FINAL PROJECT REPORT

On

"EVALUATING AND ENHANCING TRUSTWORTHINESS OF TEXT"

COEN 296

BY

Aditya Randive	1409938
Arpita Singh	1412594
Lucas Huang	1266230
Shail Shah	1388286

ACKNOWLEDGEMENT

We would like to express our heartfelt gratitude to Prof. Ming-Hwa Wang for providing us with an opportunity to explore our interests in Natural Language Processing. Without his tremendous support, encouragement as well as valuable inputs, this project couldn't have materialized. The support received from all the members who contributed to this project was vital for our success.

TABLE OF CONTENT

ABSTRACT

Satire is an attractive subject in deception detection research: it is a type of deception that intentionally incorporates cues revealing its own deceptiveness. Whereas other types of fabrications aim to instill a false sense of truth in the reader, a successful satirical hoax must eventually be exposed as a jest. We propose a detection methodology that provides an effective tool to identify satire and humor, elaborating and illustrating the unique features of satirical news, which mimics the format and style of journalistic reporting. One of the main corpus we are proposing to use is the S-N-L database. In the S-N-L database, satirical news stories are carefully matched and examined in contrast with their legitimate news counterparts in 12 contemporary news topics in 4 domains (civics, science, business, and "soft" news). As conceptualized in the referring paper, we propose to design an SVM-based algorithm, enriched with 5 predictive features (Absurdity, Humor, Grammar, Negative Affect, and Punctuation) to be tested on their combinations on the referred corpus. Our aim is to achieve a F-score upward of 80% so that algorithmically identifying satirical news pieces can aid in minimizing the potential deceptive impact of satire.

1] INTRODUCTION

In 2016, the prominence of disinformation within American political discourse was the subject of substantial attention, particularly following the surprise election of President Trump. The term ‘fake news’ became common parlance for the issue, particularly to describe factually incorrect and misleading articles published mostly for the purpose of making money through pageviews. In this project, we seek to produce a model that can accurately predict the likelihood that a given article is fake news.

Facebook has been at the epicenter of much critique following media attention. They have already implemented a feature for users to flag fake news on the site; however, it is clear from their public announcements that they are actively researching their ability to distinguish these articles in an automated way. Indeed, it is not an easy task. A given algorithm must be politically unbiased--since fake news exists on both ends of the spectrum--and also give equal balance to legitimate news sources on either ends of the spectrum. In addition, the question of legitimacy is a difficult one: what makes a news site ‘legitimate’? Can this be determined in an objective way?

In this research summary we compare the performance of models using three distinct feature sets to understand what factors are most predictive of fake news: tf-idf using bi-gram frequency, syntactical structure frequency (probabilistic context free grammars, or PCFGs), and a combined feature union. In doing so, we follow the existing literature on deception detection through natural language processing, particularly the work of Feng, Banerjee, and Choi (2012) with deceptive social media reviews. We find that while bi-gram TFIDF yields predictive models that are highly effective at classifying articles from unreliable sources, the PCFG features do little to add to the models’ efficacy. Instead, our findings suggest that, contrary to the work of Feng, Banerjee, and Choi’s application, PCFGs do not provide meaningful variation for this particular classification task. This suggests important differences between deceptive reviews and so-called ‘fake news’. We then suggest additional routes for work and analysis moving forward.

2] THEORETICAL BASES AND LITERATURE REVIEW:

2.1] Definition of the problem

The main problem with increasing the trustworthiness of text in context to news is that it is always going to be a **reactive rather than proactive process**. This is because it is impractical to keep a check and on generation of new text and it's subsequent publishing. Moreover, there are no fix sets of corpora that we can absolutely rely upon. New text is generated daily and is accessible readily at an instant because of the capability of the modern age internet.

The fundamental purpose of designing a system to improve trustworthiness of text is to flag and control untrustworthy text's spread as early as possible so that the readership is limited.
So we need an automated assistive tool for both - the content creators as well as the readers to correctly identify and flag out false content.

2.2] Theoretical background of the problem:

In the course of text creation in the form of news production, dissemination, and consumption, there are ample opportunities to deceive and be deceived. Direct falsifications such as journalistic fraud or social media hoaxes pose obvious predicaments. While fake or satirical news may be less malicious, they may still **mislead inattentive readers**. Taken at **face value**, satirical news can intentionally create a false belief in the readers' minds, per classical definitions of deception.

News satire is a genre of satire that mimics the format and style of journalistic reporting. The fake news stories are typically inspired by real ones, and cover the same range of subject matter: from politics to weather to crime. The satirical aspect arises when the factual basis of the story is "comically extended to a fictitious construction where it becomes incongruous or even absurd, in a way that intersects entertainment with criticism" . News satire is most often presented in the **Horatian style**, where humor softens the impact of harshness of the critique – the **spoonful of sugar that helps the medicine go down**. More than mere lampoon, untrustworthy news stories aim to "arouse the reader's' attention, amuse them, and at the same time awaken their capacity to judge contemporary society".

Several factors contribute to the believability of fake news online. Recent polls have found that only 60% of Americans read beyond the headline (The Media Insight Project, 2014). Furthermore, on social media platforms like Facebook and Twitter, stories which are "liked" or "shared" all appear in a common visual format. Unless a user looks specifically for the **source attribution**, an article from The Onion looks just like an article from a credible source, like The New York Times. In an effort to counteract this trend, we propose the creation of an **automatic satire detection system**.

2.3] Related research to solve the problem:

There exists a sizeable body of research on the topic of machine methods for satire and stance detection, most of which has been focused on classifying online reviews and publicly available social media posts (Rubin, 2017). Particularly since late 2015 during the American Presidential election, the question of determining 'fake news' has also been the subject of particular attention within the literature.

The major research paper the on which we are proposing solution to the problem is : *"Fake News or Truth? Using Satirical Cues to Detect Potentially Misleading News" - Victoria L. Rubin, Niall J. Conroy, Yimin Chen, and Sarah Cornwell*[1]. This research paper proposes a SVM-based algorithm consisting of five predictive features.

2.4] Shortcoming of related research:

The problem while filtering out untrustworthy text is the factor of "**false negative**". It may happen that an article with valid contents may get classified as untrustworthy. Also, in case of non-news fun article which is written in a funny language may get flagged. This is not a desired result and hence a more refined approach is needed for such exceptional cases.

2.5] Other approaches and differences with chosen approach

The papers that we have referred outline several approaches that seem promising toward the aim of correctly classifying misleading articles. They note that simple content-related n-grams and

shallow part-of-speech (POS) tagging have proven insufficient toward the classification task, often failing to account for important context information. Rather, these methods have been shown useful only in tandem with more complex methods of analysis.

Deep Syntax analysis using Probabilistic Context Free Grammars (PCFG) have been shown to be particularly valuable in combination with n-gram methods. Feng, Banerjee, and Choi (2012) are able to achieve **85% to 91%** accuracy in deception related classification tasks using online review corpora.

Other Approaches:

- Feng & Hirst (2013) implement a Semantic Analyzer looking at object:descriptor pairs for contradictions with the text on top of Feng's initial deep syntax model for additional improvement.
- Rubin & Lukoianova (2014) analyze rhetorical structure using a vector space model with similar success.
- Sentiment and fact-based argument analysis (Pang & Lee, 2008).
- Language pattern similarity networks (Ciampaglia et al., 2015) requiring a pre-existing knowledge base.
- Social networks using inter-article links using centering resonance analysis (Papacharissi & Oliviera, 2012).

2.6] Why your solution is better:

Considering the previous as well as the ongoing research pertaining to untrustworthy text, the component of 'funny' or 'satire' can be found as a common thread in many such corpora. Hence, if we tackle this component, a major and important factor in eliminating untrustworthy text may be achieved. This component can then be combined with other factors to make a more robust system. Hence our solution is better because it considers solving a major problem.

Ultimately, there is still much work to be done within the field to advance the work toward a well functioning model for detection.

3] HYPOTHESIS

In this project we implement the classification models(Support Vector Machines(SVM), Stochastic Gradient Descent (SGD), Gradient Boosting (GB), Bounded Decision Trees (DT), Random Forests (RF)) using SciKit - Learn and evaluate them to understand what factors are most predictive of fake news using features set below:

1. **Bigram Term Frequency-Inverse Document Frequency** - it is a vectorized bigram Term Frequency-Inverse Document Frequency. This is a weighted measure of how often a particular bigram phrase occurs in a document relative to how often the bigram phrase occurs across all documents in a corpus.
2. **Normalized frequency of parsed syntactical dependencies** - for this we use Spacy to tokenize and parse syntactical dependencies of each document.

4] METHODOLOGY:

As mentioned previously, we have two fundamental measurements to leverage fake news detection. They are stance detection from Fake News Challenge (FNC) (http://www.fakenewschallenge.org/) and satire detection from University of Western Ontario (http://victoriarubin.fims.uwo.ca/). We will experiment and find a suitable methodology by utilizing theirs.

4.1] How to generate/collect input data:

We collect Stance Detection dataset for FNC from a Github repository: https://github.com/FakeNewsChallenge/fnc-1/tree/29d473af2d15278f0464d5e41e4cbe7eb58231f2. The data provided is in (headline, body, stance) instances, where stance is one of {unrelated, discuss, agree, disagree}. The dataset is provided as two CSVs: train_bodies.csv and train_stances.csv. The train_bodies.csv contains the body text of articles (the articleBody column) with corresponding IDs (Body ID). The train_stances.csv contains the labeled stances (the Stance column) for pairs of article headlines (Headline) and article bodies (Body ID, referring to entries in train_bodies.csv).

We collect Satire Detection dataset directly from the Associate Prof. Victoria Rubin who is a director of the Language and Information Technology Lab at Faculty of Information and Media Studies, Western University, London, Canada. This collection was part of the 2015-2018 News Verification Project funded by the Social Sciences and Humanities Research Council of Canada (SSHRC). The first 240 articles (published in 2015) were aggregated into a 2x2x12 design (US and Canadian; satirical and legitimate online news; varying across 4 domains (civics, science, business, and “soft” news) with 3 distinct topics within each domain (see labels in Column E). The 240 news pieces were carefully selected with an equal representation (5 articles per subtopic listed in Column E). An additional set of 120 articles was collected from online publications in 2016, to expand the inventory of sources and topics and to serve as a reliability test for the manual findings within the first set, the second set was still evenly distributed between satirical and legitimate news.

4.2] How to solve the problem:

4.2.1] Algorithm design:

As for stance detection, the classifier utilized in this model is Gradient Boosted Trees. An exceptionally efficient implementation of GBDT is XGBoost. In Satire Detection, the text classification pipeline was scripted in Python and used the scikit-learn open source machine learning package as the SVM classification (http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html).

There is still an ongoing discussion which machine learning algorithm is the finest. Perhaps there is no such thing. Even if the algorithmic concept is identical, various implementations could result differently with the same dataset. It might also vary depending on the input data. Excepting for applying the same algorithms, we also plan to design a pipeline to combine features from our two main directions and to experiment a few machine learning algorithms with different machine learning frameworks, for instance, TensorFlow.

4.2.2] Language (to be) used:

- Python 3.x for the project

4.2.3] Tools (to be) used:

- NLTK toolkit
- Scipy Stack: numpy, scipy and pandas
- Gensim (for tf-idf and word2vec)
- Scikit-Learn
- TensorFlow
- PyCharm CE
- MatplotLib

4.3] How to generate output:

First, we utilize the NLTK toolkit to perform some pre-processing on the input dataset. The labels are encoded into numeric target values, for instance 1, 2, 3, and 4. The text of headline and body are then tokenized and stemmed. Finally Uni-grams, bi-grams and tri-grams are created out

of the list of tokens. These grams and the original text are used by the following feature extractor modules.

Next step is feature engineering. There are several of them:

- Basic count takes the uni-grams, bi-grams and tri-grams and creates various counts and ratios which could potentially signify how a body text is related to a headline.
- TF-IDF constructs representations of the headline and body by calculating the Term-Frequency of each gram and normalize it by its Inverse-Document Frequency in order to reflect how important the headline is to the body.
- As an extension of TF-IDF feature, Singular-Value Decomposition is also applied to them to obtain a compact, dense vector representation of the headline and body respectively feature for enhancing the accuracy on whether the body is related to the headline or not.
- Absurdity feature is implemented by using Part of Speech tagger and Named Entity Recognizer from NLTK toolkit. We defined the list as the non-empty set (LNE), and compared this with the set (NE) of named entities appearing in the remaining article. The article was deemed absurd when the intersection (LNE ∩ NE) was empty (0=non-absurd, 1=absurd).

And more secondary features to consider:

- Word2Vec feature were trained on a Google News corpus with 100 billion words and a vocabulary size of 3 million. The resulting word vectors can be used to find synonyms, predict the next word given the previous words, or to manipulate semantics. For the current problem constructing the vector representation out of word vectors could potentially overcome the ambiguities introduced by the fact that headline and body may use synonyms instead of exact words.
- Sentiment feature uses the Sentiment Analyzer in the NLTK package to assign a sentiment polarity score to the headline and body separately. This score can be informative of whether the body is being positive about a subject while the headline is being negative. But it does not indicate whether it's the same subject that appears in the body and headline; however, this piece of information should be preserved in other features.
- Humor (Hum) detection was based on the premises of opposing scripts and maximizing semantic distance between two statements as method of punchline identification (Mihalcea et al., 2010). Similarly, in a humorous article, the lead and final sentence are minimally related. Our modification of the punchline detection method assigned the binary value (humor=1) when the relatedness between the first and last article sentences was the minimum with respect to the remaining sentences.

- Grammar (Gram) feature vector was the set of normalized term frequencies matched against the Linguistic Inquiry and Word Count (LIWC) 2015 dictionaries, which accounts for the percentage of words that reflect different linguistic categories (Pennebaker, Boyd, Jordan, & Blackburn, 2015). We counted the presence of parts of speech terms including adjectives, adverbs, pronouns, conjunctions, and prepositions, and assigned each normalized value as the element in a feature array representing grammar properties.
- Negative Affect (Neg) and Punctuation (Pun) were assigned as feature weights representing normalized frequencies based on term-for-term comparisons with LIWC 2015 dictionaries. Values were assigned based on the presence of negative affect terms and punctuation (periods, comma, colon, semicolon, question marks, exclamation, quotes) in the training and test set.

Then, we apply those features to our pipeline and utilize either Scikit-learn or TensorFlow to experiment the result.

4.4] How to test against hypothesis:

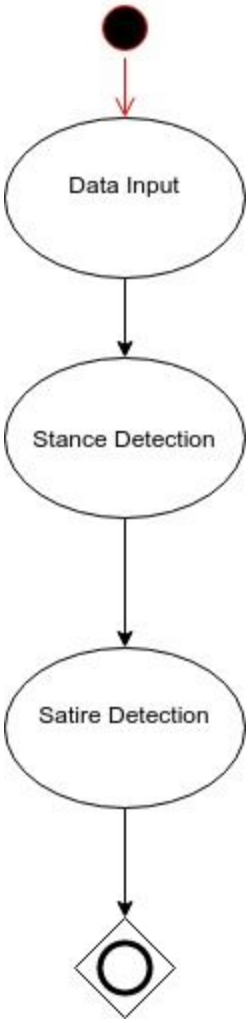
- 10-fold cross validation confidence score: In k-fold cross-validation, the original sample is randomly partitioned into k equal sized subsamples. Of the k subsamples, a single subsample is retained as the validation data for testing the model, and the remaining k − 1 subsamples are used as training data.
- F-score measurement: p is the number of correct positive results divided by the number of all positive results, and r is the number of correct positive results divided by the number of positive results that should have been returned. F1 = 2 · precision · recall / precision + recall

5] IMPLEMENTATION:

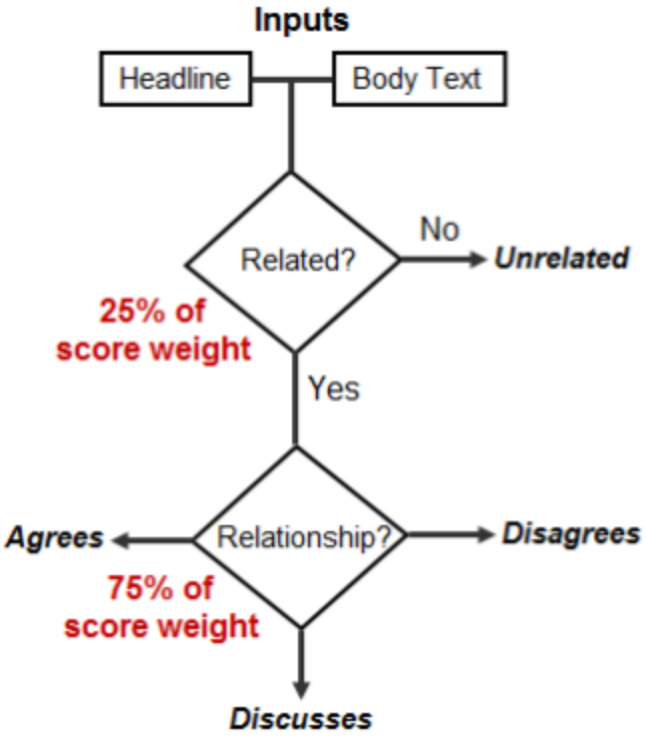
5.1] Code:

- Stance detection: https://github.com/Lucashuang0802/FNC-Project-Stance
- Satire detection: https://github.com/AdityaRandive/satire

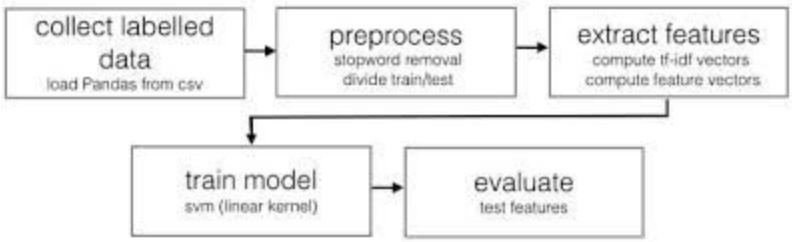
5.2] Design document and flowchart:



5.2.1] Stance Detection :



5.2.2] Satire Detection :



6] DATA ANALYSIS AND DISCUSSION:

6.1] Output generation:

6.1.1] Satire:

- **build_model()** This uses the scikit-learn machine learning library to calculate the standard tf.idf values for terms in each document, and to train and evaluate an SVM classifier.
- **enhance_terms()** This uses the spaCy NLP library to first identify named entities corresponding to people or organisations. It then uses spaCy's dependency parser to identify the corresponding noun chunks and the associated verbs. These are enhanced by simply appending multiple copies of these words at the end of the document. Because we are using a tf.idf representation, the word order does not matter, so the effect is simply to increase the significance of these terms. The motivation is that much satire is about famous or influential people or organisations, so the words corresponding to these targets and their actions are likely to be especially significant.
- **train_test()** This is the top level function that calls others to load the data, enhance the documents, split into training and test (evaluation) sets, build and evaluate a classifier model, and display the results. For simplicity, we train and test on non-overlapping subsets of Baldwin's 'training set', and ignore his 'test set' of articles. This is unlikely to significantly change the results.

6.1.2] Stance detection:

- A binary file named stance.pickle: Since we have coded the project in Python 3, the pickle python module is utilized to save and load a trained model for future development. The pickle module implements a fundamental, but powerful algorithm for serializing and de-serializing a Python object structure.
- Evaluation files saved as .txt:
    - confusion_matrix.txt: The confusion matrix contains four metadata: "agree", "disagree", "discuss", and "unrelated". A confusion matrix is a table that is often used to describe the performance of a classification model(or "classifier") on a set of test data for which the true values are known. The confusion matrix itself is relatively simple to understand, but the related terminology can be confusing.

- ***_evaluation.txt: Each label (“agree”, “disagree”, “discuss”, and “unrelated”) will contain one evaluation text file regardings three standard evaluation scores: prediction, recall, and F1 score.

6.2] Output analysis:

```
aditya@aditya-X550CC: ~/Downloads/Final
aditya@aditya-X550CC:~/Downloads/Final$ ls
FNC-Project-Stance  pipeline.sh  resources  satire
aditya@aditya-X550CC:~/Downloads/Final$ time ./pipeline.sh
Running Stance Analysis
Reading dataset
Total stances: 49972
Total bodies: 1683
Reading dataset
Total stances: 25413
Total bodies: 904
[Parallel(n_jobs=1)]: Done 200 out of 200 | elapsed:   17.3s finished
[Parallel(n_jobs=1)]: Done 200 out of 200 | elapsed:    0.3s finished
Score for fold 0 was - 0.8103701887717578
[Parallel(n_jobs=1)]: Done 200 out of 200 | elapsed:   16.5s finished
[Parallel(n_jobs=1)]: Done 200 out of 200 | elapsed:    0.2s finished
Score for fold 1 was - 0.8263723983796619
[Parallel(n_jobs=1)]: Done 200 out of 200 | elapsed:   17.6s finished
[Parallel(n_jobs=1)]: Done 200 out of 200 | elapsed:    0.2s finished
Score for fold 2 was - 0.8291654550741495
[Parallel(n_jobs=1)]: Done 200 out of 200 | elapsed:   18.8s finished
[Parallel(n_jobs=1)]: Done 200 out of 200 | elapsed:    0.3s finished
Score for fold 3 was - 0.8460396707596816
[Parallel(n_jobs=1)]: Done 200 out of 200 | elapsed:   19.3s finished
[Parallel(n_jobs=1)]: Done 200 out of 200 | elapsed:    0.2s finished
Score for fold 4 was - 0.8127056803548434
[Parallel(n_jobs=1)]: Done 200 out of 200 | elapsed:   19.4s finished
[Parallel(n_jobs=1)]: Done 200 out of 200 | elapsed:    0.2s finished
Score for fold 5 was - 0.8088365243004418
[Parallel(n_jobs=1)]: Done 200 out of 200 | elapsed:   18.0s finished
[Parallel(n_jobs=1)]: Done 200 out of 200 | elapsed:    0.2s finished
Score for fold 6 was - 0.7888537776501006
[Parallel(n_jobs=1)]: Done 200 out of 200 | elapsed:   19.0s finished
[Parallel(n_jobs=1)]: Done 200 out of 200 | elapsed:    0.2s finished
Score for fold 7 was - 0.8039881204921511
[Parallel(n_jobs=1)]: Done 200 out of 200 | elapsed:   17.4s finished
[Parallel(n_jobs=1)]: Done 200 out of 200 | elapsed:    0.2s finished
Score for fold 8 was - 0.8177777777777778
[Parallel(n_jobs=1)]: Done 200 out of 200 | elapsed:   17.4s finished
[Parallel(n_jobs=1)]: Done 200 out of 200 | elapsed:    0.2s finished
```

```
Score for fold 9 was - 0.7931900641920179
[Parallel(n_jobs=1)]: Done 200 out of 200 | elapsed:    0.5s finished
Scores on the dev set
-------------------------------------------------------------
|           |   agree   | disagree  |  discuss  | unrelated |
-------------------------------------------------------------
|   agree   |    178    |     1     |    504    |    79     |
-------------------------------------------------------------
| disagree  |    26     |     4     |    115    |    17     |
-------------------------------------------------------------
|  discuss  |    77     |     5     |   1545    |    173    |
-------------------------------------------------------------
| unrelated |     6     |     0     |    75     |   6817    |
-------------------------------------------------------------
Score: 3613.25 out of 4448.5    (81.22400809261549%)


[Parallel(n_jobs=1)]: Done 200 out of 200 | elapsed:    1.1s finished
Scores on the test set
-------------------------------------------------------------
|           |   agree   | disagree  |  discuss  | unrelated |
-------------------------------------------------------------
|   agree   |    217    |     3     |   1395    |    288    |
-------------------------------------------------------------
| disagree  |    36     |     0     |    415    |    246    |
-------------------------------------------------------------
|  discuss  |    241    |     2     |   3496    |    725    |
-------------------------------------------------------------
| unrelated |     8     |     0     |    268    |   18073   |
-------------------------------------------------------------
Score: 8754.25 out of 11651.25  (75.13571505203305%)
Running Satire Analysis
load_training_data: Loaded 2638 documents
enhance_docs: Applying 2-boost enhancement to 2638 docs
train_test_split: Splitting data into 2110 training, 528 testing docs
build_model: Model trained
```

```
             precision    recall  f1-score   support

      False       0.88      0.47      0.61        30
       True       0.97      1.00      0.98       498

avg / total       0.96      0.97      0.96       528


real    12m3.559s
user    15m27.152s
sys     12m35.392s
aditya@aditya-X550CC:~/Downloads/Final$
```

6.2.1] Satire Detection:

Dataset: https://people.eng.unimelb.edu.au/tbaldwin/resources/satire/

Here we summarise some example results using this code. In each case, we report the F1 score for the 'satire' class. Baseline classifier (scikit-learn's 'dummy / stratified' approach): **F1 = 0.04**

SVM classifier (linear kernel, C=10) **F1 = 0.64**

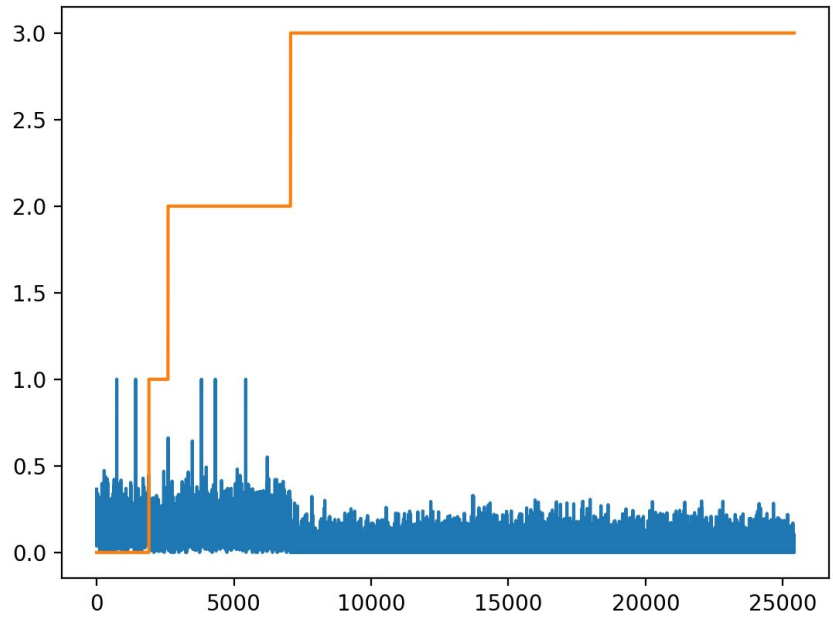
SVM classifier (enhanced entities) **F1 = 0.67**

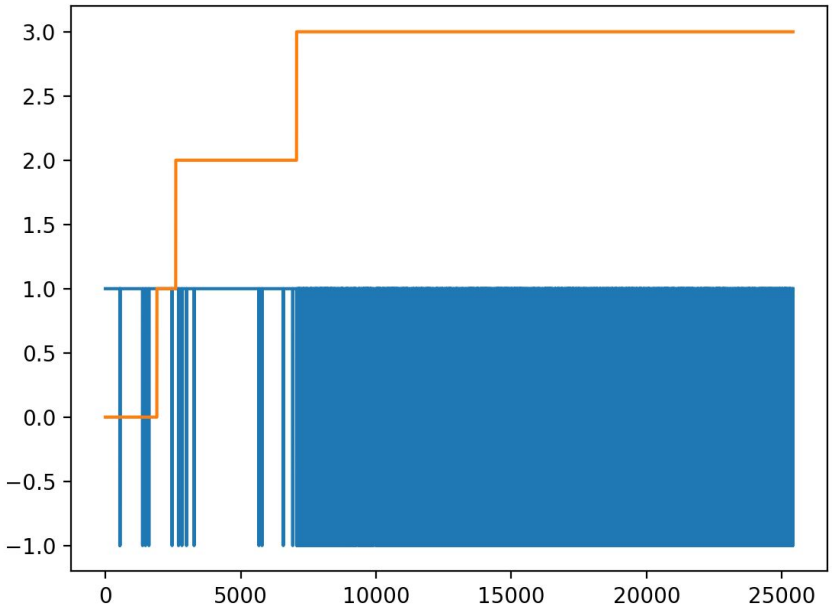
For all the SVM results observed, the precision was close to 1 with a recall of around 0.5. This is similar to the initial results in the Burfood & Baldwin (2009) paper. The baseline 'dummy' classifier here scores very badly, as it depends on the class frequencies and very few documents are labelled as satire.

6.2.2] Stance Detection:

Our stance detection is based on the baseline implementation of Fake News Challenge. The dataset contains two sets of information: headline and body text. The classifier is aimed to have the capability of finding the relationship between a headline and a body text among "agree", "disagree", "discuss", and "unrelated". But be that as it may, we found it very much low confident to classify among "agree", "disagree", and "discuss" without considering natural language understanding and natural language disambiguation; and they both take a very while to experiment and engineer features.

By analyzing the tf-idf feature as the first chart shown underneath, the label 3 (unrelated) has an obvious advantage to differentiate itself from others. Inside those three, we basically could not depend on the TF-IDF to extract further more information. The SVD feature has a comparative effect on model as the second chart shown underneath.





We decided to seek for another approach. It is not hard to use a tiny trick to turn this multi-classifier into binary classifier. We combine “agree”, “disagree”, and “discuss” into one class called “related”. From the viewpoint of feature engineering, we did not essentially alter our goal, which is to figure out the relatedness of a headline and a text body, in spite of the fact that our accuracy might be lower. We aim to build a decision tree with our another model to improve the overall accuracy.

6.3] Comparison of Output against hypothesis:

We had stated in the hypothesis that we would be utilizing Bigram TF-IDF and SVM with Random Forests using Scikit. We have used these features. The output expectation has been ±5% with abnormal case explanation provided.
While implementing syntactical dependencies, we faced some issues which parsing. This feature has been included in the code but may have some effect on specific news articles.

6.4] Abnormal case explanation:

When we tried to include latest news articles which could be categorized as not completely true and satirical, the success factor was more or less similar. But then we tried to run it with some *The Onion* and *The Garlic* news sources, some abnormalities were observed in that the confidence factor fell down drastically.
Explanation: We got to know that there is a Fake-News generator based on Machine Learning algorithms which is used by the above mentioned websites for their news generation.
This generator is specifically designed to avoid the keywords/features applied to detect fake news. Hence the abnormal case and it’s reason.

7] CONCLUSION AND RECOMMENDATIONS:

7.1] Summary and Conclusions:

Detecting satire is part of a wider goal of reducing misinformation and disinformation in a collection of news articles. A perfect satire detector could be used to reduce the risk of some types of misinformation, but is not enough in itself. To an end-user, different types of mistake may be more or less problematic. Put simply, is it a worse mistake to label a news article as satire, or vice versa? Presenting obviously humorously false information as if it were genuine is likely to undermine user’s faith in the system, whereas perhaps falsely labelling a few genuine

stories as suspect or fake is less critical, as other sources of the same stores are likely to appear and (hopefully) be labelled as genuine.

Initial results show that term frequency is potentially predictive of fake news; an important first step toward using machine classification for identification. However, we remain concerned about overfitting and learning topical patterns that predict the partisan split of the legitimate vs. fake sources as identified by OpenSources.co.

As for stance detection, a good solution would allow a human fact checker to enter a claim or headline and instantly retrieve the top articles that agree, disagree or discuss the claim/headline in question. They could then look at the arguments for and against the claim, and use their human judgment and reasoning skills to assess the validity of the claim in question. Such a tool would enable human fact checkers to be fast and effective. In this way, the various stances (or lack of a stance) news organizations take on a claim, as determined by an automatic stance detection system, could be combined to tentatively label the claim as True or False. While crude, this type of fully-automated approach to truth labeling could serve as a starting point for human fact checkers, e.g. to prioritize which claims are worth further investigation.

The results show that it is much more easier to identify relatedness rather than actually matching “agree”, “disagree”, and “discuss” by our research progress. However, being able to figure out relatedness contributes a major part to classify fake news.

Ultimately an objective way of classifying ‘fake’ from legitimate news continues to be barrier that will make adoption difficult. Whereas fake reviews of restaurants might follow different syntactic structures to true reviews without intent, fake news is intended to mislead. In this case, it is likely that the unreliable sources will do their best to mimic the syntactical qualities of legitimate news sources.

7.2] Recommendations for future studies:

Our very first obstacle was unexpected. We thought fake news detection was to identify if the news is truthful or not. The task is obviously not so simple. Beginning from the definition, we very rapidly discovered that there are many different categories misinformation can fall into. There are articles that are blatantly false, articles that provide a truthful event but then make some false interpretations, articles that are pseudoscientific, articles that are really just opinion pieces disguised as news, articles that are satirical, and articles that are comprised of mostly tweets and quotes from other people. Without a question, detecting fake news is much more harder than we could envision. Here in our project, we have uncovered some advantages and

workable solutions by utilizing satire and stance detection. Except for these two, one potential improvement is to discover more perspectives. For instances, we could analyze the grammatical errors. Perhaps, we can seek for an effective manner to combine them into one high accuracy model. Secondly, maybe simplifying the problem would be the key to a higher degree of accuracy. So we really thought about what the problem was we trying to solve. Maybe the answer isn't detecting fake news, but detecting real news? Real news is much easier to classify. Its factual and to the point, and has little to no interpretation. And there were plenty of reputable sources to get it from.

8] BIBLIOGRAPHY AND OTHER REFERENCES

- Victoria L. Rubin, Niall J. Conroy, Yimin Chen, and Sarah Cornwell, "Fake News or Truth? Using Satirical Cues to Detect Potentially Misleading News", Proceedings of the *2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* , San Diego, CA, 2016, pp. 7-17.
- Fan Yang , Arjun Mukherjee and Eduard Gragut, "Satirical News Detection and Analysis using Attention Mechanism and Linguistic Features," *2017* **arXiv:1709.01189v1** [cs.CL] ,2017.
- Victoria L. Rubin, Tatiana Vashcilko, "Identification of truth and deception in text: application of vector space model to rhetorical structure theory"
- Qi Su , Chu-Ren Huang , Helen Kai-yun Chen, Evidentiality for text trustworthiness detection, Proceedings of the 2010 Workshop on NLP and Linguistics: Finding the Common Ground, p.10-17, July 16-16, 2010, Uppsala, Sweden
- Eugene Agichtein , Carlos Castillo , Debora Donato , Aristides Gionis , Gilad Mishne, Finding high-quality content in social media, Proceedings of the 2008 International Conference on Web Search and Data Mining, February 11-12, 2008, Palo Alto, California, USA
- https://github.com/FakeNewsChallenge
- https://github.com/dcorney/satire
- https://github.com/aldengolab/fake-news-detection
- https://github.com/violation-feedback/ml-text-generator

9] APPENDICES

Program Flowchart

Pg. 14

Program source code with documentation

Pg. 13

Input/Output listing

Pg. 18