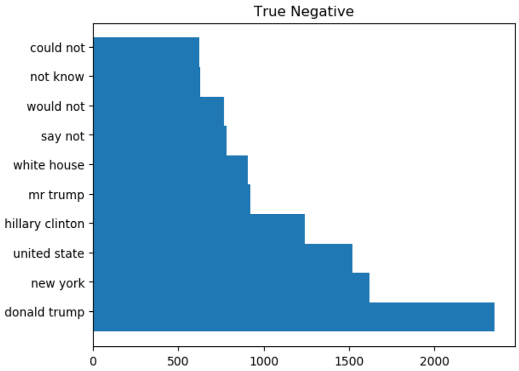
The Winner:

- Top 10 word-pair distributions on the test set divided by the following predicted labels using the best model(that is, the ensemble of fastText-CNN + cntvecnb-LR)



The Winner:

- Top 10 word-pair distributions on the test set divided by the following predicted labels using the best model(that is, the ensemble of fastText-CNN + cntvecnb-LR)



Discussion:

- The ensemble is a simple combination of the top 2 and top 3 performing models. By averaging their prediction probabilities, we gain about 1% cross validation accuracy as opposed to fastText + CNN, which is the top 2 performer.
- This confirm the understanding that, in case of ensemble methods, combining vastly different models can lead to better generalization on data that is never seen.
- The Naïve Bayesian transformation on count-vectors also proves to be very effective on enhancing predictive power. We observe about 1.8% enhancement in cross-validation accuracy when simple logistic regression is applied and Naïve Bayesian transformation is applied to count-vectors.

Discussion:

- As a matter of fact, the cross validation accuracy of Naïve Bayesian Count-Vectors + Logistic Regression, which is 95.9%, is close to fastText + CNN, which is 96.3%. Yet, the former is order of magnitude faster to train than the latter.
- our GloVe + CNN model is not performing close to fastText + CNN. One possible explanation is that we have only used a 50-dimension GloVe pre-trained vectors while for fastText we have used 300-dimension word vectors.
- all models except for one (Multinomial NB + Count Vector) yield >90

Discussion:

- Combined with the knowledge of the accuracies people reported on Fake News Challenge (FNC-1), where >90% accuracies are also reported, we think that the fake news detection task might not be a difficult task for machine learning algorithm. And that might be the reason we have such high cross validation accuracy (>97%) even with light parameter tuning on the models.

Conclusion and Future Work:

- We have undergone the machine learning life cycle for a NLP classification problem-fake news detection. Various feature extraction methods and machine learning algorithms are explored.
- The best feature-classifier pair is selected using cross-validation accuracies.
- The final accuracy on the test set is shown to be 97.3%.
- We think that this particular text classification may be an easy problem for machine learning algorithm.
- In the future, the models should be tested on various source of labeled news in order to investigate the hypothesis of easiness of the fake news detection problem. Further more, features other than the news body(such as authors, titles) can be further explored to improve predictive power of the models.

Thank You for Your Attention!