Fake reviews detection and analysis

Parth Shroff*
Subhash Holla H S*
parths11@vt.edu
subbu23@vt.edu
Virginia Polytechnic Institute and State University
Blacksburg, Virginia

ABSTRACT

Reviews are important part of customer decision to opt for a product or service. This has created a huge industry for fake reviews. This project gives a comparative account of the performance of different models for the classification of fake reviews. We also create and analyze the impact of different features that are used for this purpose. Finally we evaluate an ensemble neural network called StackNet for the purpose of fake review detection reporting the best models when all of these are compared against each other on the Yelp real world dataset.

KEYWORDS

fake reviews, Yelp, feature engineering

ACM Reference Format:
Parth Shroff and Subhash Holla H S. 2018. Fake reviews detection and analysis. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Reviews have now become a vital part of product and/or service preference for customers. This has lead to an increasing demand for fake reviews. Fake reviews have a huge market which is used to improve the visibility and likability of a specific business or attack that of a competitor [8]. Customers believe the reviews to be genuine as long as they read or appear to be so. Hence, it will be the responsibility of the platform hosting this review functionality to filter reviews and remove fake ones. Fake reviews also have an impact on the platforms reputation and success as fake reviews reduce the reliability of the platform as a source of information.

In the arms race of securing the review platforms review flagging and filtering systems have been kept as a black box to a large extent, but research has shown that infiltrating these black-box systems have been a lucrative business that has not diminished in recent years as the cost of employing someone to provide fake reviews is not high [11]. This reduced cost questions the effectiveness the existing systems.

*Both authors contributed equally to this research.

Conference'17, July 2017, Washington, DC, USA
© 2018 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/1122445.1122456

This possible ineffectiveness could be partly because of the fact that there is no defined "fake review" or ground truth that these flagging or filtering system will have. The lack of labels reduce the capabilities of the systems as even if they employ unsupervised learning efforts the attacker will only have to remain under the radar to prevent the detection of the fake reviews. This paired with the fact that attackers could have access to botnets which could help perpetuate these attacks with just brute force raise a serious and immediate concern for the review platforms.

2 LITERATURE REVIEW

The research community has provided quite a few defence mechanisms to detect fake reviews[13]. The trend we see in the frameworks that have been used is to use the existing or available features from the raw review, reviewer and product/service data to first generate new higher level, meaningful features and then use this amalgamated dataset with existing and created features to classify the reviews as fake. The features generated generally fall in one of two categories, review based features and network based features. In review based features we see text features to be the most frequently used tool while feedback on review through ratings, tags, likes,etc. In network based features research has used two different levels of network features with a nested use of each level. The features could provide information on relationship between reviewer and review, reviewer and product/service, review and product/service, reviewers and finally products.

In [19] we see a system called SpEagle which uses a unified framework consisting of metadata (text,time,rating) and relational data(network) to generate features which are used to spot a suspicious user and review. SpEagle employs a review-network-based classification task which accepts prior knowledge on the class distribution of the nodes, estimated from metadata. It works in an unsupervised fashion, but can easily leverage labels (if available). A light version of SpEagle called SpLite is also introduced which uses a very small set of review features as prior information, providing significant speed-up. Finally they evaluate their work on real world data.

[12] analyses the different features generated with the available data for the Yelp platform. This paper, by considering the effectiveness of supervised solutions, discusses and analyzes on a general level the most appropriate review- and reviewer-centric features and then proposes some new features.

Research has also used multi-feature-based approaches, which employ several features of different nature in addition to simple linguistic ones, either by applying supervised or semi-supervised machine learning [14, 15, 18], or by implementing the Multi-Criteria

Decision Making (MCDM) paradigm [21]. These approaches usually focus on small labeled datasets for evaluation purposes, constituted in most of the cases by 'near ground truth' data [13]. They usually avoid to consider features that are extracted from the social ties constituting the network of entities (e.g., users, products, reviews) considered by the review site. On the contrary, this kind of features is often utilized (together with the other features previously described) by graph-based approaches [10, 22]. These latter approaches are in most of the cases unsupervised, even if sometimes they can be coupled with a supervised learning phase on a limited number of classification labels [19]. With respect to supervised approaches, totally unsupervised solutions generally provide slightly worst results [11, 13, 20].

In [18] the authors present an interpretation of what the Yelp Filtering system could be doing. In this work they assume the "flagging" of a review to be their ground truth as fake and try to mimic the same system. They achieve a F1 score of 71% with the various text based features that they create. They also get similar precision and recall scores. Though this can only claim to be an estimate of the system performance at the time the output showed that the researchers were able to get a good estimate of the supposedly black-box system that Yelp maintains with access to the public data that Yelp publishes. Our work tries to reproduce the results of this paper trying to provide more insight on the fake review detection and the contribution of different features to help with the process.

In this project we will asses the use of existing and generated features using Yelp Dataset to detect fake reviews. We have also attempted develop acumen on the features that have been used in past similar research analyzing their significance. The rest of the report is divided into the following sections: We first present the threat model of the research problem followed by our research questions. We then present the methodology adopted and the results we achieved. The report concludes with a discussion of our results, a limitations, conclusion and a future work section.

3 THREAT MODEL

The threat of a fake review and the consequences of letting a fake review persist on a platform can be best captured by a threat model. Here the goal is to succinctly capture the overall threat and to set goals for the defenses that will be built to help mitigate said threat.

1. **Things of value** [16]:
   - *Reputation of businesses and service providers.* Most of the fake reviews posted online will be intended to hurt or boost the reputation of a business or a service provider.
   - *Revenue.* Since the revenue stream of all the businesses tied to an online review platform is dependent on the reputation the business or service will suffer huge losses when the reputation is hit. This direct relation also means that a boost in the reputation using fake reviews can skyrocket the income.
2. **Attack surfaces**:
   - *Reviews.*
   - *Ratings.*
3. **Attackers**:
   - *Automated reviews/Botnets.* With the reduction in cost of computation and improvement in the review generators attackers can now create reviews that appear to be very real. This helps them fly under the radar and also ensure that the fake accounts or bots behaviour is never anomalous.
   - *Crowd sourced reviews.* For attackers that wish to have more realistic reviews from profiles that are maintained by real humans there are services online where crowd sourcing is used to post fake reviews on businesses or services.
4. **Mitigation**:
   - *Fake review detection and filtering.* The goal of every defense is to first detect a fake review. Once detection is successful the objective is to eliminate such reviews. Ideally a defender will want to minimize the time between the review posting and the filtering. This is to ensure that a fake review will have minimum impact on innocent users who might believe them to be true.
5. **Costs and risks**:
   - *False positives.* The risk that is very taxing on any review detection and filtering system are false positives as we do not wish to filter a real review thinking it is fake. This can have a negative impact on platforms credibility.

4 RESEARCH QUESTIONS

- **RQ1**: *What models operate the best in classifying a review as fake?*
- **RQ2**: *What features contribute the most in classifying a review as fake?*
- **RQ3**: *Do boosting and ensemble methods help improve overall classification accuracy?*

5 DATA

The data that was used for this project was provided by Dr. Bing Liu of University of Illinois, Chicago[5]. The data was previously used in [18]. The data comprised of two databases. The first was a database with hotel information. The second was a database with Restaurant information. Each table had 3 different tables. The first provided review based information. The second had reviewer based information and the final had the service provider (restaurant or hotel) information. Each of the different tables had a ID which can be used in a relational setting like restaurant/hotelID, reviewerID or reviewID.

For this project we restricted ourselves to the Restaurant database. Table 1 has a description of the different types of features that we encountered in the data that was considered. The identifiers included IDs and names. We had two different types of categorical data, first was the binary class which had a 'yes' or 'no' while the second was the multi-class label. The data had one label called "flagged" which had four possible values "N","NR","Y" and "YR". The "Y" and "N" values indicated flagged and not flagged as pulled from a restaurants site. The "YR" and "NR" were the same as pulled from a reviewers profile. We restricted our data to "Y" and "N" values as this was the data as these were used in [18] and any results from our study would have a comparable baseline. This gave us 67016 reviews with 8303 flagged or "Y" reviews and 58716 not flagged or "N" reviews.

6 METHODOLOGY

We used two different systems for the purpose of implementing the preprocessing and models. The first was a local machine with a i7-7700 Intel processor, 16 GB RAM and GTX 1050 Ti graphics and the second was the VM GPU cluster provided for the course. We ran all the preprocessing and feature engineering on the local machine. The models were run on the cluster.

6.1 Data Preprocessing

The purpose of the data preprocessing was to clean the raw data and merge the different data tables to get a single data frame to be used for learning. To this end we used the following steps:

6.1.1 Handling missing values: The missing values by percentage in the data can be seen in fig.1. We replaced all the missing values from the restaurant data frame with a new class called "None". This was to ensure that while we do not loose any information by just eliminating all the rows with the missing values we did not want to misinterpret the data by using methods like mean value or mode value replacement.

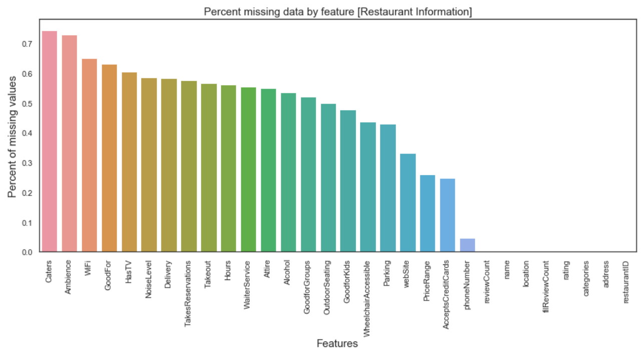

Figure 1: Percent missing data by feature [Restaurant Information]

6.1.2 Extracting location information: We decided to capture all of the location data in the city level extracting and encoding the same from the geographical features from each of the data tables.

6.1.3 Time feature: The review table had a time feature which was the date that the specific review was posted. We normalized this to a standard date, viz., the 1st of July 2004 (the start date of Yelp according to Google). This would give a number that would capture the date as a feature. We also used a similar technique to capture the age of the reviewer account in the months that the account was active which was a provided feature.

6.1.4 Encoding features: The remaining features were encoded in one of two ways: Label encoding or Multi-Label Binarizer. The label encoder was used for features that were binary categorical features and ones that had only a single category assignment per row/review. For rows that had multiple values in a single row/review we used the Multi-Label Binarizer. For example, the "GoodFor" feature column could have values like "Lunch", "Dinner",etc. with multiple values in each row separated by a comma(,). The multi-label binarizer when used in conjunction with a custom function that separates the entries to a list will give a one hot encoding that will capture this information.

6.1.5 Removing columns: We eliminated the following columns as we felt they did not add any useful information to help with the classification or were providing redundant information: 'address', 'webSite', 'phoneNumber', 'Hours', 'name'. The 'categories' was also eliminated as this feature had rougly 250 unique entries which blew up our data. As we were unable to come up with an effective method to capture this information in an effective manner we decided to eliminate this feature initially. We have proposed a possible solution in our future work.

6.2 Model Building

Once the data preprocessing was completed and we merged the data to get one data frame ready for learning, we scaled all of the numerical features. This gave us a "X" data frame with all the features and "Y" with the respective labels. We used 6 different models to train on this classification problem. Each of these models were tuned for hyperparameter only intuitively with no employment of grid search or other systematic tuning methods. We set the random state constantly to be **0** whenever needed. We also set the jobs to **-1** when parallelization was possible. This ensured that the models used as many cores as they found vacant.

Each of these models were run on a 5 fold cross validation with Strified KFold sampling for initial evaluation. We used cross-validation to

6.2.1 Logistic Regression: We used this linear model to be the simplest model of the ones that we chose.

6.2.2 Stochastic Gradient Descent (SGD):. This classifier uses gradient descent to help classify the data to a specific label. The ensemble method from Scikit-learn was used for this purpose. We set the loss to be log-loss, the penalty to be 'L2-norm' and the maximum iterations or the number of epochs to be 5.

6.2.3 Random Forest: The random forest classifier was used with 500 estimators or trees and a maximum depth of 3. To estimate the generalization accuracy we used out-of-bag samples. We also wanted the learner to add to previous calls rather than learning fresh every single time. Hence we set the warm start parameter to be true.

6.2.4 Multilayer Perceptron (MLP):. We used a simple neural network, viz., Multilayer Percpetron (MLP). Here we used âĂŸlbfgsâĂŹ as our solver/optimizer which is from a family of quasi-Newton methods. This was reported to converge faster and perform better on the sklearn website. We used an L2 regularization of 0.00001 and a learning rate that was inversely scaled, i.e., it decreased with time. The neural network had 2 hidden layers with 5 and 2 neurons respectively.

6.2.5 K-Nearest Neighbour (KNN):. The KNN classifier was used with 5 neighbours.

Table 1: Data features details

Feature type Table name	Identifiers	Geographical	Categorical (Binary)	Categorical (Multi)	Numerical	Label	Total
Restaurant	2	2	12	11	3	0	30
Reviewer	2	1	0	0	10	0	13
Reviews	3	0	0	0	5	1	9
Total	7	3	12	11	18	1	52

6.2.6 AdaBoost: This was a boosting method we used. This had 500 estimators and a learning rate of 0.001.

6.3 Feature Engineering

Once each of these models were run on the preprocessed data we decided to generate text based features that were used in past research. Literature review should that each of these features helped improve model accuracy in multiple instances. We assumed that a culmination of multiple text features will definitely improve our model output. Hence we generated 14 different features. Of these 11 were count based features which involved simple counts of different elements of the review. The remaining 3 higher level features: Topic model (Latent feature), profanity score and polarity and subjectivity scores.

6.3.1 Count based features: In [12] the authors present "text-statistics" as a set of features that help improve review classification. These include, number of words, capital letters, capital words, person pronouns and exclamations in a sentence. In [11, 13] research shows that Part Of Speech (POS) tagging helps with the classification of reviews. Based on these inputs we engineered 11 different features. They are: "review_length", "no_of_stopwords", "char_count", "word_density", "punctuation_count", "upper_case_word_count", "noun_count", "verb_count", "adj_count", "adv_count" and "pron_count". The last 5 are POS tags while the first 6 could be counted as "text-statistics".

Based on the generated count for each of the reviews we added 11 columns. Appendix A has a table with the statistical properties of these features.

6.3.2 Topic model: A topic model is a higher level feature that is extracted from the Latent Dirichlet allocation (LDA) of a review. This will supposedly give a latent abstraction of the topic that the review is talking about. This can supposedly be used to assign a review to one or few topics. Using such a model from the standard gensim[2] library we tried to create a topic model. But the coherence, the metric that evaluates separation between clusters of topics, we got from the inbuilt model was low (0.3). Generally a higher coherence is preferred. This led us to Mallet[17] which is a topic modeller which can be used with a wrapper provided by the gensim library. This model improved the coherence (0.45). The difference in the topics generated by the 2 models can be seen in the fig.2. Finally we evaluate the coherence change with the number of topics considered. Fig.3 gives a description of the change.

We decided to choose 25 topics based on this assessment and used this topic modeller to assign a number from 1 to 25 to each review.

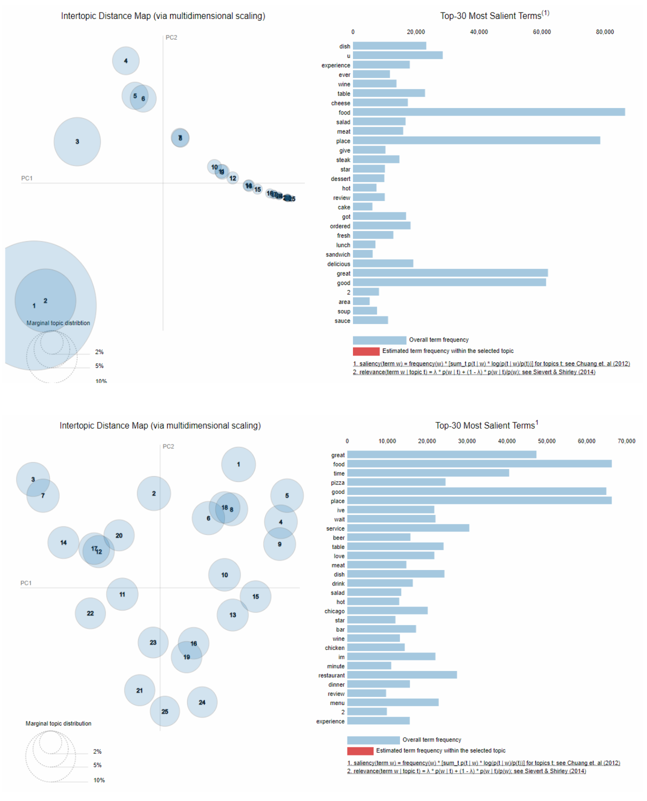

Figure 2: Comparison of topics generated by the 2 models

Table 2: Profanity score basic statistics

Mean	0.025798654
Standard Error	0.000612389
Median	0
Mode	0
Standard Deviation	0.158535354

6.3.3 Profanity Score: We used profanity-check[6] to help assign a profanity score for each review. This is a linear SVM model trained on 200 thousand human-labeled samples of clean and profane text strings. [Trained on t-davidson/hate-speech-and-offensive-language[4], and the Toxic Comment Classification Challenge on Kaggle[9].] This was a feature that we assumed could help with the classification of the reviews. Our hypothesis was that the attackers

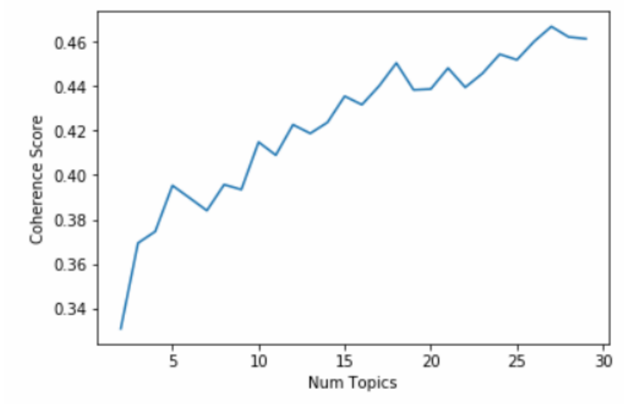

Figure 3: Coherence values for topics ranging between 2 and 30

would most likely refrain from the use of words that are consider to be profane so as to avoid flagging. Profanity-check also performed better and was faster than other similar modules like profanity and profanity-filter. An example that recieved a positive (1) profanity score is shown below:
This place is the shit!!! With 396+ plus reviews need I say more?
-Frankie R.

Table 3: Polarity and Subjectivity scores basic statistics

	Polarity Score	Subjectivity Score
Mean	0.263311828	0.56730241
Standard Error	0.000743392	0.000476495
Median	0.2511	0.5667
Mode	0	0.6
Standard Deviation	0.192449472	0.123355241

6.3.4 Polarity and Subjectivity score: Sentiment analysis was presented as an important feature to help classify reviews in [6]. We used an off the shelf sentiment analyzer from the Python library called textblob [1]. This gave us 2 values.

- **Polarity:** A value in the range of -1 to 1 indicating the negative or positive sentiment of the review with 0 as neutral.
- **Subjectivity:** The subjective nature of the polarity score.

The above example had a polarity score of 0.0547 and a subjectivity of 0.65.

Once these features were engineered we added the same to our data. We then ran our model once again with the amalgamated dataset. We used the same models and hyperparameter values.

6.4 Ensemble Model: StackNet

StackNet[3] is a computational, scalable and analytical framework with multiple stacked levels to improve accuracy in machine learning problems. In contrast to feedforward neural networks, rather than being trained through back propagation, the network is built iteratively one layer at a time (using stacked generalization), each of which uses the final target as its target.Marios Michailidis[7] built this as a part of his PhD thesis and also used this to win a competition in Kaggle. This is an ensemble method that has been used in multiple real world applications since the first implementation in 2015.

For our ensemble instance we used the same 6 initial models used above. We built a 2 layered StackNet neural net with the first having all 6 models while the second having only the Random Forest classifier.

6.5 Metrics

We initially used "balanced accuracy" to be one of the metrics for evaluation. We were achieving very bad accuracies. After a brief literature review we realised that when the class imbalance favours the negative class over the positive class, i.e., more negative examples than positive examples, the F1 score is preferred as the balanced accuracy metric will fail.

6.5.1 Fit and Score Time: We chose to report these times as the ideal end goal of these model evaluations to have a filtering system that can flag fake reviews in real time. Hence we would want to pick a model keeping in mind that the time required to score the reviews need to be low. This would also depend on the computational resources available to the platform that will be filtering the review.

6.5.2 Precision and Recall: We computed the weighted precision and recall of the cross-validated models. We also have computed the Precision-Recall plots of a 60-40 data split for each data once the cross validation was completed and reported the same.

6.5.3 F1 Score: We think that the F1 score will be the true measure of our models. We have reported the cross validation F1 score along with the F1 score for the models with a 60-40 train-test split.

7 RESULTS AND DISCUSSION

7.1 Base models [Without text based features]

The results reported in [18] use the precision, recall and F1 scores on the fake/positive class. We find that the precision we are able to achieve are higher than most models presented in the paper. We argue that the precision takes a higher importance than recall as recall is dependent on the false negatives while the precision is dependent on the false positives which are a higher risk as mentioned in our threat model. Hence, we claim that our primitive model without any added features still provides comparable if not better performance than the model presented in [18].

7.2 Base models + Engineered features

We see from the results that the models performed comparably if not better without any text features. This implies that the text features did not contribute a lot to the classification of the reviews. This rises two questions:

- Was the method or pipeline followed to extract the features right?
- Did the features actually contribute to improvement in system performance in the previous work?

Table 4: Results (normal models)

Data Input	Classifier	Class	Precision	Recall	F1 Score	Support
Base models [Without generated features]	Logistic Regression	0	0.94	0.96	0.95	23515
		1	0.66	0.53	0.59	3292
		Weighted avg	0.90	0.91	0.90	26807
	Stochastic Gradient Descent (SGD)	0	0.88	0.99	0.93	23515
		1	0.45	0.03	0.06	3292
		Weighted avg	0.83	0.88	0.83	26807
	Random Forest	0	0.93	0.98	0.96	23515
		1	**0.77**	0.50	0.61	3292
		Weighted avg	**0.91**	**0.92**	**0.91**	26807
	Multi-Layer Perceptron (MLP)	0	0.88	1.00	0.93	23515
		1	0.00	0.00	0.00	3292
		Weighted avg	0.77	0.88	0.82	26807
	K-Nearest Neighbour (KNN)	0	0.91	0.95	0.93	23515
		1	0.46	0.30	0.36	3292
		Weighted avg	0.85	0.87	0.86	26807
	AdaBoost	0	0.94	0.96	0.95	23515
		1	0.68	**0.57**	**0.62**	3292
		Weighted avg	**0.91**	0.91	**0.91**	26807
Base models + Generated features	Logistic Regression	0	0.93	0.96	0.95	23515
		1	0.67	0.51	0.58	3292
		Weighted avg	0.90	0.91	0.90	26807
	Stochastic Gradient Descent (SGD)	0	0.94	0.91	0.92	23515
		1	0.45	0.56	0.50	3292
		Weighted avg	0.88	0.86	0.87	26807
	Random Forest	0	0.93	0.98	0.96	23515
		1	**0.77**	0.50	0.60	3292
		Weighted avg	**0.91**	**0.92**	**0.91**	26807
	Multi-Layer Perceptron (MLP)	0	0.95	0.66	0.78	23515
		1	0.23	**0.75**	0.36	3292
		Weighted avg	0.86	0.67	0.72	26807
	K-Nearest Neighbour (KNN)	0	0.91	0.95	0.93	23515
		1	0.46	0.30	0.36	3292
		Weighted avg	0.85	0.87	0.86	26807
	AdaBoost	0	0.94	0.96	0.95	23515
		1	0.68	0.57	**0.62**	3292
		Weighted avg	**0.91**	0.91	**0.91**	26807

Before we discuss the implications we would like to point out that a very important feature that we have not used are the Bag of Words(BOW) and LIWC scores. Even though there is a possibility that the pipeline or the procedure that we have adopted could have a flaw that we have overlooked, assuming that there isn't a major flaw because of the multiple sanity checks we have run, we can say that the features could have possibly contributed negligibly in previous work as well. Most of the work that is presented in the literature review give logical explanations for the introduction of these features and immediately show the overall improvement in the model classification of the model output. Our result could imply that the models from previous work improved only because of the BOW and LIWC based features with no contribution from any of the features that we have generated.

We show the different precision-recall graphs of the top 3 models. For the precision-recall graph we used a 60-40 split of the data for which we also computed the confusion matrix. We see that the average precision of random forest is 0.99 which is highly desirable.

Another takeaway, assuming that [18] evaluated other models but reported the best one, could be that the basic models like Random Forest, Adaboost, etc. have improved implementation as compared to the time when these publications were made hence leads to improved results.

A reason that the text features don't really improve the classifiers could be that many of these fake reviews must have been written by actual humans and not machine generated. As these could have been written by humans, there couldn't be any structural differences between them.

We would like to remind that the models were not tuned for the ideal hyperparameters and we could see an improvement in models performance just by tuning these parameters.

Table 5: Results (normal models)

Data Input	Classifier	Class	Precision	Recall	F1 Score	Support
Base model [Without generated features]	**StackNet**	**0**	0.94	0.98	0.96	23515
		1	0.79	0.55	0.65	3292
		Weighted avg	0.92	0.93	0.92	26807
Base model + Generated features	**StackNet**	**0**	0.94	0.98	0.96	23515
		1	0.78	0.55	0.64	3292
		Weighted avg	0.92	0.93	0.92	26807

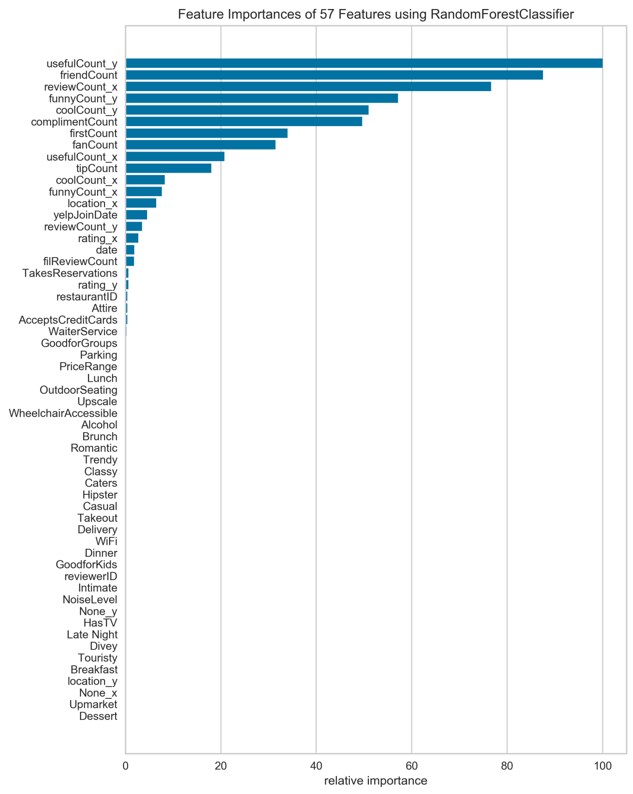

Figure 4: Contribution of features to Classification (Without Text Features)

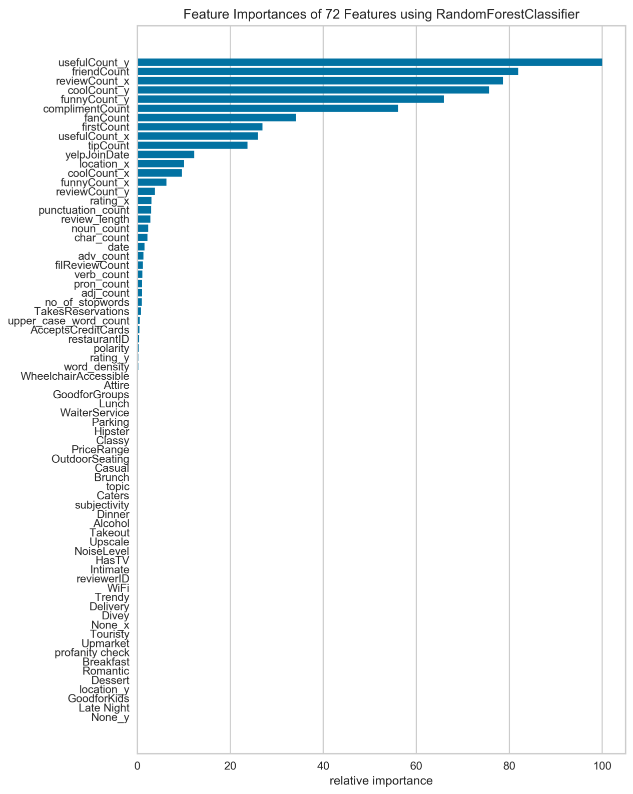

Figure 5: Contribution of features to Classification (With Text Features)

8 LIMITATIONS

Our project has its fair share of limitations. We have tried to list the ones we noticed:

- We did not have access to the proprietary LIWC feature generator and hence did not evaluate this important feature.
- We were unable to tune the hyperparameters of the models that were evaluated and hence cannot definitively claim an improvement in the model accuracies after tuning them.
- We were unable to test our model with current data, for example, from the Yelp Dataset Challenge 2019.
- We have the assumption that the off the shelf models and libraries that we have used are 100% right. We have not imposed any conditions on their accuracy.
- We were able to achieve a good result from the models that we have but have not be able to clearly attribute the difference in the result from the paper to a specific element of our model.

9 CONCLUSIONS

In conclusion, we have given an evaluation of different models that can be used to classify reviews on online platforms. We have also given an analysis of the different features that can be generated using feature engineering showing that these features that are generated have not contributed to improve system performance. We would like to raise a concern that existing research has not provided concrete evidence on the contribution of different features on the improvement of model output and taking this in face value

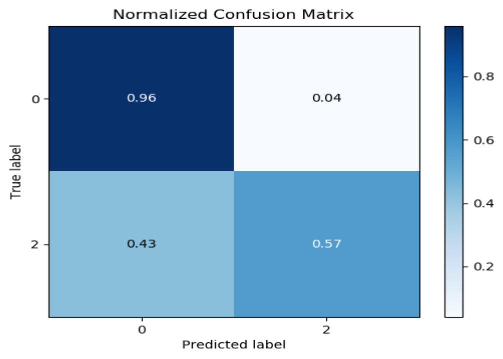

Figure 6: Without Text Features

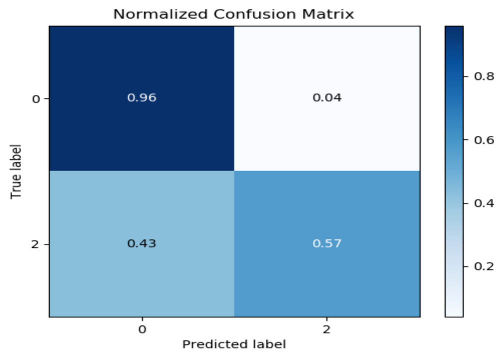

Figure 7: With Text Features

Figure 8: ADABoost Confusion Matrices

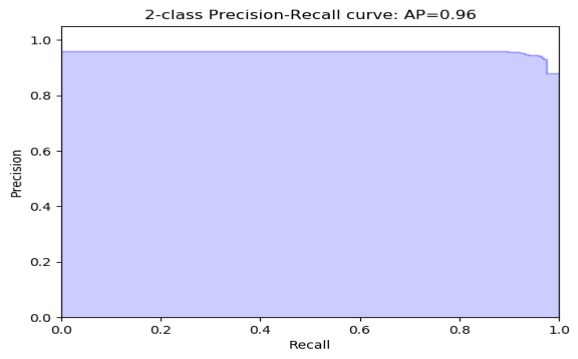

Figure 9: Without Text Features

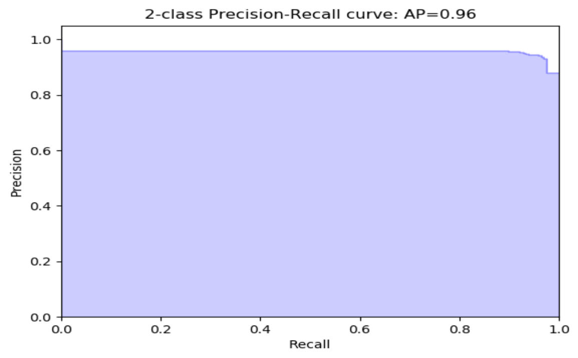

Figure 10: With Text Features

Figure 11: ADABoost Precision-Recall Curves

could lead to unimpressive results as was in our case. We have also been able to show a good precision in our models output, especially with our ensemble neural network.

Future work should focus on properly evaluating the contribution of a feature generated for a model and also the performance of different models for the classification task before deciding on the better performer. What could be done as a part of future work is studying the similarity between the reviews of the same user as well as between users. However, we believe that this would not be that useful as even a legitimate user could be using a fixed template for their reviews and also it is highly possible that a legitimate user could copy the style of the review of some other user. We do acknowledge, omitting the use of bag of words to actually represent the content of the review as a vector but it is possible that that too might not be of much help because of the above reasons. Another direction could also involve the usage of deep neural networks to conduct end to end machine learning with no feature generation which could eliminate the need for extraction of features.

REFERENCES

[1] [n. d.]. API Reference âĂŤ TextBlob 0.15.2 documentation. https://textblob.readthedocs.io/en/dev/api_reference.html#textblob.blob.TextBlob.sentiment. (Accessed on 05/08/2019).
[2] [n. d.]. gensim: Topic modelling for humans. https://radimrehurek.com/gensim/index.html. (Accessed on 05/08/2019).
[3] [n. d.]. h2oai/pystacknet. https://github.com/h2oai/pystacknet. (Accessed on 05/08/2019).
[4] [n. d.]. hate-speech-and-offensive-language/data at master Âů t-davidson/hate-speech-and-offensive-language. https://github.com/t-davidson/hate-speech-and-offensive-language/tree/master/data. (Accessed on 05/08/2019).
[5] [n. d.]. liu.cs.uic.edu/download/yelp_filter/. http://liu.cs.uic.edu/download/yelp_filter/. (Accessed on 05/08/2019).
[6] [n. d.]. profanity-check Âů PyPI. https://pypi.org/project/profanity-check/. (Accessed on 05/08/2019).
[7] [n. d.]. Stacking Made Easy: An Introduction to StackNet by Competitions Grandmaster Marios Michailidis (KazAnova) | No Free Hunch. shorturl.at/bhjkU. (Accessed on 05/08/2019).
[8] [n. d.]. Study finds 61 percent of electronics reviews on Amazon are 'fake' - Marketing Land. https://marketingland.com/study-finds-61-percent-of-electronics-reviews-on-amazon-are-fake-254055. (Accessed on 05/05/2019).
[9] [n. d.]. Toxic Comment Classification Challenge | Kaggle. https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/data. (Accessed on 05/08/2019).
[10] Leman Akoglu, Rishi Chandy, and Christos Faloutsos. 2013. Opinion fraud detection in online reviews by network effects. In *Seventh international AAAI conference on weblogs and social media.*
[11] Michael Crawford, Taghi M Khoshgoftaar, Joseph D Prusa, Aaron N Richter, and Hamzah Al Najada. 2015. Survey of review spam detection using machine learning techniques. *Journal of Big Data* 2, 1 (2015), 23.
[12] Julien Fontanarava, Gabriella Pasi, and Marco Viviani. 2017. Feature Analysis for Fake Review Detection through Supervised Classification. In *2017 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*. IEEE,

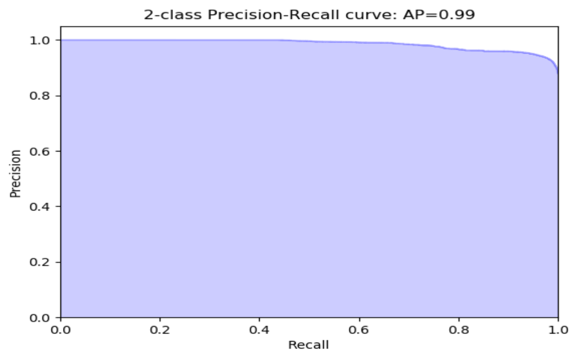

Figure 15: Without Text Features

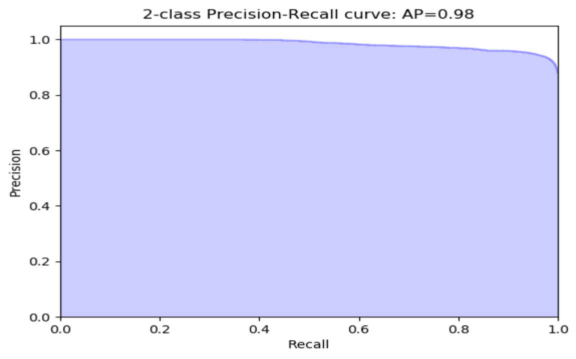

Figure 16: With Text Features

Figure 17: Random Forest Precision-Recall Curves

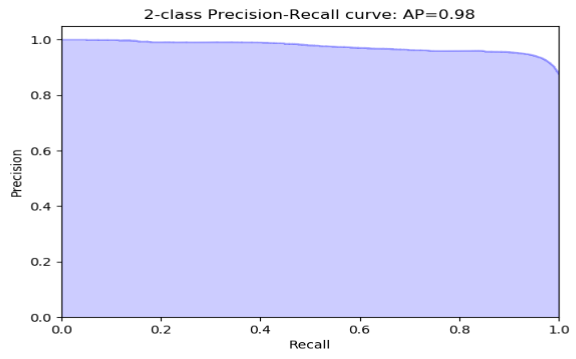

Figure 21: Without Text Features

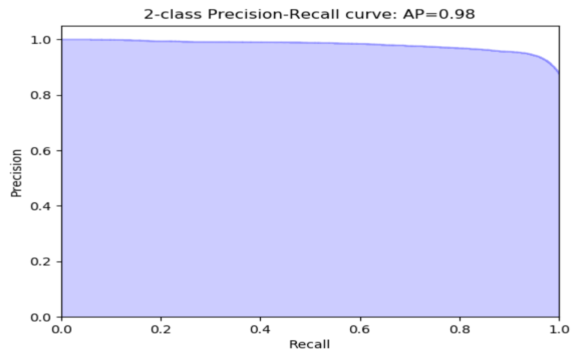

Figure 22: With Text Features

Figure 23: Logistic Regression Precision-Recall Curves

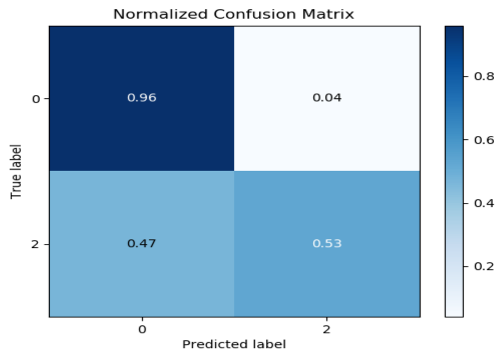

Figure 18: Without Text Features

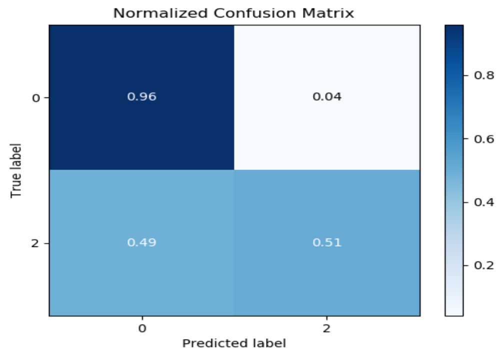

Figure 19: With Text Features

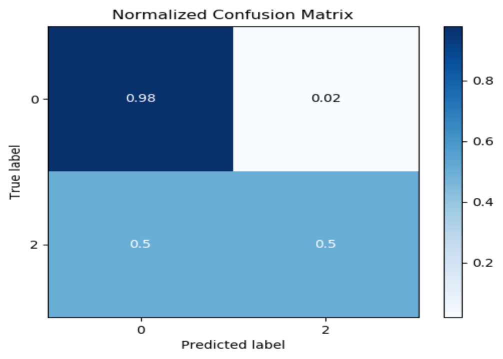

Figure 12: Without Text Features

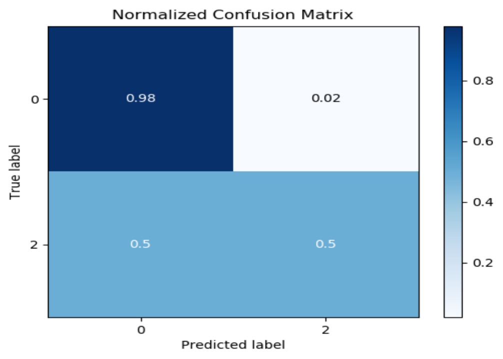

Figure 13: With Text Features

658–666.
[13] Atefeh Heydari, Mohammad ali Tavakoli, Naomie Salim, and Zahra Heydari. 2015. Detection of review spam: A survey. *Expert Systems with Applications* 42, 7 (2015), 3634–3642.
[14] Nitin Jindal and Bing Liu. 2008. Opinion spam and analysis. In *Proceedings of the 2008 international conference on web search and data mining*. ACM, 219–230.
[15] Fangtao Huang Li, Minlie Huang, Yi Yang, and Xiaoyan Zhu. 2011. Learning to identify review spam. In *Twenty-Second International Joint Conference on Artificial Intelligence*.
[16] Michael Luca and Georgios Zervas. 2016. Fake it till you make it: Reputation, competition, and Yelp review fraud. *Management Science* 62, 12 (2016), 3412–3427.
[17] Andrew Kachites McCallum. 2002. MALLET: A Machine Learning for Language Toolkit. (2002). http://mallet.cs.umass.edu.
[18] Arjun Mukherjee, Vivek Venkataraman, Bing Liu, and Natalie Glance. 2013. What yelp fake review filter might be doing?. In *Seventh international AAAI conference on weblogs and social media*.
[19] Shebuti Rayana and Leman Akoglu. 2015. Collective opinion spam detection: Bridging review networks and metadata. In *Proceedings of the 21th acm sigkdd international conference on knowledge discovery and data mining*. ACM, 985–994.
[20] Marco Viviani and Gabriella Pasi. 2017. Credibility in social media: opinions, news, and health informationâĂŤa survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 7, 5 (2017), e1209.
[21] Marco Viviani and Gabriella Pasi. 2017. Quantifier guided aggregation for the veracity assessment of online reviews. *International Journal of Intelligent Systems* 32, 5 (2017), 481–501.
[22] Guan Wang, Sihong Xie, Bing Liu, and S Yu Philip. 2011. Review graph based online store review spammer detection. In *2011 IEEE 11th International Conference on Data Mining*. IEEE, 1242–1247.