SEPARATING VALID FORM-LETTERS FROM FRAUDULENT RESPONSES IN PUBLIC COMMENTS

Steve Landherr
Department of Computer Sciences
University of Wisconsin - Madison
landherr@cs.wisc.edu

William J. Yamada
Department of Biological Systems Engineering
University of Wisconsin - Madison
wyamada@wisc.edu

Rafael E. P. Ferreira
Department of Dairy Science
University of Wisconsin - Madison
referreira@wisc.edu

Xiang Li
Department of Electrical and computer Engineering
University of Wisconsin - Madison
xli2238@wisc.edu

May 8, 2020

ABSTRACT

Detecting fake public comment submissions to governmental proceedings, though similar to detecting fake product reviews, differs in a significant way: duplicate comments, also known as form-letter comments, are perfectly valid. They serve as a vote either for or against when measuring public support for a docket item. This project attempts to identify machine learning features and models that can assist in separating fake duplicate comments from legitimate form-letter comments, and contribute to a more accurate measure of public support for proposed government actions.

***K*eywords** FCC, 17-108, Net Neutrality, Fake Comments, Meta-Data, Machine Learning, Logistic Regression, LR, Artificial Neural Networks, ANN, Support Vector Machine, SVM, Principal Component Analysis, PCA

1 Introduction

In April of 2017, the United States Federal Communications Commission (FCC) opened docket 17-108, Restoring Internet Freedom to public comment [4]. By the time the comment period ended on November 3, 2017, the proceedings included 22,158,109 comment filings [4]. Jeff Kao analyzed these comments using semantic clustering and determined that less than 800,000 comments were truly unique [12]. The remaining comments formed over 300 campaign clusters where they were either word-for-word identical, or semantically identical. Follow-on analyses have focused on identifying fake comments and determining the source. For example, two have have determined that:

1. A large number of the owners of email addresses associated with campaign comments deny making the comments [12, 1].
2. The e-mail addresses used in one campaign significantly overlapped those found in the Modern Business Solutions data breach [25].

The FCC has since reopened docket 17-108 to additional comments for a period ending March 30, 2020 [8], and subsequently extended to April 20, 2020 [7]. However, they have changed little in how the comments are collected. Assuming that these new proceedings are equally as likely to be stuffed with fake submissions, it becomes important to be able to find a way to extract the valid form-letter submissions used by legitimate campaigns. Citizens submit form-letters to easily to make their opinions known, often aided by third parties who coordinate the campaigns. Spam detection tools that classify based on content duplication are not appropriate for this task.

2 Objectives

This project will build on the work done by Jeff Kao [13, 11, 14] and the Startup Policy Lab [26] in which they identified clusters of comments (campaigns) and surveyed a sampling of commenters to see if they actually made the respective comment. This data will be merged with the complete dataset of comments containing additional metadata captured in the FCC [5] proceedings.

This project will not cluster comments or consider the sentiment of the comments, instead relying on the campaign groupings of prior analyses. It will instead focus on separating fake comments from valid form-letter-style comments, a task prior analyses have deemed difficult.

The following will be explored.

1. Evaluate the most promising existing ML learning algorithms to determine if any are capable of accurately categorizing a comment as either fake or valid using the survey results at the ground truth.

2. Generate additional comment features quantifying the rate at which comments are being received at the time the comment was submitted. Evaluate if these features improve categorization.

3 Related Works

3.1 Deceptive Comment and Spam Detection

Oh and Park [20] studied the detection of deceptive comments using a dataset consisting of 866 truthful and 869 deceptive comments on social issues. Using a Support Vector Machine and Neural Networks they achieved models with 89% accuracy on their test set. This performance is quite consistent across social issues and well beyond that of human judges.

Etaiwi and Naymat [3] investigated the effects of preprocessing steps on the accuracy of review spam detection. Different machine learning algorithms were applied, such as Support Victor Machine (SVM) and Naive Bayes (NB), and a labeled dataset of Hotels reviews were analyzed and processed. Results were mixed, with no specific preprocessing identified as consistently beneficial.

According to Han et al. [9], Logistic Regression is one of the successful methods for spam filtering, but it still has certain disadvantages. In this research, the LR model training was modified to accelerate the weight changes on features more frequently associated with either spam or ham, but not both. This resulted in competitive spam detection.

Lin et al. [16] conducted an experiment with true hostel review comments from "TripAdvisor" and the comparison group "Fake reviews" on Amazon Mechanical Turk. Applying Logistic Regression to the data set, they obtained performance that was better than the benchmark method which is based on Linguistic Inquiry and Word Count (LIWC) and SVM.

Rayana and Akoglu added metadata to improve their results to their previous work that used Markov Random Fields (MRF) to classify fake and real reviews [22]. The dataset they used was a collection from yelp.com reviews containing ratings and timestamps as metadata in addition to their relational data. Their results were superior to several baselines and state-of-the-art techniques after adding the metadata results.

The behaviour-based approaches often leverage indicative features of spam extracted from the metadata associated with user behavior, content, and profile. Li et al. trained semi-supervised models, and used the two views from reviews and users under a co-training framework to spot fake reviews [15]. Ye and Akoglu proposed graph-based methods to identify group spammers solely based on their abnormal network footprints [29]. Jindal and Liu use supervised learning based on 36 such features on a (pseudo) ground truth dataset, constructed by labeling the duplicate reviews in an Amazon dataset as fake reviews [10].

3.2 Research about FCC 17-108

In attempting to extract the sentiment from valid comments, Singel [24] chose to focus on unique comments, noting that due to the large amount of noise created by fake comments, it remains very difficult to locate the real signals in the non-unique comments.

Weiss [28] found that the problem of separating bot comments from valid ones is only getting more difficult with the advent of Deepfake Text generation, and suggests implementing technological reforms (e.g., CAPTCHAs) [...] to help prevent massive numbers of submissions by bots. Weiss points out that the FCC is considering adding CAPTCHAs to the comment filing process, though it is not clear if that would include the filing APIs, which were used to file the vast majority of 17-108 comments. They also note that even the most advanced CAPTCHAs are easily broken by determined adversaries.

Kao used Natural Language Processing techniques to analyze the FCC 17-108 comments [11, 13, 14]. His analysis did not include the use of metadata to improve his results and found three key findings, as stated in [12]:

1. One pro-repeal spam campaign used mail-merge to disguise 1.3 million comments as unique grassroots submissions.
2. There were likely multiple other campaigns aimed at injecting what may total several million pro-repeal comments into the system.
3. Its highly likely that more than 99% of the truly unique comments were in favor of keeping net neutrality.

4 Datasets and Features

The datasets used in generating models for comment classification were built using the following methodology.

1. Identify interesting features in the source datasets.
2. Extract and standardize the feature representation.
3. Derive features with the potential of improving accuracy.
4. Eliminate features unlikely to contribute to accurate classification.
5. Define subset feature spaces for use in evaluating feature importance.
6. Use PCA to reduce the dimensionality of the feature spaces.

4.1 Source Datasets

The FCC provides a public API for querying submissions to proceedings.[6] Kao used this API to periodically harvest comment submissions and create a 64GB dataset of over 22 million comments. He then generated word and document vectors from the comment text and clustered the comments as either "unique" or as one of nearly 150 "campaigns".[12]

4.1.1 Startup Policy Lab Survey

Belle et al. set out to survey the participants in the campaigns identified by Kao's clustering to determine if they had truly submitted their attributed comment to the FCC. Starting with the 100 largest campaigns, only campaigns with at least 12,000 comments were selected to participate. From each of the 42 selected campaigns, a random sample of the comments were selected to be surveyed. An additional pseudo-campaign of unique comments not otherwise clustered was also included and sampled.[1]

A total of 456,288 e-mails were sent in six separate blocks.[1] This implies that in order to be surveyed, the commenter must have provided a valid e-mail address, which the FCC does not require. A scan of the FCC dataset shows that of the 22,158,109 comment submissions, 19,842,880 include a non-empty contact_email element and 19,816,741 of those contact_emails validate successfully (address properly formatted and successful DNS MX record lookup for the domain) using the email_validator Python library[27].

For our project, we are assuming that fake comments in a campaign will either all have a valid contact_emails, or none will. This implies that a comment being fake and the supplied contact_email being valid are conditionally independent events given the campaign. Models built on the email survey of campaign comments can be applied to other campaigns where emails were not valid.

The dataset published by the Startup Policy Lab contains 449,658 instances. The following interesting features are included in the feature space.

- **campaign** - (categorical) The integer campaign ID assigned to the comment.
- **email_hash** - (md5hash) The MD5 hash of the contact_email information provided with the comment.
- **send_failed** - (boolean) True if the survey email was not delivered for any reason, including no contact_email given, delivery bounced, or auto-reply received. False if email was accepted by the recipient's mail server. Note: this feature does not indicate if the survey email was read.
- **bounced_or_filtered** - (boolean) True if the recipient's mail server replied with an error or auto-response. False otherwise. Note: If both send_failed and bounced_or_filtered is False, the email is considered delivered.
- **not_original_commenter** - (boolean) 1 if the recipient responded that they did not submit the comment to the FCC. 0 if the recipient responded that they did. Empty if no response recorded.

When building models, only instances where not_original_commenter is not empty were used, meaning a survey response was received. This biases the training data towards comments that were legitimate, and towards fake comments where the forger used stolen identities. Forgers and anonymous valid commenters using invalid e-mail addresses or no contact_email at all are not represented in the training data. This should be acceptable given the conditional independence assumption between email_valid and not_original_commenter given the campaign.

4.1.2 FCC 17-108 Proceedings Submissions

On November 7, 2017, the FCC published three zip files containing all information on submissions to the 17-108 proceedings.[5]. The data is divided into JSON files containing 10,000 submissions each. In total, 22,158,109 comments are included. For this project, these JSON elements were selected for analysis.

- **addressentity.city** - (string) The self-reported city where the commenter lived. The FCC performed no validation on this data.
- **addressentity.state** - (string) The self-reported state where the commenter lived. The FCC performed no validation on this data.
- **contact_email** - (string) The self-reported email address of the commenter. The FCC neither required nor validated this data.
- **date_submission** - (date) Date and time when the comment was sent.
- **date_received** - (date) Date and time when the comment was processed.
- **date_disseminated** - (date) Date and time when the comment was made available via the API.
- **emailConfirmation** - (boolean) "true" if the commenter requested an email response confirming the submission. "false" if confirmation not requested. Missing if not included in the submission.
- **express_comment** - (boolean) 1 if submission was made in the express comment format. 0 otherwise.
- **submissiontype.abbreviation** - (categorical) The type of the proceedings submission. "CO" means comment. "OP" means opposition. Other types of submissions are omitted from this project.
- **text_data** and **text_data_docs[].data** - (text) Content of the free-form comment being submitted.

4.2 Feature Representation

- **boolean** - Boolean features were standardized to the following integer values.
  - False: -1
  - Unknown: 0
  - True: 1

- **string** - Free-form strings were transformed into lowercase and had extraneous spaces removed. The number of words in a string was limited to 10 to weed out garbage data being submitted by pranksters.
- **md5hash** - A 32 character MD5 hash hexdigest generated from a non-empty string.
- **categorical** - These features were standardized as one-hot vectors.
- **date** - Date strings were converted to floating point numbers equaling the number of seconds since the epoch (1970-01-01 00:00:00.000 UTC) with millisecond resolution using the dateutil Python library.[17]
- **interval-bin** - Date features can be used to group instances by a common time interval containing the date. The minimum interval size is one minutes.
- **text** - Text data was processed into a bag-of-words vector using the NLTK Python library[18] to tokenize the data and remove non-English words along with English stop words. Additionally, the name of the filer was removed from the comment if it was included in the submission.

4.3 Derived Features

4.3.1 Comment Submission Interval Bins

This project theorizes that using the rate of comment submissions at the time of a particular submission may be helpful in predicting if the submitted comment is fake or legitimate. To quantify this rate, comments were placed in two bins based on the minute during which the comment was submitted: one bin for all comments in the FCC dataset, and another bin for comments from the same campaign (if the comment was associated with a campaign). Then features were generated for each comment in the survey dataset as follows.

- **all_submitted_bin** (interval-bin) - Number of comments found in the 10 one-minute all-comment bins starting at $\lfloor \text{date\_submission}/600 \rfloor$.
- **all_centered_bin** (interval-bin) - Number of comments found in the 10 one-minute all-comment bins centered around date_submission.
- **campaign_submitted_bin** (interval-bin) - Number of comments found in the 10 one-minute campaign-comment bins starting at $\lfloor \text{date\_submission}/600 \rfloor$.
- **campaign_centered_bin** (interval-bin) - Number of comments found in the 10 one-minute campaign-comment bins centered around date_submission.

The following chart shows that as the rate of all-comment submissions increases, the likelihood that a comment is fake also increases.

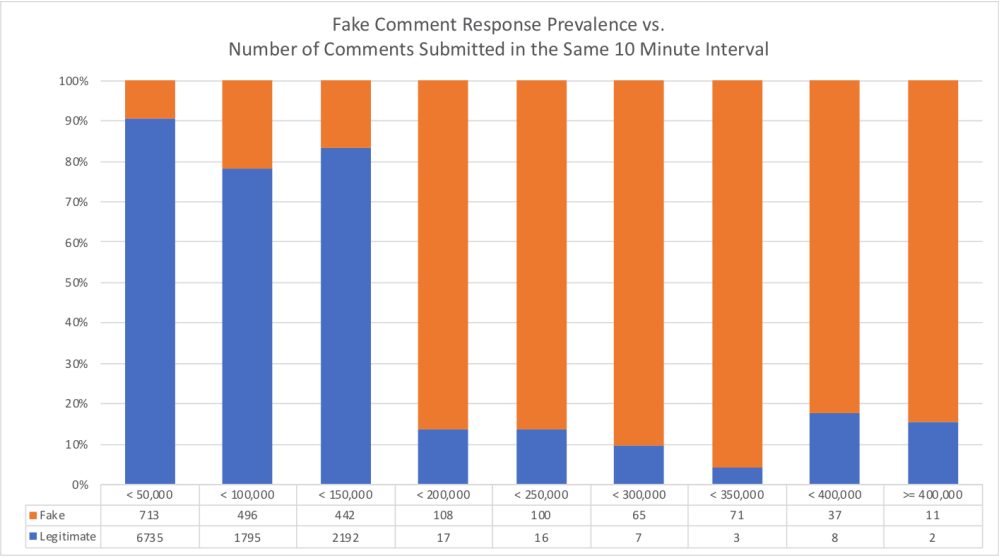

Figure 1:

4.3.2 Other Features

The following features were also synthesized from the source data to facilitate analysis.

- **email_hash** (md5hash) - Non-empty contact_email elements were replaced with a MD5 hash hexdigest of the supplied string. It was then used to match instances from the survey dataset with the corresponding FCC instances.
- **city_state** (categorical) - This feature is the combination of the addressentity.city and addressentity.state string elements, separated by a space. The state element by itself was considered too broad, and the city element was ambiguous without the state element.

5 Methodology and Experimental Setup

5.1 Feature Elimination

An effort was made to reduce feature space dimensionality by eliminating features that did not contribute to classification accuracy.

5.1.1 Low Variance

The following features were eliminated due to low variance among the training instances.

- **express_comment** - This feature was always equal to 1 for instances where a survey response was received.
- **submissiontype.abbreviation** - This feature was always equal to "CO" for instances where a survey response was received.
- **city_state** - This feature, when transformed into a one-hot vector, added 8449 dimensions to the feature space. This caused the PCA software being used to consume more than 32GB of memory and fail to complete. Eliminating one-hot features for city_state values that occur only once reduced the number of features added to 3762. This also had the benefit of removing many misspelled city_state values.
- **text** - The comment text bag-of-words vocabulary contained 29327 unique words pulled from all instances in the FCC dataset. To keep dimensionality manageable, the vector representation was limited to the top 100 most commonly occurring words among the training instances.

5.1.2 Least Importance

- **city_state** - Given the large number of dimensions added by the city_state one-hot vector, an experiment was conducted to determine if this feature improved the accuracy of the models generated. Logistic regression and SVM models were built both with and without this feature. Both learning methods showed a reduction in accuracy when the city_state features were included. For SVM, adding the 80 most frequent dimensions reduced test accuracy from just under 94% to under 88%. For logistic regression, a more modest reduction in test accuracy was seen, from 85.5% to 83.7%. For this reason, we removed the city_state features from the feature spaces being studied.

5.2 Feature Spaces Modeled

A goal of this project is to find features that help accurately predict if a comment from a campaign is fake or legitimate. To facilitate this analysis, we divided the total feature space into five subsets of interest.

1. Baseline: date_disseminated, date_received, date_submission, emailConfirmation
2. Baseline + Campaigns: adds campaign one-hot vectors to #1.
3. Baseline + Campaigns + Bins: adds campaign_submitted_bin and campaign_centered_bin to #2.
4. Baseline + Text Bag-of-Words: adds text bag-of-words vectors to #1.
5. Baseline + Text Bag-of-Words + Bins: adds all_submitted_bin and all_centered_bin to #4.

5.3 PCA Dimension Reduction

The PCA algorithm was used to further reduce feature set dimensions and it was implemented in Python using NumPy.[19] We chose the number of components for each subset aiming at retaining at least 95% of the original data variance and achieved the following results.

- Subset 1: From 4 features to 2 Principal Components
- Subset 2: From 46 features to 40 Principal Components
- Subset 3: From 48 features to 40 Principal Components
- Subset 4: From 104 features to 22 Principal Components
- Subset 5: From 106 features to 23 Principal Components

5.4 Learning Algorithms

In the Related Work section, we see that Logistic Regression [9, 16], Neural Networks [3, 20], and Support Vector Machines [3, 20] are machine learning methods that provide consistent results to the task of text classification and detection of fake/spam comments. We implemented these algorithms, and studied the resulting performance on the FCC and survey datasets.

For training all classification models, only instances where `not_commenter` is not '0.0' were included, since 0.0 indicates that no response to the survey was recorded. The remaining data was normalized and split into train/test data in a 80/20 ratio respectively, in a stratified way. Each model was trained using 10-fold cross validation.

5.4.1 Logistic Regression

Logistic Regression (LR) has been used to perform Natural Language Processing tasks and obtains good results evaluating false and spam comments [9, 16]. LR is characterized by a logistic function to model the conditional probability of the label Y with variables X, $P(Y|X)$, that can be represented by a *sigmoid* function,

$$\sigma(\theta^\top X) = \frac{1}{1 + e^{-\theta^\top X}}$$

where θ is the vector containing the model parameters and is selected by a Maximum Likelihood Estimation (MLE). The implementation was made using the `sklearn` [23] python library and set up to use the limited-memory BFGS solver with L2 regularization.

5.4.2 Fully-Connected Neural Network

The Fully-Connected Neural Network model was built using programming language R and the h2o package.[2] The algorithm was evaluated on 10 different datasets: each of the five subsets of interest, with and without PCA. A Grid Search algorithm was performed for each fold of each dataset, using 5-fold stratified cross-validation and choosing based on the highest classification accuracy. The models were trained using cross-entropy loss function and 30 epochs. They had one hidden layer, and the hyperparameters used for tuning were:

- Hidden layer sizes:
  - 0.10 * (number of features)
  - 0.25 * (number of features)
  - 0.50 * (number of features)
  - 0.75 * (number of features)
  - 1.00 * (number of features)
- Activation functions: Rectified linear unit (ReLU) and Hyperbolic tangent (TanH)
- Dropout Regularization: 0, 0.1 and 0.2
- Lasso (L1) Regularization: 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}

- Ridge (L2) Regularization: 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}

The following search criteria were used:

- Strategy: Random
- Stopping metric: Misclassification
- Total of 100 models evaluated

5.4.3 SVM

For each feature space subset, we use 10 fold cross validation on the training set to find the best possible hyperparameters. We used sklearn to generate the SVM models.[21] The rbf kernel was used, along with the following parameters.

Subset	C	gamma
1	10000	10.0
2	1000	0.1
3	1000	1.0
4	1000	0.1
5	1000	0.1

Table 1: SVM Parameters

After choosing the best parameters using cross validation, we used the training data to tune the SVM hyperparameters and evaluate its performance on the test set.

5.5 Metrics

Performance of the generated models was measured using the accuracy, precision, recall, $F1$, and AUC scores, similar to related works[3, 9, 16, 20]. The metrics are defined as follows.

- True Positives (TP): number of positive labels correctly predicted
- True Negatives (TN): number of negative labels correctly predicted
- False Positives (FP): number of items incorrectly predicted as positive
- False Negatives (FN): number of items incorrectly predicted as negative
- Accuracy $= \frac{TP+TN}{TP+FP+FN+TN}$
- Precision $= \frac{TP}{TP+FP}$
- Recall $= \frac{TP}{TP+FN}$
- $F1 = 2\frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$
- AUC is the area under the ROC (Receiver Operating Characteristic) curve.

6 Results

6.1 Logistic Regression

Subset	Accuracy	Precision	Recall	F1	AUC
1	0.855	0.747	0.169	0.276	0.708
2	**0.957**	0.895	**0.838**	**0.866**	**0.928**
3	**0.957**	0.895	**0.838**	**0.866**	0.924
4	0.952	0.892	0.804	0.846	0.917
5	0.954	**0.899**	0.807	0.850	0.918

Table 2: Logistic Regression scores

Subset	Accuracy	Precision	Recall	F1	AUC
1	0.837	0.000	0.000	—	0.621
2	**0.952**	**0.876**	**0.825**	**0.850**	0.921
3	**0.952**	**0.876**	**0.825**	**0.850**	**0.925**
4	0.929	0.852	0.687	0.760	0.905
5	0.939	0.872	0.732	0.796	0.912

Table 3: Logistic Regression scores with PCA

All LR models (with and without PCA) were trained on a Department of Computer Sciences Instructional Linux Computer at the University of Wisconsin-Madison, and took less that 10 minutes to train. The training the subset 1 with PCA resulted in a zero precision and recall, meaning that it was only predicting the majority class and not using any of the information of the PCA features. Both experiments (training with PCA and without PCA) showed best performance using the Baseline + Campaign (2) and Baseline + Campaign + Bins (3) subsets.

6.2 Fully-Connected Neural Network

Subset	Accuracy	Precision	Recall	F1	AUC
1	0.688	0.201	0.297	0.240	0.619
2	**0.951**	**0.876**	**0.818**	**0.846**	0.918
3	0.950	**0.876**	0.816	0.845	**0.919**
4	0.948	0.863	**0.818**	0.840	0.915
5	0.948	0.861	**0.818**	0.839	0.916

Table 4: Artificial Neural Network scores

Subset	Accuracy	Precision	Recall	F1	AUC
1	0.313	0.192	**0.986**	0.322	0.634
2	**0.950**	**0.877**	0.809	**0.842**	0.919
3	0.949	0.875	0.807	0.839	**0.921**
4	0.948	0.866	0.809	0.837	0.915
5	**0.950**	**0.877**	0.809	**0.842**	0.914

Table 5: Artificial Neural Network scores with PCA Features

Reducing the subsets' dimensions using PCA didn't affect overall performance, and it greatly decreased average time required to train the models, specially on subsets 4 and 5:

- Subset 1: From 8 to 6 minutes (-25%)
- Subset 2: From 60 to 55 minutes (-8%)
- Subset 3: From 62 to 55 minutes (-11%)
- Subset 4: From 335 to 27 minutes (-92%)
- Subset 5: From 336 to 29 minutes (-91%)

6.3 Support Vector Machine

Subset	Accuracy	Precision	Recall	F1	AUC
1	0.942	0.842	0.796	0.818	0.881
2	**0.951**	0.875	**0.818**	**0.845**	0.890
3	**0.951**	0.876	**0.818**	**0.845**	**0.900**
4	0.947	0.876	0.787	0.829	**0.900**
5	0.945	**0.877**	0.778	0.824	0.899

Table 6: SVM scores

Results:

- The most significant difference between SVM and the other two algorithm is the SVM performs extremely well on the original baseline dataset, achieving 94% accuracy.
- The campaign features help increase the prediction accuracy as well as recall and f1 score.
- Other features including submitted bins and bag-of-words do not improve prediction much.
- Overall the SVM models work well, and achieved an average accuracy over 94%

7 Predictions on Remaining Surveyed Comments

As has been mentioned, the models were trained and tested using instances from the survey dataset where a response was recorded, for a total of 12,818 instances. That leaves 436,827 survey instances with unknown validity. Although all algorithms performed well, the logistic regression models were chosen for their accuracy and speed to classify these instances.

Subset	Predicted Fake	Percent Fake
1	62165	14.231
2	181007	41.437
3	176991	40.517
4	148006	33.882
5	161765	37.032

Table 7: Comments Predicted as Fake by LR

Despite the bag-of-words subset model (4) having nearly identical precision and accuracy when compared to the campaign subset (2) model, bag-of-words predicted significantly fewer fake comments.

More interestingly, when the bins were added to both subsets, bag-of-words+bins (5) predicts more fakes than (4), and campaigns+bins (3) predicts fewer than (2). One possible explanation for this is that the unknown ground-truth number of fake comments lies somewhere between the model 2 and 4 predictions, and the addition of the interval-bins allowed both models to become more accurate and predict closer to the ground-truth.

8 Conclusions and Future Work

The project found that campaign clustering was by far the most influential feature in predicting fake comments. Bag-of-words vectors generated from the comments turned out to be a reasonable substitute for the campaign feature, and the interval bins did little to improve the model test accuracy.

When presented with comments for which no survey response was recorded, the bag-of-words model and the campaign model predictions diverged, with bag-of-words predicting 33,001 fewer fake comments. Adding the bins to both models brought them closer to consensus, reducing the gap to 15,226. This suggests that the interval-bins may improve prediction accuracy for comments from previously unseen campaigns. Future work could involve training models on comments from subsets of campaigns to see how they perform of the remaining campaigns.

All the models showed an greater precision than recall. This would mean that there are more false negatives than false positives. Since the training data contained more negative labels than positive labels, it would be expected of an unbiased classifier to have more false positives (mislabeled negatives) than false negatives (mislabeled positives). Having the opposite might be an indicative of a bias error related to the cardinality of each class used for training: as we have more samples in negative class, it would be easier to predict true or false negatives.

A possible future work is the analysis of impacts of using imbalanced classes and the re-evaluation of the models correcting the data trough undersampling, oversampling, or adding synthetic data (with SMOTE, VAE, GAN, or other techniques).

Code and Jupyter notebooks used in this project have been made available on GitHub at https://github.com/wisc-landherr/cs760-fcc.

References

[1] Charles Belle, Gina Cooper, Jeffery Kao, and Sarah Rigdon. The Federal Communications Commissions public comment process for Net Neutrality was deliberately manipulated using false identities. https://static1.squarespace.com/static/554441dae4b07d3f990170ea/t/5a32d72d9140b78109fd4dd4/1513281327086/SPL+TiPC+Preliminary+Report+12-14-17.pdf. Retrieved 2020-04-30.

[2] Erin LeDell and others. R Interface for the 'H2O' Scalable Machine Learning Platform. https://cran.r-project.org/package=h2o, 2020. Version 3.30.0.1.

[3] Wael Etaiwi and Ghazi Naymat. The Impact of applying Different Preprocessing Steps on Review Spam Detection. In Shakshuki, E, editor, *8th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN 2017) / 7th International Conference on Current and Future Trends of Information and Communication Technologies in Healthcare (ICTH-2017) / Affiliated Workshops*, volume 113 of *Procedia Computer Science*, pages 273–279, SARA BURGERHARTSTRAAT 25, PO BOX 211, 1000 AE AMSTERDAM, NETHERLANDS, 2017. ELSEVIER SCIENCE BV. doi: {10.1016/j.procs.2017.08.368}. 8th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN) / 7th International Conference on Current and Future Trends of Information and Communication Technologies in Healthcare (ICTH), Lund, SWEDEN, SEP 18-20, 2017.

[4] FCC. CONSUMER AND GOVERNMENTAL AFFAIRS BUREAU GUIDANCE ON FILING COMMENTS IN THE RESTORING INTERNET FREEDOM PROCEEDING. https://ecfsapi.fcc.gov/file/0427547924954/DOC-344623A1.pdf, 2017. Retrieved 2020-04-30.

[5] FCC. FCC FACILITATES REVIEW OF RESTORING INTERNET FREEDOM RECORD. https://apps.fcc.gov/edocs_public/attachmatch/DA-17-1089A1.pdf, 2017. Retrieved 2020-04-30.

[6] FCC. ECFS Public API Documentation. https://www.fcc.gov/ecfs/help/public_api, 2020. Retrieved 2020-03-30.

[7] FCC. ORDER. https://ecfsapi.fcc.gov/file/032574136961/DA-20-331A1.pdf, 2020. Retrieved 2020-04-30.

[8] FCC. WIRELINE COMPETITION BUREAU SEEKS TO REFRESH RECORD IN RESTORING INTERNET FREEDOM AND LIFELINE PROCEEDINGS IN LIGHT OF THE D.C. CIRCUITS *MOZILLA* DECISION. https://ecfsapi.fcc.gov/file/0219046559245/DA-20-168A1.pdf, 2020. Retrieved 2020-04-30.

[9] Yong Han, Muyun Yang, Haoliang Qi, Xiaoning He, and Sheng Li. The Improved Logistic Regression Models for Spam Filtering. In Zhang, M and Li, HZ and Lua, KT and Dong, MH, editor, *2009 INTERNATIONAL CONFERENCE ON ASIAN LANGUAGE PROCESSING*, International Conference on Asian Language Processing, pages 314–317, 10662 LOS VAQUEROS CIRCLE, PO BOX 3014, LOS ALAMITOS, CA 90720-1264 USA, 2009. IEEE COMPUTER SOC. ISBN 978-0-7695-3904-1. doi: {10.1109/IALP.2009.74}. International Conference on Asian Language Processing, Singapore, SINGAPORE, DEC 07-09, 2009.

[10] Nitin Jindal and Bing Liu. Opinion spam and analysis. In *Proceedings of the international conference on Web search and web data mining - WSDM 08*. ACM Press, 2008. doi: 10.1145/1341531.1341560. URL https://doi.org/10.1145%2F1341531.1341560.

[11] Jeff Kao. FCC Net Neutrality Comments Clustered. https://www.kaggle.com/jeffkao/fcc-net-neutrality-comments-clustered/version/1, 2017. Retrieved 2020-03-25.

[12] Jeff Kao. More than a Million Pro-Repeal Net Neutrality Comments were Likely Faked. https://hackernoon.com/more-than-a-million-pro-repeal-net-neutrality-comments-were-likely-faked-e9f0e3ed36a6, 2017. Retrieved 2020-04-02.

[13] Jeff Kao. FCC Net Neutrality Comments (4/2017 - 10/2017). https://www.kaggle.com/jeffkao/proc_17_108_unique_comments_text_dupe_count/version/1, 2017. Retrieved 2020-03-25.

[14] Jeff Kao. FCC Net Neutrality Comments Vectorized Sample. https://www.kaggle.com/jeffkao/proc_17_108_level0_unclustered_sample_vectorized/version/1, 2017. Retrieved 2020-03-25.

[15] Fangtao Li, Minlie Huang, Yi Yang, and Xiaoyan Zhu. Learning to identify review spam. In *IJCAI*, 2011.

[16] Mei Yu Lin, Ping Yu Hsu, Ming Shien Cheng, Hong Tsuen Lei, and Ming Chia Hsu. Identifying Fake Review Comments for Hostel Industry. In Tan, Y and Takagi, H and Shi, Y and Niu, B, editor, *ADVANCES IN*

SWARM INTELLIGENCE, ICSI 2017, PT II, volume 10386 of *Lecture Notes in Computer Science*, pages 421–429, GEWERBESTRASSE 11, CHAM, CH-6330, SWITZERLAND, 2017. Peking Univ, Computat Intelligence Lab; Kyushu Univ, Res Ctr Appl Perceptual Sci; IEEE Computat Intelligence Soc; IEEE Syst, Man & Cybernet Soc, Japan Chapter, SPRINGER INTERNATIONAL PUBLISHING AG. ISBN 978-3-319-61833-3; 978-3-319-61832-6. doi: {10.1007/978-3-319-61833-3_45}. 8th International Conference on Swarm Intelligence (ICSI), Fukuoka, JAPAN, JUL 27-AUG 01, 2017.

[17] Gustavo Niemeyer, Tomi Pieviläinen, Yaron de Leeuw, and Paul Ganssle. dateutil - powerful extensions to datetime. `https://pypi.org/project/python-dateutil/`, 2019. Version 2.8.1.

[18] NLTK Project. Natural Language Toolkit. `https://www.nltk.org/`, 2019. Version 3.4.5.

[19] NumPy developers. NumPy. `https://numpy.org/`, 2020. Version 1.18.1.

[20] Yu Won Oh and Chong Hyun Park. Machine Cleaning of Online Opinion Spam: Developing a Machine-Learning Algorithm for Detecting Deceptive Comments. *AMERICAN BEHAVIORAL SCIENTIST*. doi: {10.1177/0002764219878238}.

[21] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[22] Shebuti Rayana and Leman Akoglu. Collective opinion spam detection. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD 15*. ACM Press, 2015. doi: 10.1145/2783258.2783370. URL `https://doi.org/10.1145%2F2783258.2783370`.

[23] ScikitLearn. LogisticRegressionCV. `https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegressionCV.html`. Retrieved 2020-05-06.

[24] Ryan Singel. Filtering Out the Bots: What Americans Actually Told the FCC about Net Neutrality Repeal. `https://cyberlaw.stanford.edu/files/blogs/FilteringOutTheBotsUnique2017NetNeutralityComments1024Update.pdf`, 2018. Retrieved 2020-04-02.

[25] Jeremy Singer-Vine and Kevin Collier. Political Operatives Are Faking Voter Outrage With Millions Of Made-Up Comments To Benefit The Rich And Powerful. `https://www.buzzfeednews.com/article/jsvine/net-neutrality-fcc-fake-comments-impersonation`, 2019. Retrieved 2020-04-30.

[26] Startup Policy Lab. FCC Public Comment Survey Results Deidentified. `https://www.kaggle.com/startuppolicylab/fcc-public-comment-survey-results-deidentified/version/1`, 2017. Retrieved 2020-03-25.

[27] Joshua Tauberer. email_validator. `https://pypi.org/project/email-validator/`, 2020. Version 1.1.0.

[28] Max Weiss. Deepfake Bot Submissions to Federal Public Comment Websites Cannot Be Distinguished from Human Submissions. `https://techscience.org/a/2019121801/`, 2019. Retrieved 2020-04-02.

[29] Junting Ye and Leman Akoglu. Discovering opinion spammer groups by network footprints. In *Machine Learning and Knowledge Discovery in Databases*, pages 267–282. Springer International Publishing, 2015. doi: 10.1007/978-3-319-23528-8_17. URL `https://doi.org/10.1007%2F978-3-319-23528-8_17`.