Real, Fake and Sarcastic News Detection in a Class Imbalance Setting

Seungjun (Josh) Kim

sekim@vassar.edu

1 Introduction

Dissemination of fake news has obnoxious societal effects. Previous literature warned that it weakens public trust in governments, distorts the way people respond to legitimate news and may cause tangible harm to the economy as illustrated by the example of the fake news that former President Barack Obama was injured in an explosion wiping off $130 billion in stock value. [2] Even now, fake news on COVID-19 is being generated and the ability of people to differentiate fake news from real ones has not improved significantly. Fake news, on the other hand, is becoming more "neural" – The style of fake news is becoming increasingly similar to that of real news. Using various supervised algorithms, I aim to achieve similar or better classification performance using just the titles / headlines.

2 Previous literature and how my final project differs from them

Previous literature and research mainly focused on generating fake new or building models that detect fake news using various text data including body text of articles, headlines, tweets and user comments. Liu, Shu, Zafarani, and Zhou created the FakeNewNet which is the repository of news, social context, special and temporal information for studying fake news. Natali proposed a hybrid deep learning framework to model news text, user response, and post source simultaneously for fake news detection.[4] Guo utilized a hierarchical neural network to detect fake news, modeling user engagements with social attention that selects important user comments.[5]

One of the leaders of fake news detection research in terms of performance is the Grover framework model for controllable fake news text generation. For instance, if a headline such as 'Link Found Between Vaccines and Autism' is given, Grover can generate the rest of the article. Humans find these generations to be more trustworthy than human-written disinformation. Models featured in the Grover paper can classify neural fake news from real news with 73% accuracy and using Grover itself against Grover generated fake news data yields a score of 92% accuracy.

Shu, Cui, Wang, Lee, and Liu dealt with the topic of "explanability" of fake news detection models and the performances of the models were almost on-par with those of the Grover research with an accuracy score of 90.4% for the PolitiFact data. [1] But, the feature representations of deep learning models in this paper were still difficult to interpret. Furthermore, these major papers focused on classifying news data into two classes – real and fake news.

My final project differs from previous literature in three ways. Not much research focused on differentiating sarcastic news from fake news and non-sarcastic real news from sarcastic real news. Hence, I will expand previous research to a multiclass setting with three classes – non-sarcastic real, fake and sarcastic news. Furthermore, I will only be using titles / headlines of news articles and achieve similar performance as previous literature, thereby presenting ways to still create effective classification models

when researchers have limited access to text data from the body of articles.

3 Data

I collected 5 datasets from mainly two different sources. One source is the FakeNewsNet aforementioned in section 2. From this source, I got the fake and real news data for PolitiFact and GossipCop, which are political fact checking websites. The second source was Kaggle, a data science competition platform. From this source, I got news articles data from 15 different websites, BuzzFeed's real and fake news data, and the Sarcastic News Headlines Data extracted from TheOnion and HuffPost. I merged them and the merged dataset had 53851 observations.

4 Methods, Results and Discussion

4.1 Preprocessing and Meta Features Creation

For preprocessing data, I stripped HTML tags, removed accented characters and remove special characters. Then, I created basic meta features that might help improve model performance. They were word count, unique word count, stop word count, average word length, median word length, and character count.

4.2 Baseline Models

I first created multiple baseline models to see how they perform for this multiclass classification task. I tried multinomial naïve bayes (NB), linear discriminant analysis (LDA), logistic regression and Quadratic Discriminant Analysis (QDA) on various feature combinations. N-gram and tf-idf features were created with the Countvectorizer and tfidfvectorizer function respectively from python's sklearn library. I tweaked the tf-idf formula such that wf = 1+ log(tf) and used this new wf I defined (by specifying the sublinear_tf parameter to be True). I defined this new wf because it seemed unlikely that k occurrences of a term in a document truly carry k times the significance of a single occurrence. The results with cross validation fold of 5 can be seen from Table 1.

I encountered two issues when trying logistic regression and quadratic discriminant analysis. First, I experienced convergence issues for logistic regression. The data seemed too sparse for convergence to happen. Second, the computational cost for QDA was too high. My jupyter notebook died when it was ran on an environment with 8GB RAM and normal CPU without any access to GPU or TPUs. It also experienced the multicollinearity issue amongst features which meant a lot of the features were highly correlated to one another. Between NB and LDA, NB, which is traditionally known for performing well for text classification, scored higher on all three multiclass classification metrics (balanced accuracy, AUC

score, and weighted F1 score). In terms of AUC score, in particular, NB applied on n-gram features combined with the meta features I created performed the best.

Table 1: Performance of N on Various Feature Sets (cv=5)

	Balanced Accuracy	AUC Score	Weighted F1 Score
NB on n-gram feats	0.34	0.54	0.54
NB on n-gram + meta feats	**0.367**	**0.579**	**0.541**
NB on tf-idf feats	0.33	0.54	0.54
NB on tf-idf + meta feats	**0.354**	**0.574**	**0.548**
LDA on n-gram feats	0.33	0.52	0.54
LDA on n-gram + meta feats	0.33	0.56	0.54
LDA on tf-idf feats	0.33	0.53	0.54
LDA on tf-idf + meta feats	0.33	0.56	0.54

4.3 Exploratory Data Analysis and Creation of New Features based on EDA

I performed exploratory data analysis (EDA) to better understand the structure of data and be able to create my own customized features from the patterns I observe from EDA which will come in handy for improving model performance later.

Based on the kernel density estimate (KDE) plots of each feature, I created bin variables for ranges where the distributions for three different news types did not match. For instance, the 45-60 range for the number of characters feature was one example where the distributions did not match. In fact, fake news had the highest density for character counts followed by real and sarcastic news. Thus, I created the "title_cc_45_60" feature which indicated whether the number of characters for a certain observation falls under this range (1: yes, 0: no).

Regarding the TF-IDF based wordclouds, real and fake news share similar words in the wordcloud while sarcastic news have very different words featured in the wordcloud. Based on this trait, I created a new feature which counts the number of times any word in that sarcastic news wordcloud appears in the title / headline text.

Figure 1/2: Distribution of Number of Characters (left); Distribution of Unique Word Count (right)

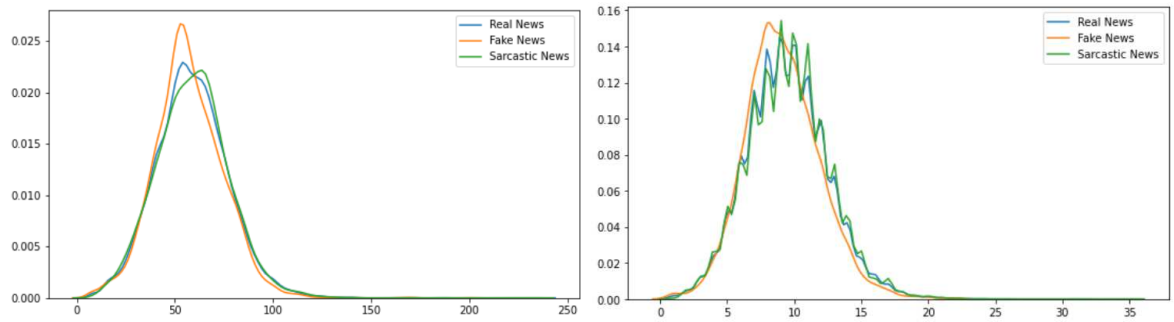


Figure 3/4: Distribution of Word Count (left); Distribution of Average Word Length

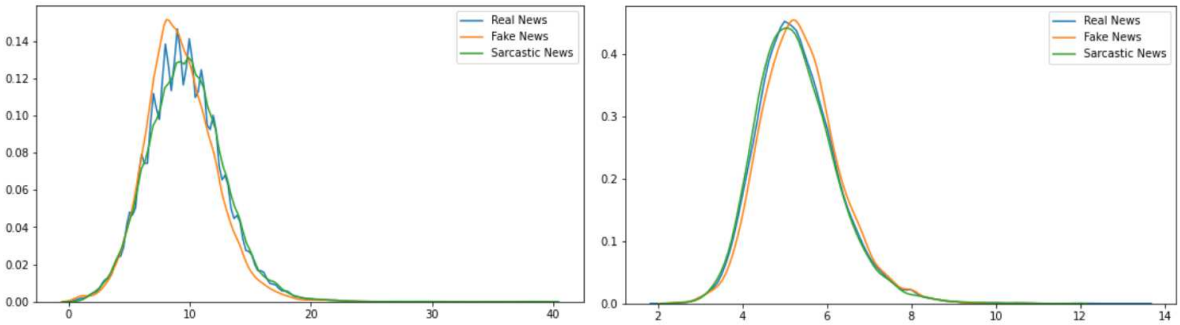


Figure 5/6: Distribution of Median Word Length (left); TF-IDF based WordCloud for real news (right)

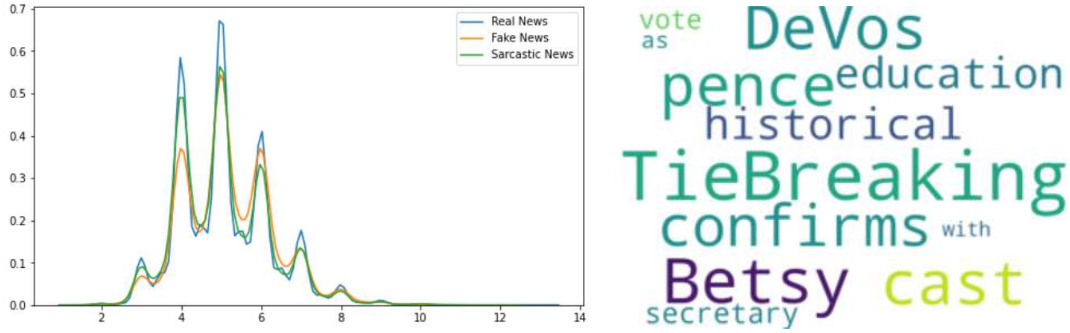


Figure 7/8: TF-IDF based WordClouds for fake (left) and sarcastic (right) news

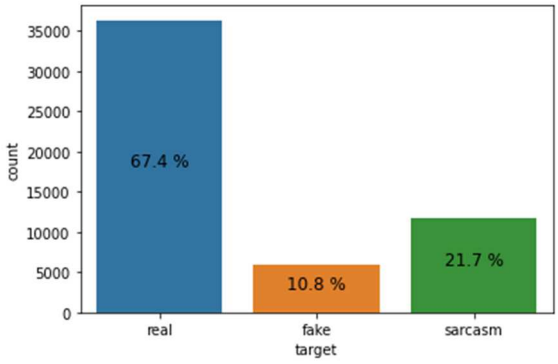
4.4 New models applied on all new features I created

Table 2: Performance of Naïve Bayes Classifiers (cv=5)

	Balanced Accuracy	**AUC Score**	**Weighted F1 Score**
NB on n-grams + meta feats + EDA feats	0.3700	**0.582**	0.541
NB on tf-idfs + meta feats + EDA feats	0.3596	0.579	**0.550**
NB on n-grams + poly(meta feats, EDA feats)	**0.3704**	0.547	0.469
NB on tf-idfs + poly(meta feats, EDA feats)	0.3701	0.546	0.469

After adding the new features I created based on patterns discovered from EDA, balanced accuracy scores, AUC scores and weighted F1 scores rose by 0.002 – 0.004. Next, I attempted polynomial transformation on all the hand-made features and, hence, added 100+ new polynomial features to the data. Training NB on that new data barely yielded increases in balanced accuracy scored of 0.0004 – 0.0105. AUC scores and weighted F1 scores actually dropped when NB was applied to this new data which included these polynomial features. Considering how AUC score is often a more reliable metric for gauging the classifier's predictive power especially for this kind of skewed data (see Figure 9 below), drop in AUC scores indicate that polynomial features probably lead to overfitting. Furthermore, it can be inferred that the multicollinearity issue which I encountered in the QDA algorithm must have gotten worse when I added these polynomial features.

Figure 9: Percent (%) of Observations in each Class (real, fake and sarcastic)



4.5 Tree and Forest Based Algorithms with Bagging and Boosting

Tree and forest based algorithms are known to perform well for categorical features. Considering that most of the features (e.g. n-gram features) are dummy variables which are categorical (either 0 or 1), it seemed likely that tree and forest based algorithms will surpass performances of Naïve Bayes classifiers.

Among such algorithms, I tried the Random Forest (RF) algorithm. It is a meta-estimator that fits

a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. [6] This "bagging" component of Random Forest often makes it a more powerful algorithm typical decision tree classifiers.

As expected, applying RF to n-gram features combined with all the features I created led to an increase of approximately 0.02 for balanced accuracy and 0.05 for AUC score and weighted F1 score in comparison to the highest scores achieved from the NB models.

Next, I used the Light Gradient Boosting Machine (LGBM) to achieve further boosts in performance. LGBM is a "boosting" based algorithm which has faster training speed, lower memory usage and capabilities of handling large-scale data than other boosting algorithms such as Extreme Gradient Boosting Machine (XGBoost). [7] Boosting is the act of iteratively learning weak classifiers with respect to a distribution and adding them to a final strong classifier. After a weak learner is added, re-weighting, which is the process of data weights being readjusted, occurs and misclassified data gain higher weights and the weight of observations that were correctly classified goes down. [8] This way, the next weak learners focus more on the observations that previous weak learners misclassified. The reason why I opted for LGBM instead of XGBoost was due to concerns on computational costs. LGBM uses Gradient-based One-Side Sampling (GOSS) to filter out the data instances for finding a split value. GOSS is better than histogram based algorithm splits which XGBoost uses in terms of speed and memory usage. LGBM outperformed RF across all three metrics. LGBM's performance scores were 0.05 – 0.07 higher than those of RF in all three metrics.

Table 3: Performance of RF and LGBM Classifiers (cv=5)

	Balanced Accuracy	**AUC Score**	**Weighted F1 Score**
RF(n_estimators = 300) on n-grams + meta feats + EDA feats	0.395	0.637	0.600
LGBM(boosting_type='gbdt', n_estimators = 5000, learning_rate=0.03, max_depth=-1) on n-grams + meta feats + EDA feats	0.464	0.706	0.656

4.6 Feature Selection and Rebalancing of skewed data

In order to retain only the most meaningful features, I decided to remove some features that do not contribute as much to the improvement of model performance. Considering that tree and forest based algorithms such as RF and LGBM do not work well with features that have low variability, I removed all features that had variance values lower than a certain threshold. I used Python scikit-learn library's VarianceThreshold(threshold = 0.01) function to get rid of all features that had variance smaller than 0.01.

As a result, 918 features were removed. Interestingly, all of those removed features were n-grams features. The hand-made features I created all had high enough variance (higher than 0.01) and thus were not removed during this feature selection process. In the end, my data had 82 n-gram features and 15 hand-made features kept.

Earlier, I alluded to the "class imbalance" problem where there are too many observations concentrated in one or few classes (figure 9). In this case, the real news observations made up 67% of the entire data while sarcastic and fake news only made up 22% and 11% of the entire data respectively. I intentionally used metrics such as "balanced" accuracy instead of just accuracy to account for this class imbalance. However, understanding the limitations of using appropriate metrics to address the class imbalance issue, I decided to use the "resampling" methods.

There are two ways of resampling data to address the class imbalance issue. One is to artificially remove data points from the majority class (a.k.a. under-sampling). The other way is to add observations to minority classes (a.k.a. over-sampling). I resorted to using a combination of both under-sampling and over-sampling.

First, I used the Tomek Links method to perform under-sampling and applied LGBM on that under-sampled data. Tomek links are pairs of very close instances, but of opposite classes. Removing the instances of the majority class of each pair increases the space between the two classes, facilitating the classification process. As a result of under-sampling, 3944 observations from the majority class (a.k.a. real news class) were removed. In comparison to the data before it was under-sampled, this under-sampled data yielded better LGBM model results; I was able to achieve an increase of 0.02 – 0.03 across all three metrics.

Table 4: Performance of LGBM Classifier on under-sampled data (cv=5)

	Balanced Accuracy	**AUC Score**	**Weighted F1 Score**
LGBM(boosting_type='gbdt', n_estimators = 5000, learning_rate=0.03, max_depth=-1) on n-grams + meta feats + EDA feats	0.496	0.733	0.665

Then, I used the SMOTE (Synthetic Minority Oversampling Technique) method for over-sampling on this already under-sampled data. SMOTE (Synthetic Minority Oversampling Technique) consists of synthesizing elements for the minority class, based on those that already exist. It works randomly picking a point from the minority class and computing the k-nearest neighbors for this point. The synthetic points are added between the chosen point and its neighbors. After this under-sampling and over-sampling process, I was able to acquire a very well balanced dataset with 32214, 32264, and 32267 data points for real, fake and sarcastic news classes.

The performance boost was significant. The LGBM model applied on this new well-balanced data recorded a balanced accuracy score of 80% which is a whopping 34% increase from the previous score of 46% achieved by the LGBM model applied on imbalanced data. It also recorded 0.933 and 0.793 for AUC score and weighted F1 score. The previous best scores on imbalance data for those two metrics were 0.706 and 0.656.

Table 5: Performance of LGBM Classifier on well-balanced data
(under-sampled and then over-sampled data; cv=5)

	Balanced Accuracy	**AUC Score**	**Weighted F1 Score**
LGBM(boosting_type='gbdt', n_estimators = 5000, learning_rate=0.03, max_depth=-1) on n-grams + meta feats + EDA feats	0.804	0.933	0.793

5 Conclusion and Future Work

In this final project, I was able to create a machine learning model that achieved 80% balanced accuracy which is higher than the baseline score of 73% accuracy achieved by the Grover paper, which secured 1st place in the leaderboard for fake news classification accuracy (but lower than the best score of 92% accuracy). This result is significant because my model was a multi-class classification model that can discriminate between not only real and fake news but also fake news and sarcastic news while the Grover paper was still confined to a binary (real v.s. fake) setting. In addition, I did not use any text from the body of the articles but instead used only the "titles" of articles to create my model. Thus, I demonstrated that researchers can build strong models even with little text data, lower computational cost and less memory usage.

Moreover, I have shown that the creation of new hand-made features such as the median length of article titles contribute to the improvement of model performance. I proved that resolving class imbalance issues via Tomek Link under-sampling and SMOTE over-sampling leads to more robust models with huge increases in AUC score and accuracy. This will be meaningful when researchers are lacking data in a certain class (e.g. fake news) considering how it is difficult to collect enough fake news data that are already labeled as fake unless you have an outstanding web scraper that can do so.

The limitation of this model I built is the potential inability to classify new articles that are grossly different in nature from the data I collected. In my data, most of the real and fake news come from politics related sources. If a new unseen article from a completely different category (e.g. health, technology) is given to my model, the performance may not be as good.

Future work includes diversifying the sources from which I collect the articles, attempting new

feature representations including Part-Of-Speech tagging, Word2Vec, Doc2Vec, using neural networks including CNN, LSTM and using stacking, which is an ensemble learning technique that combines multiple classification models via a meta-classifier, to get an extra boost in performance.

6 References

[1] *"Defending Against Neural Fake News"*, Yonatan Bisk, Yejin Choi, Ali Farhadi, Ari Holtzman, Hannah Rashkin, Franziska Roesner, and Rowan Zellers, https://arxiv.org/pdf/1905.12616v2.pdf

[2] *"dEFEND: Explainable Fake News Detection,"* Kai Shu, Limeng Cui, Suhang Wang, Dongwon Lee, and Huan Liu, http://pike.psu.edu/publications/kdd19.pdf

[3] *"Fake News Research: Theories, Detection Strategies, and Open Problems,"* Reza Zafarani, Xinyi Zhou, Kai Shu, and Huan Liu, https://dl.acm.org/doi/pdf/10.1145/3292500.3332287

[4] *"CSI: A hybrid deep model for fake news detection,"* Natali Ruchansky, Sungyong Seo, and Yan Liu, 2017, https://arxiv.org/abs/1703.06959.

[5] *"Rumor Detection with Hierarchical Social Attention Network,"* Han Guo, Juan Cao, Yazi Zhang, Junbo Guo, and Jintao Li, 2018, https://dl.acm.org/doi/10.1145/3269206.3271709.

[6] *"Bagging and Random Forest Ensemble Algorithms for Machine Learning,"* Jason Brownlee, https://machinelearningmastery.com/bagging-and-random-forest-ensemble-algorithms-for-machine-learning/.

[7] *"LGBM Documentation,"* https://lightgbm.readthedocs.io/en/latest/.

[8] *"Boosting (machine learning),"* https://en.wikipedia.org/wiki/Boosting_(machine_learning).