DETECTING FAKE NEWS THROUGH NATURAL LANGUAGE PROCESSING

James Lai, Julia Mitroi, Shahla Molahajloo, and Manvir Sekhon
York University

January 17, 2019

Abstract

Fake news detection has become a key societal issue and an emerging research area that has attracted extensive attention. News content can be analyzed through big data and mathematical models that process human language and classify text, such as Natural Language Processing (NLP). Because the issue of fake news detection is both challenging and relevant, we conducted a project to construct machine learning (NLP) models that can flag news content as fake, and to further facilitate research on the problem. Building on previous binary classification work by Bajaj [1], we used a joint dataset of fake and real news and explored various feature extraction methods and algorithms for classifying news content into 'fake' and 'real' categories. We evaluated a total of twenty feature-classifier combinations, and selected the best feature-classifier pair using cross-validation accuracies, with the final accuracy on the test set being 97.3%.

1. Introduction

Recent major political decisions like Brexit and the last U.S. presidential elections surprised journalists and professional predictors alike with their unexpected outcomes. In the aftermath, one major contributor has been identified as being ″fake news": stories and memes with no basis in truth that may have been created to sway the outcome of these political events. Over the past two years, fake news has become a key societal issue and a technical challenge for media organizations to deal with. This type of content can be difficult to identify because it often hides under the appearance of a legitimate news organization; and because the term "fake news" covers intentionally false, deceptive stories, as well as factual errors and satire, and sometimes stories that a person just does not like (but that may be factually true). Addressing the problem of fake news requires clear definitions and examples, and validated bias-free methods.

Fake news is harmful because it affects the functional logic and integrity of the society. In today's world, where people are informed mainly through the media and form their political opinions through it, this process is threatened when misinformation spreads through the media. When it is no longer clear, what is false and what is correct, people lose their confidence in official sources. "Is it true, what they tell me on TV? What did I read in the newspaper? What I've heard on the net?" In addition to creating confusion and misunderstanding about important social and political issues, there may be biased and misleading news stories related to medical treatments and life-threatening diseases. Trusting these false stories could lead people to make decisions that may be harmful to their health. Research has shown that news consumers have difficulty distinguishing between real and fake news sources and information.

Fake news is clearly a serious problem, and many wondered what data scientists can do to detect it and stymie its viral spread: How can Artificial Intelligence (AI) and deep learning be leveraged to detect fake news? Efforts in this direction have shown that news content can be analyzed through big data and mathematical models that process human language and classify text. Natural Language Processing (NLP) is the field of AI devoted to processing and analyzing any form of natural human language. It aims to bridge the gap between how computers and humans process information, and by using it data scientists and machine learning engineers can analyze large quantities of human communication data. In the context of the fake news problem, NLP enables breaking down news articles into their components and choosing important features, to then construct and train models to identify unreliable information.

The current project addresses the issue of fake news by analyzing news text with NLP. The business objective is to construct a model (or several) that can flag news content as fake, and thus help reduce or stop the destructive consequences of the proliferation of fake news.

2. Exisiting Work

Text classification is a foundational task in many NLP applications, including fake news detection. Most of the existing studies on fake news detection are based on the classical supervised model. In recent times, there has been an interest towards developing deep learning-based fake news detection systems. For example in [6] two social media datasets (Twitter and Weibo) used and developed a hybrid deep learning model which showed the accuracies of 89% on Twitter data and 95% on Weibo data. They showed that both capturing the temporal behavior of the articles as well as learning source characteristics about the behavior of the users, are important for fake news detection in the social media. Further integrating these two elements improves the performance of the classifier. Bajaj [1] used a binary classification framework (i.e., fake or true) and applied various deep learning strategies on a dataset composed of fake news articles from Kaggle and real news articles from a data source called Signal Media News, and observed that classifiers based on Gated Recurrent Unit (GRU), Long Short Term Memory (LSTM), Bi-directional Long Short Term Memory (Bi-LSTM) performed better than the classifiers based on CNN, with precision of 0.89, 0.93, and 0.88 respectively.

Part of the existing research on fake news detection has been done through the Fake News Challenge [4](FNC-1), a public competition that aims to find automatic methods for detecting fake news. Projects within FNC-1 use a non-binary classification approach. The dataset for the challenge consists of 49,972 headline-body pairs, with the objective being to classify the pairs as unrelated, agreeing, disagreeing, or discussing. The primary task of FNC-1 is stance detection between headline-article pairs. Stance detection is the act of identifying the relationship between two pieces of text in FNC-1, as noted above, the possible relationships are “agree", “disagree", “discuss", or “unrelated". Several model architectures for fake news detection have been used within the FNC-1, ranging from dense neural networks (DNN), to recurrent neural networks (RNN), to pre-trained word embeddings, and have achieved high model accuracy. For example, in [5] a model with lexical and similarity features passed through a multi-layer perceptron (MLP) with one hidden layer and achieved 88% overall accuracy; and in [3] the best model was a bag-of-words followed by a three-layer multi-layer perceptron (BoW MLP), their model achieving categorical test-set accuracy of 93%, and on the competition-specific FNC-1 metric it achieved a test-set score of 89%.

In our current project for fake news detection, we followed up on previous binary classification work done for example by Bajaj, noted above, using the same Kaggle fake news dataset that Bajaj used, but merged with a different source of real news (described in the next section). We attempted a number of supervised machine learning models, as well as deep learning algorithms, to assess whether our combination of real and fake news sources can produce results comparable to previous classification work done for fake news detection.

3. Methodology

3.1. Approach

The news text analysis goal noted above is to be achieved in this project by using

a) two publicly available news datasets,
b) a machine learning pipeline for text classification task.

The broad steps of the machine learning project are:

- Identifying the data source(s) (described below, under Data Collection),
- Data preparation: pre-processing, cleaning,
- Selecting the relevant information from the data and convert it into formats recognizable for machine learning algorithms,
- Identifying the best performing text classification models using 5-fold cross validation,
- Interpreting and reporting the results .

3.2. Data Understanding

3.2.1 Data Collection

The data used for this project was sourced from two public datasets from the Kaggle Repository, a fake and a real news dataset, which we merged to obtain a balanced mix of real and fake news items. The news articles in both datasets were published from October to December of 2016.

The fake news contains text and metadata from 244 websites and represents 12,999 posts in total posted across 30 consecutive days, tagged by the 'BS Detector' Chrome Extension by Daniel Sieradski (https://github.com/bs-detector/bs-detector). Each website is classified according to the BS Detector. Our group collected (accessed and downloaded) the fake news data from the Kaggle platform, at https://www.kaggle.com/mrisdal/fake-news.

The real news was sourced from a Kaggle dataset named "All the news": https://www.kaggle.com/snapcrack/all-the-news/home. The publications include the New York Times, Breitbart, CNN, Business Insider, the Atlantic, Fox News, Talking Points Memo, Buzzfeed News, National Review, New York Post, the Guardian, NPR, Reuters, Vox, and the Washington Post. The data falls between the years 2015 and 2017.

3.2.2 Dataset Description

This dataset **Fake news** classifies news items described by a set of attributes such as "Bias", "BS", "Conspiracy", "Fake", "Satire". It has 12,999 observations and 20 attributes (columns),11 continuous and 8 categorical; and the classification variable, "Type". Table 1 shows the list of variables, and their data types and descriptions.

This dataset **Real News** contains news items described by a set of attributes such as publication date, publisher, URL, author. It has 49,920 observations and 9 attributes (columns), 4 continuous and 6 categorical. Table 2 below shows the list of variables, and their data types and descriptions.

3.2.3 Dataset Preparation (for combined real and fake news)

The Jupyter Notebook called News-DataPrep-EDA.ipynb on GitHub in the 'codes' folder documents the steps for combining real and fake news. 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),

Table 1: *Fake News*

	Attribute	Variable Name	Variable Description
1	numerical	uuid	unique identifier
2	numerical	ord-in-thread	order in thread
3	qualitative	author	author of story
4	date	published	date published
5	qualitative	title	title of the story
6	qualitative	text	text of story
7	qualitative	language	data from webhose.io
8	date	crawled	date the story was archived
9	qualitative	site-url	site URL from BS detector
10	qualitative	country	data from webhose.io
11	qualitative	domain-rank	data from webhose.io
12	qualitative	thread-title	thread title
13	numerical	spam-score	data from webhose.io
14	qualitative	main-img-url	image from story
15	numerical	replies-count	number of replies
16	numerical	participants-count	number of participants
17	numerical	likes	number of Facebook likes
18	numerical	comments	number of Facebook comments

Table 2: *Real News*

	Attribute	Variable Name	Variable Description
1	numerical	id	unique identifier
2	qualitative	title	article title
3	qualitative	publication	publication name
4	qualitative	author	author name
5	qualitative	date	date of publication
6	qualitative	year	year of publication
7	date	month	month of publication
8	qualitative	url	URL for the article
9	qualitative	content	article content

we only used news articles published from October to December 2016.

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. Those included are listed below. We used a media bias/fact check website (source: https://mediabiasfactcheck.com/) to assess each news publication company. The results are below (each company is rated on their factual reporting). Factual reporting:

- Very High: Reuters, NPR
- High: new york times, Atlantic, guardian, Washington post, vox
- Mixed: fox news, CNN, national review

From the fake news dataset, we selected English articles only, and also only those articles labeled "BS" by the Chrome BS Detector. We created a Target variable called 'fake' that has a value of 0 for real news articles and 1 for fake news items. The merged real and fake news dataset had 9 columns, 8 of which were common to both data sources, and the Target variable ('fake'), and 26612 rows (news articles).

3.3. Exploratory Data Analysis

3.3.1 Data Exploration

The Jupyter Notebook called News-DataPrep-EDA.ipynb on GitHub in the 'codes' folder includes the details of our Exploratory Data Analysis (EDA). In any NLP project, a first activity is to prepare the data for model building, which involves loading the dataset into Python, exploring the data, and performing basic preprocessing and cleaning. In the Jupyter notebook News-DataPrep-EDA.ipynb, Summary statistics were computed, and patterns and distributions were charted. The 'Title' column in the fake news dataset had missing values; however, we noticed that the 'Title' and 'Thread-title' variables were the same. We, therefore, replaced the missing values in 'Title' with the values from 'Thread-title'. As noted above,

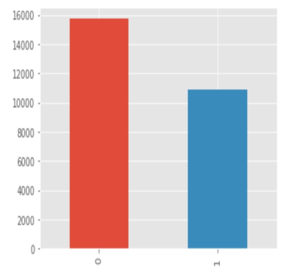


Figure 1: *Distribution of Target Variable, "fake"*

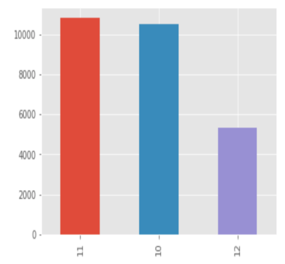


Figure 2: *Distribution of "month" Variable*

only English-language news records were kept, filtered by 'Language' in the fake news dataset. We also created a 'Month' variable extracted from the publication date from the real and fake news datasets.

Figure 1 shows the distribution of our Target variable, 'fake'.

There are 15725 real news articles and 10900 fake news articles in the data. The Target variable was used for building supervised machine learning classification models.

Figure 2 shows the distribution of the 'month' variable. The counts for articles published in the months of October, November, and December 2016 were 10483, 10816, and 5313 respectively. 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 (or Tag cloud), a data visualization technique used for representing text data in which the size of each word indicates its fre-

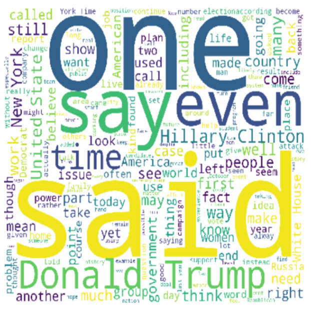


Figure 3: *WordCloud of News Data*

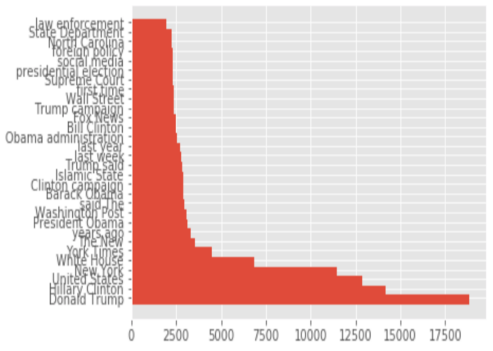


Figure 4: *Highest Frequency Bigrams*

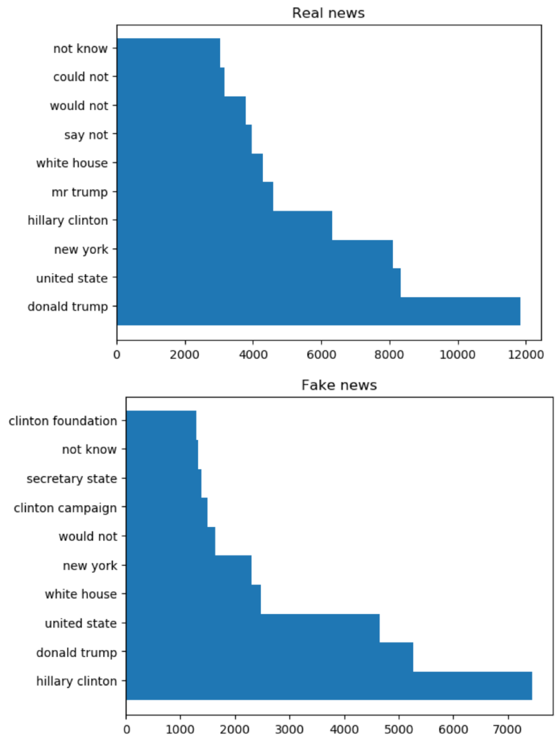


Figure 5: *Word-Pair Distribution on the Real News and Fake News*

quency or importance. The WordCloud on our news data(see Figure 3) reveals that words or pairs of words such as “one", “said", “Donald Trump", “time", “Hillary Clinton", and “New York" have the highest frequency in the text data. Further, we did a bigram (word pairs) analysis using NLTK to generate all possible bigrams and select the ones with the highest frequency. Figure 4 shows the most frequent bigrams in the news text data; 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". As an interesting exercise, we also plotted the word-pair distribution on the real news and fake news respectively. Interestingly, “Donald Trump" dominates the real news and “Hillary Clinton" dominates the fake news(see Figure 5).

Lastly, we plot the distribution of sentence lengths, Figure 6. The length of a sentence is defined as the number of tokenized words in it, after the sentence being normalized. Based on the plot, we see the majority of sentence length is less than 2000. With this information, we have subsequently clamp the sentence length to be 5000 when working with convolutional neural networks, as discussed later in this write-up.

3.3.2 Text Preprocessing and Normalization

One way to process the text from our news dataset to build models to flag fake news would be to analyze the text from the headlines (article title) only. However, based on previous work done in the area of fake news detection (e.g., in this project: https://medium.com/@Genyunus/detecting-fake-news-with-nlp-c893ec31dee8), the text from the headline only is not sufficient for

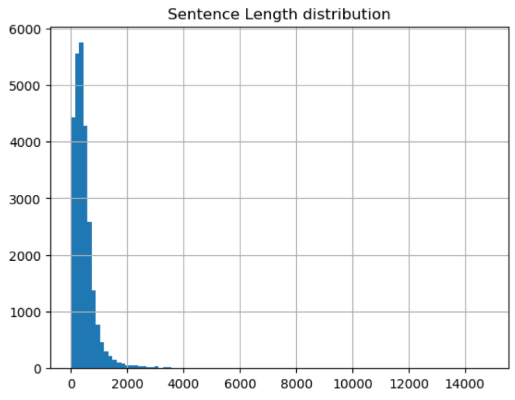


Figure 6: *Distribution of Sentence Lengths*

accurate classifications and predictions. That leads to the following 3 alternatives:

1. Build classification models on the news body.
2. Build classification models on the concatenated texts of news body and news title. However, we lose information regarding what constitutes a title and what not.
3. Build ensemble models based on classifiers trained on news body and news title, and possibly other features such as authors, dates, etc.

For this project, only the 1st option is implemented and evaluated with machine learning algorithms. 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

The normalisation steps, and code sections, were taken from the in-class materials with minimum modifications. The steps are documented in “codes/normalization.py".

3.4. Feature Engineering and Classifiers

Our experiments have explored 6 kinds of different features 6 kinds of classifiers. However, not all possible feature-classifier combinations are evaluated. Table 3, Feature-Classifier Evaluation Matrix summarized the feature-classifier pair that has been evaluated during this project.

3.4.1 Feature Engineering

In this project, all the feature extractors are wrapped in “codes/ Vectorizer.py" so that they all have the same software interfaces. One special note related to feature engineering is that we have taken care not to fit our vectorizer on the whole dataset. Instead, we have fitted all the vectorizer based only on training set. In 5-fold cross validation, this means the vectorizer is fitted on the 4 folds of the training data instead of all the 5 folds of the training data. For count-vectors and TF-IDF vectorizer, 2-gram is used. For word embedding, two pre-trained word vectors fastText(300 dimensions) and GloVe(50 dimensions) are downloaded and evaluated with convolutional neural networks. Since we have varying document length for each piece of news in the dataset, each piece of news is clamped to be 5000 words of length. If the news has less than 5000 words, 0s are padded to the news so that the total length is exactly 5000. If greater than 5000, only the first 5000 words are preserved.

One special feature transformation that needs more elaboration is the Naïve Bayesian Count-Vectors and Naïve Bayesian TF-IDF. The ideas are adopted from [7] and the codes are referenced from this online jupyter notebook

Table 3: *Feature-Classifier Evaluation Matrix*

Features \ Classifiers	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						✓

Table 4: *Vectorizer Parameters*

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

[2]. Let $f^{(i)} \in R^{|V|}$ be the feature count vector for training case i with label $y(i) \in \{1, 0\}$ where V is the set of features. Define the count vectors as

$$p = \sum_{\{i:y(i)=1\}} f^{(i)}$$

and

$$q = \sum_{\{i:y(i)=0\}} f^{(i)},$$

where α is the smoothing parameter. Within the training label vector $y(i)$, let the number of labels whose values equal to 1 to be L_p, and the number of labels whose values equal to 0 to be L_q. Then log-count ratio is

$$r = log \frac{(p + \alpha)/(L_p + \alpha)}{(q + \alpha)/(L_q + \alpha)}.$$

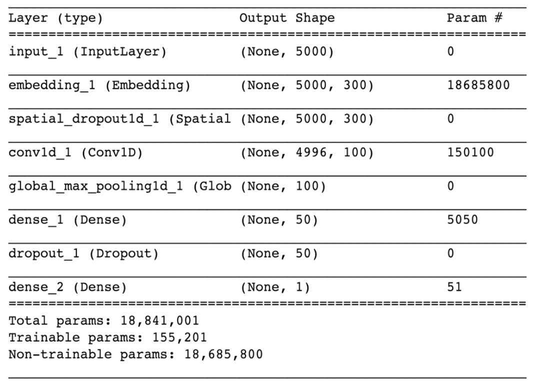
The resulting is a vector whose length equal to $|V|$. This r is then multiplied with the training vectors and also the test vectors. The resulting vectors are then used with different classifiers.

The vectorizers and their parameters are summarizes in Table 4.

3.4.2 Classifiers

Our projects have used models from sklearn and Keras packages. All the classifiers are wrapped in "codes/ClassifierWrapper.py" so that they have the same software interfaces. The parameters for the models evaluated are summarized in Table 5. The structures of the convolutional neural networks are shown in Figure 7 . In the final stage of our projects, we ensemble the following two models by averaging their probability outputs:

- fastText + CNN



```
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
_________________________________________________________________
```

Figure 7: *CNN Network Structure*

- Naïve Bayesian Count-Vectors + Logistic Regression

As documented in the modeling results, the ensemble methods have offered the best cross-validation accuracies.

3.5. Modeling

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 on the training set 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. One special note to document here is that, shuffling should be turned when generating training/test sets and also the K-fold cross validation sets. In our experiments we observe up to 3% difference in cross validation accuracy with and without shuffling turned on. After picking the best feature-classifier pair based on the cross-validation results, we will retrain the feature-classifier on the whole training set. Then the final evaluation on the test sets are reported.

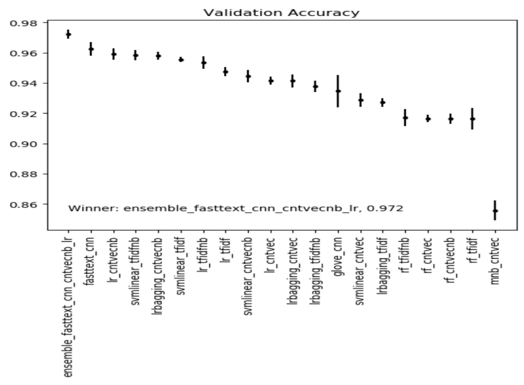


Figure 8: *Cross-Validation Accuracy*

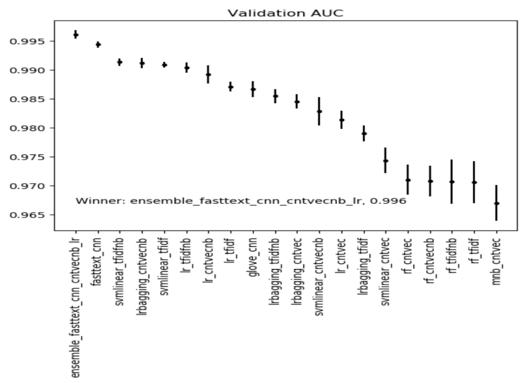


Figure 9: *Cross-Validation AUC*

4. Result

We have documented the analysis on the modeling results in "codes/ModelingResults.ipynb". The mean and standard deviation of cross validation accuracies of all feature-classifier pair is plotted in Figure 8 and Figure 10. Likewise, the mean and standard deviation of cross validation AUC score is plotted in Figure 9.

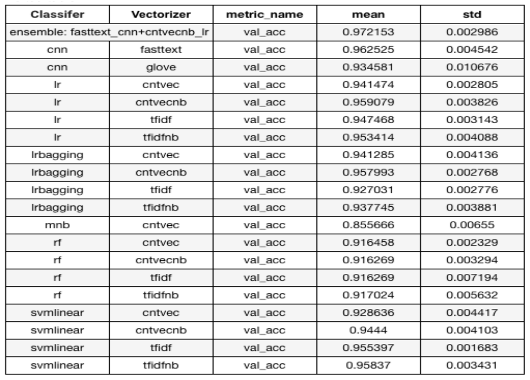
Based on these results, we have a winner. That is the ensemble of the following two models by averaging their probability outputs:

- fastText + CNN
- Naïve Bayesian Count-Vectors + Logistic Regression

It is easy to see that we ensemble the 2nd best

Table 5: *Classifiers' Parameters*

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



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

Figure 10: *Cross-Validation Accuracy of Models*

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.

5. Discussions

In this project, we have evaluated a total of 20 feature-classifier combination on a dataset that we create by combining several sources of news. The best model is ensemble of the following two models by averaging their probability outputs:

- fastText + CNN
- Naïve Bayesian Count-Vectors + Logistic Regression

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. 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.

Another realization after modeling activities is that, 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. Nevertheless, in all of our modeling results, all models except for one (Multinomial NB + Count Vector) yield $>$ 90% cross validation accuracy. 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 binary classification 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.

6. 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. And the best feature-classifier pair is selected using cross-validation accuracies. The final accuracy on the test set is shown to be 97.3%.

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. Furthermore, features other than the news body (such as authors, titles) can be further explored to improve predictive power of the models.

References

[1] Bajaj, S. (2017). “The Pope Has a New Baby! Fake News Detection Using Deep Learning." CS 224N - Winter 2017.

[2] Howard, J. (2018). NB-SVM strong linear baseline: Python notebook using data from Toxic Comment Classification Challenge. Retrieved from https://www.kaggle.com/jhoward/nb-svm-strong-linear-baseline

[3] Davis, R., and Proctor, C. (2017). Fake News, Real Consequences: Recruiting Neural Networks for the Fight against Fake News. https://web.stanford.edu/class /cs224n/reports/2761239.pdf.

[4] FNC. (2017). Fake News Challenge. http://www.fakenewschallenge.org/

[5] Riedel, S., Yao, L., McCallum, A., and Marlin, B.M. (2013). Relation extraction with matrix factorization and universal schemas. In Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 74-84, Atlanta, Georgia, June. Association for Computational Linguistics.

[6] Ruchansky, N., Seo, S., and Liu, Y. (2017). CSI: A Hybrid Deep Model for Fake News Detection. CIKM.

[7] Wang, S., and Manning, C. (2012). Baselines and Bigrams: Simple, Good Sentiment and Topic Classification. In “Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)".