{12} - An NLP Approach For Satire (Onion) and Fake News

Authors: Ko Gi Hun, Daryl Koh Yi Kai
Email: e0318604@u.nus.edu, e0325843@u.nus.edu

Abstract

Satire is a genre that falls through the cracks in fake news classifiers, which are usually binary. This project aims to use natural language features and machine learning techniques to distinguish satirical headlines from real and fake news headlines. In our work, we evaluate various combinations of machine learning and deep learning networks with natural language features. We also take a microscopic look at data that is consistently mislabeled, and look for insights on the genre of satire and its relation to fake and real news.

1 Introduction

The war against fake news has received much attention, with some academics even deeming it a crisis [12]. The Internet and its vast amount of data has been tested against a number of fake news detection approaches. However, the genre of satire has received relatively little attention. Satire shames shortcomings, usually to inspire social change or improvement [5]. Satirical news articles, by definition, are fake. But does that make them equally detrimental as fake news? George Orwell's Animal Farm is a piece of satire that is a part of many school syllabuses, and its educational value persists till today.

This raises the question: Can we differentiate satire from fake news, with the help of NLP?

Satire and fake news originate from different intentions - the goal of satire is to mock and criticize with its obvious untruthfulness, but fake news aims to deceive and appear as true. Our team wanted to find out if we could use Natural Language Processing features to train a classifier model that can not only distinguish between satire and fake news, but do so in the presence of actual news, as it appears in the real world. Successful classification models would have positive impacts on downstream applications (e.g. knowledge base population) as well as give more insight into the linguistic features of satire.

Our main research questions are therefore: RQ1) can we successfully train a classifier to distinguish between real, fake and satirical headlines and RQ2) what insights can we draw about the nature of satire from these classifiers' performance?

2 Related Work

Previous work largely explores the binary classification of satirical news from real news [10][17], or between satirical and fake news [9]. However, this is not representative of the real-life situations, where real news, fake news, and satirical news are circulated equally, and can be mistaken for each other [6]. An automatic system for satire detection is computationally difficult, where even humans are fallible when it comes to picking out satirical news [2].

Within the context of binary classification, some methods previously proposed include: analysis based on textual content of articles [9], manually engineered features including bag-of-words, n-grams, lexical features [4] [14], as well as deep learning methods for satire detection [17].

Based on the previously mentioned limitations of previous works, this project will attempt the multinomial classification of news headlines into 3 classes: real, fake and satirical. The methods we used are similar to those used in previous methods, including simple classifier models, as well as deep learning methods with natural language features.

3 Method

3.1 Dataset Analysis & Data Preprocessing

Our team mainly used 2 distinct datasets for analysis and training:

a. OnionOrNot Dataset (Kaggle) :
   24k News headlines from The Onion (a satirical news website) & real news headlines from subreddit r/NotTheOnion [8]
   Label 0: not onion, Label 1: onion (satirical)
b. Fake News Dataset (University of Victoria):
   44.8k News headlines from Reuters and various fake news sources [1]
   Label 0: true news, Label 2: fake news

Both datasets were merged and randomly shuffled, before data was preprocessed. For a total of 68898 headlines. One problem we noticed was the resulting uneven distribution from merging the datasets, which pose risks for affecting the model's performance during training and prediction.

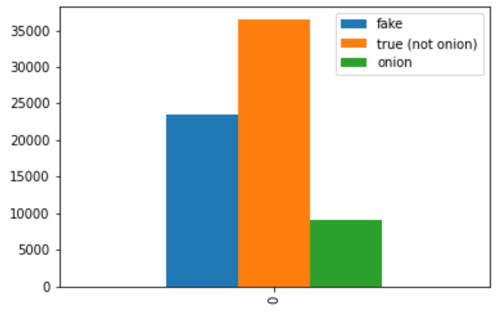


Figure1. Merged Dataset Data Distribution Chart by Label

From figure 1 above, there is more data labelled as 'true', followed by 'fake' and 'onion' data. Hence it is likely that the model might overfit with the data labelled 'onion' due to insufficient amount of text data, and it was later taken to account when evaluating the model's performance in our findings.

3.2 Feature Extraction(POS tagging)

First, we performed POS-tagging analysis on the merged dataset, with NLTK's POS tagger. The charts below show the POS-Tag histogram for each headline class.

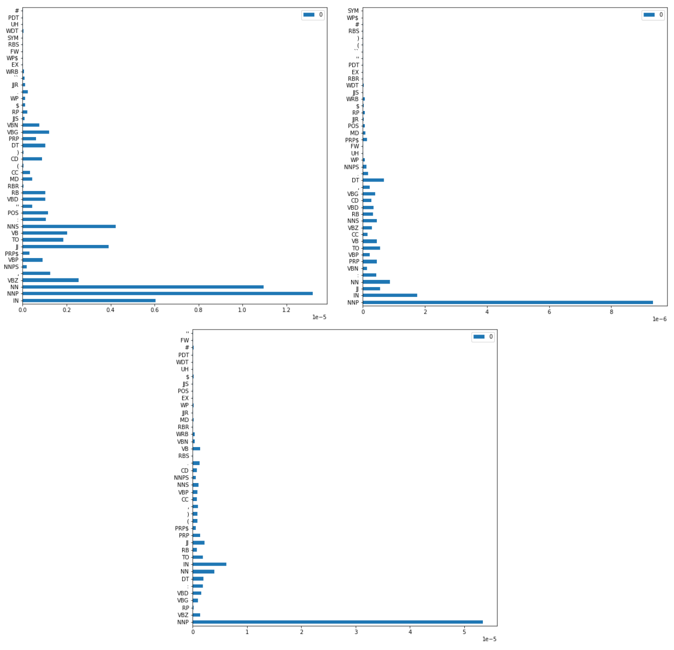


Figure 2. POS-Tag Histogram
(left: Real Labelled Data, right: Onion Labelled Data, bottom: Fake Labelled Data)

From figure 2, it can be seen that data that is labelled as 'real', which refers to real news headlines, contain a larger variety of tags. All three labels contain 'NNP', or Proper Noun as the most observed POS-tag, possibly due to the dense summarizing required of news headlines. There were also several findings that we have discovered from the histogram charts.

Both onion (satire) and fake news data have the 'IN' (Preposition or subordinating conjunction) tag as the second most common. Among satirical headlines, these tags are more common:

- RB: Adverb
- VBD: Verb, past tense
- DT: Determiner
- NNS: Noun, plural

As the quantitative distribution of POS tags between fake and satirical news was relatively similar, it was difficult to draw conclusive insights about the benefits of POS tagging as a feature to use for extraction.

3.3 Feature Extraction (TF-IDF)

Term frequency–inverse document frequency (TF-IDF) is simple, but an impactful feature extraction method we used among different methods we have applied for training the classifier model. TF-IDF vectors were generated as features, first by tokenizing headlines into vectors of unigrams and bigrams, then applying tf-idf normalization (with scikit-learn libraries).

3.4 Feature Extraction (W2V)

Word2vec [11] is a context-free model that generates a single word-embedding for each word in the vocabulary. We used word2vec models pre-trained from the Google News dataset: 300-dimensional vectors for 3 million words and phrases.

Single w2v vectors were obtained for each data point(headline) by tokenizing the headlines, and obtaining a numerical average of all the w2v vectors of the component words.

3.5 Feature Extraction (Parsing Based Lexicon Generation Algorithm)

One feature extraction attempted but ultimately not included in the training models was the Parsing Based Lexicon Generation Algorithm(PBLGA) [3]. It uses sentiment analysis to detect sarcasm by looking for phrases with positive sentiment in close proximity to a negative situation phrase. The proposed algorithm returns a binary result for the presence of sarcasm, but when tested on various data points from each of the headline classes, was not suitable for detecting clear examples of sarcasm. Furthermore, sarcasm is not a compulsory feature of satire, and hence this feature was not included when training the classifier models.

3.6 Classifier Models Used

After extracting different sets of features, we test them on three main categories of classifier models:

1. Non-neural Classifier Models
2. Convolutional Neural Networks (CNN)
3. Recursive Neural Networks (RNN)

Scikit-Learn, together with the PyTorch framework, was used in a Python environment for development.

3.7 Non-neural Classifier Models

Simple classifier models were the first approach used, as a baseline for comparison against other deep learning classifier models. The Scikit-Learn library was mainly used to perform data pre-processing, feature extraction, training and evaluation.

Below is the pipeline of the training process our team used for training the non-neural classifier models:

1. Split the merged dataset into train and test dataset, using random shuffling
2. Vectorize the corpus with N-Gram range (unigram and bigram) for both train and validation datasets
3. Transform count vectors into TF-IDF vectors
4. Feed the transformed TF-IDF vectors into the classifier model for training
5. Evaluate the model by plotting confusion matrix and calculating Macro F1-score.

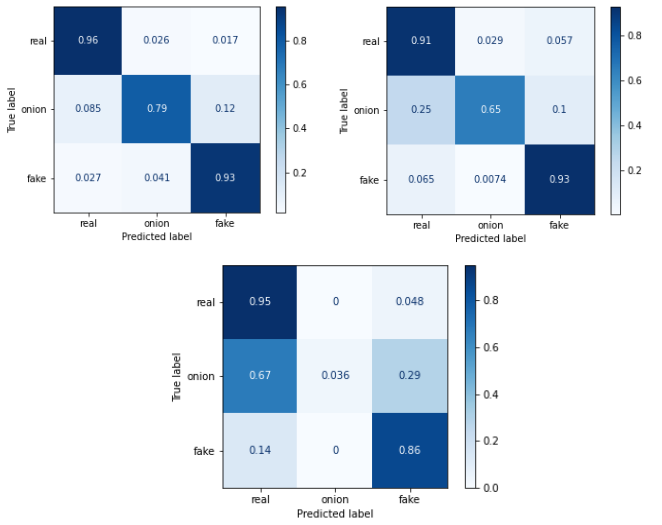


Figure 3. Confusion Matrices from training TF-IDF features over: Support Vector Machine (left), Logistic Regression (right), Multinomial Naive Bayes (center)

The results of these classifier models were that Support Vector Machine (SVM) seemed to classify the best compared to the rest of the models. All three models performed poorly classifying the satirical (onion) headlines, especially the Multinomial Naive Bayes model.

For example, from Figure 3, the Logistic Regression model predicted 65% of the entire onion labelled data correctly, from the validation dataset, whereas the support vector machine model predicted 79% of the data correctly. The Multinomial Naive Bayes (NB) model was particularly poor at labelling satirical headlines (3.6% accuracy). Due to its unsuitability for working with negative numbers, the Multinomial NB model was not trained with W2V features.

3.8 CNN

Next, Our team experimented with a Convolutional Neural Network, with TF-IDF features extracted from the dataset as input. Unfortunately, unlike inbuilt classifier models provided by scikit-learn library, using CNN with PyTorch framework had trouble handling vectors with more than 30 thousand columns (dimensions), causing crashes from insufficient memory.

Thus, the TF-IDF vectors had to be shrinked and we reduced the dimensions to the k-highest features across the whole corpus. We then attempted the training with different dimension numbers from 1000-4000.

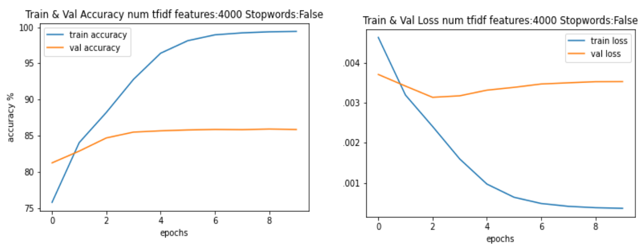


Figure 4. Training and Validation Accuracy (left) and Loss (right) over the CNN model for TF-IDF vectors (4000 dimensions)

From figure 4 above, it is likely to deduce that the model is overfitting, as its accuracy rate halts at 0.85 and remains the same after only 2 epochs of training. For the loss graph on the right, the loss values begin to rise after two epochs of training. This training result is based on using the highest number of features (4000) out of the range from 1000-4000.

3.9 RNN

Lastly, the final approach used was using a RNN model, with LSTM (Long Short-Term Memory) layers.

The model features are as follows:

- Embedding Layer
- Dropout Layer (p=0.3)
- LSTM Layer
- Linear Layer
- Training Loop: Adam optimizer, Cross entropy loss

No feature extraction was used during training, but tokenisation and lemmatisation was still applied to each news sentence from the same dataset. With 30 epochs, our team was able to observe clear improvements in accuracy and F1 score across all labels.

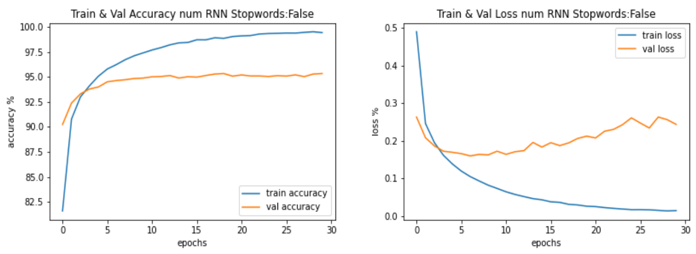


Figure 5. Train & Val Accuracy Graph RNN (left), Train & Val Loss Graph RNN (right)

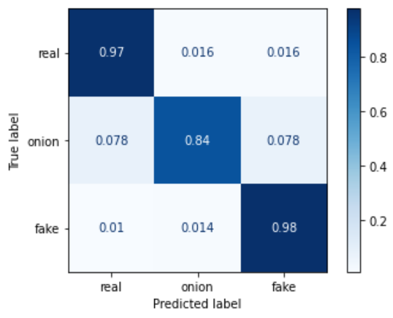


Figure 6. Confusion Matrix with RNN

Compared with the confusion matrix produced using Scikit-Learn classifier models with TF-IDF vector data, RNN seems to predict much better, without any textual processing of the dataset during training. The model can predict 84% of the onion labelled data correctly, with 97% and 98% for real and fake labelled data respectively. This was not expected by the team, as we were expecting it to use any extensive feature extraction method to achieve such results.

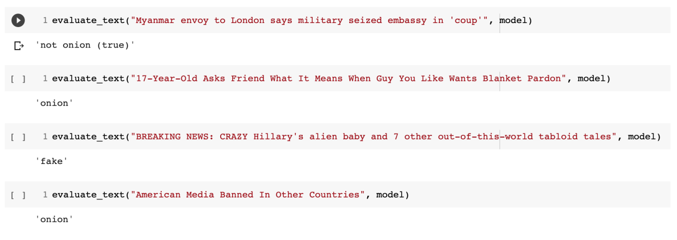
4 Evaluation

The table below describes the summary of the entire combination of classifier models, with each different feature extraction method used for training. The column 'F1-Score' displays the validation metric using macro F1-score.

Classifier	Feature Extraction	F1-Score
Multinomial Naive Bayes (NB)	TF-IDF	0.59007
Logistic Regression	TF-IDF	0.84623
Logistic Regression	W2V	0.89190
SVM	TF-IDF	0.85435
SVM	W2V	0.89309
CNN	TF-IDF (num_features=4000)	0.85
CNN	TF-IDF (num_features=3000)	0.852
CNN	TF-IDF (num_features=2000)	0.827
CNN	TF-IDF (num_features=1000)	0.82
RNN (LSTM)	None (Tokenization & Lemmatization only)	**0.932**

Figure 7. Summary of Validation Score with each classifier model with used training dataset for each feature extraction method

5 Discussion

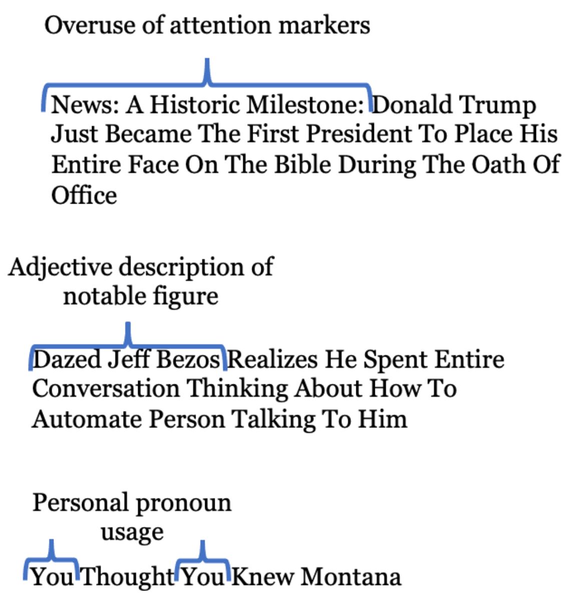


```
evaluate_text("Myanmar envoy to London says military seized embassy in 'coup'", model)
'not onion (true)'

evaluate_text("17-Year-Old Asks Friend What It Means When Guy You Like Wants Blanket Pardon", model)
'onion'

evaluate_text("BREAKING NEWS: CRAZY Hillary's alien baby and 7 other out-of-this-world tabloid tales", model)
'fake'

evaluate_text("American Media Banned In Other Countries", model)
'onion'
```

Figure 8. Screenshot taken from Google Colab Notebook using trained RNN classifier model

Although W2V vectors (which capture semantic information) combined with SVM training models came close, the RNN still outperformed all other non-neural classifiers. We attribute this to the ability of the RNN to capture context in wording. As can be observed from Figs. 3 and 6, a consistently problematic category in the confusion matrices was for satirical (onion) headlines to be classified as fake news headlines.

Microanalysis of some samples from this category (satirical headlines labeled as fake) was done, and stereotypical features of fake news [15] [16] were annotated in Fig. 9:


Overuse of attention markers
Adjective description of notable figure
Personal pronoun usage

News: A Historic Milestone: Donald Trump Just Became The First President To Place His Entire Face On The Bible During The Oath Of Office

Dazed Jeff Bezos Realizes He Spent Entire Conversation Thinking About How To Automate Person Talking To Him

You Thought You Knew Montana

Figure 9: Annotation of sample mislabeled headlines (RNN classifier)

As satire mocks by imitation, we observe that when satirical headlines imitate fake news, the grammatical/stylistic features that are used can be mistaken for actual fake news. In this case, background, or common-sense knowledge features could be useful in separating satirically "fake news" headlines from actual fake news headlines [7].

To answer our two RQs posed at the start:

RQ1) Can we successfully train a classifier to distinguish between real, fake and satirical headlines?

The best classifier (RNN) performed significantly better than the baseline models (NB and LR with TF-IDF), so RNNs show promise for such multi-class classification tasks.

RQ2) what insights can we draw about the nature of satire from these classifiers' performance?

From the RNN's significantly improved performance, the importance of context and positioning between words seems to play a part (Recurrent Neural Networks with LSTM layers use previous data points as context to make predictions). However, writing styles/patterns may also create a source of bias that allows the models to simply guess the right label by detecting writing style (as opposed to semantic content and meaning).

6 Conclusion

Although NLP features combined with supervised learning models came close, in the end Recurrent Neural Networks still were the most effective at classification, although the interesting phenomena where satirical headlines were wrongly classified as fake news shows the potential overlap between the two (when the headlines aim to mock in the style of fake news).

Due to manpower and time limitations, our work would have benefited from more manual feature engineering, which may increase the efficacy of the non-neural classifiers to match the deep learning models. Furthermore, as pointed out in [10], stylistic features of individual satirical sites may be picked up by the models when restricted to one news source. Future work may consider sourcing satirical news headlines from more sources.

7 References

[1] Ahmed, H., Traore, I., & Saad, S. (2018). Detecting opinion spams and fake news using text classification. *Security and Privacy*, *1*(1), e9. https://doi.org/10.1002/spy2.9

[2] Allcott, H., & Gentzkow, M. (2017). Social media and fake news in the 2016 election. *Journal of Economic Perspectives*, *31*(2), 211–236. https://doi.org/10.1257/jep.31.2.211

[3] Bharti, S. K., Babu, K. S., & Jena, S. K. (2015). Parsing-based sarcasm sentiment recognition in twitter data. *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015*, 1373–1380. https://doi.org/10.1145/2808797.2808910

[4] Burfoot, C., & Baldwin, T. (2009). Automatic satire detection: Are you having a laugh? *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, 161–164. https://www.aclweb.org/anthology/P09-2041

[5] Elliott, R. C. (2004). *Satire | definition & examples*. Encyclopedia Britannica. https://www.britannica.com/art/satire

[6] Garrett, R. K., Bond, R., & Poulsen, S. (2019, August 16). *Too many people think satirical news is real*. The Conversation. http://theconversation.com/too-many-people-think-satirical-news-is-real-121666

[7] Goldwasser, D., & Zhang, X. (2016). Understanding satirical articles using common-sense. *Transactions of the Association for Computational Linguistics*, *4*, 537–549. https://doi.org/10.1162/tacl_a_00116

[8] Gorgolewski, C. (2020, August 26). *Onion or not*. https://kaggle.com/chrisfilo/onion-or-not

[9] Levi, O., Hosseini, P., Diab, M., & Broniatowski, D. A. (2019). Identifying nuances in fake news vs. Satire: Using semantic and linguistic cues. *ArXiv:1910.01160 [Cs]*. https://doi.org/10.18653/v1/D19-5004

[10] McHardy, R., Adel, H., & Klinger, R. (2019). Adversarial training for satire detection: Controlling for confounding variables. *Proceedings of the 2019 Conference of the North*, 660–665. https://doi.org/10.18653/v1/N19-1069

[11] Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *ArXiv:1310.4546 [Cs, Stat]*. http://arxiv.org/abs/1310.4546

[12] Nelson, J. L., & Taneja, H. (2018). The small, disloyal fake news audience: The role of audience availability in fake news consumption. *New Media & Society*, *20*(10), 3720–3737. https://doi.org/10.1177/1461444818758715

[13] NS, A. (2020, April 7). *Multiclass text classification using lstm in pytorch*. Medium. https://towardsdatascience.com/multiclass-text-classification-using-lstm-in-pytorch-eac56baed8df

[14] Rubin, V., Conroy, N., Chen, Y., & Cornwell, S. (2016). Fake news or truth? Using satirical cues to detect potentially misleading news. *Proceedings of the Second Workshop on Computational Approaches to Deception Detection*, 7–17. https://doi.org/10.18653/v1/W16-0802

[15] Tompkins, J. (2019). Disinformation Detection: A review of linguistic feature selection and classification models in news veracity assessments. *ArXiv:1910.12073 [Cs]*. http://arxiv.org/abs/1910.12073

[16] Vereshchaka, A., Cosimini, S., & Dong, W. (2020). Analyzing and distinguishing fake and real news to mitigate the problem of disinformation. *Computational and Mathematical Organization Theory*, *26*(3), 350–364. https://doi.org/10.1007/s10588-020-09307-8

[17] Yang, F., Mukherjee, A., & Dragut, E. (2017). Satirical news detection and analysis using attention mechanism and linguistic features. *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 1979–1989. https://doi.org/10.18653/v1/D17-1211