Technical Report for a Boil-order Classifier Using Natural Language Processing Techniques on Google News English Articles

Chris Mantell, General Assembly, December 2017

Introduction

The prolonged and grossly mishandled water crisis in Flint, Michigan, starting in 2014, has shown that there is a need for transparent and easily accessible information regarding water quality issues. Although Flint is on the correct path back to a city providing clean water, other cities and towns in the United States and its territories are potentially suffering from tainted water. This is partially because information is not being circulated effectively as it eventually was in Flint. In 2017, the citizens of Moore Bend, Missouri, were notified of a standing boil order alert that was in place since 2013[1]. This is just one extreme example and reason why boil orders should be tracked. This technical report describes a supervised machine learning classification model to automatically detect boil-order alerts using Natural Language Processing (NLP) on online news articles.

Methods

Data Acquisition

Data for this project was provided by the HealthMap, a disease outbreak and epidemiological surveillance tool. The HealthMap collects information from many various sources in their data feeds (ex: Google News, World Health Organization (WHO), Baidu News, etc.) and automatically classifies it by disease or disease category (ex: Avian Influenza H7N9, Fever, Environmental, Foodborne Illness, etc.). Information collected from the tool is curated

by humans knowledgeable in public health and edits the data to increase information accuracy and/or correct information that is misclassified by the HealthMap. A .csv file of "Waterborne Illness" alerts in the United States from the Google News English feed was obtained on 10/27/17 and contained data from 5/10/10 – 10/27/17. Each row was a different location where an alert has taken place. A URL to the original link pulled from the HealthMap that holds the news article is provided as well as a headline of the article, an "Issue Date" of when the article was written, and a "Smooshed Parser Extract", which is an automated extraction of text from the article. The original dataset contained 7725 rows and 119 columns, which contained information describe above as well as many unnamed columns resulting from a mistake or an anomaly in data entry. Some articles referenced multiple locations and in this case, different rows reference the same article.

Data Cleaning and Exploratory Data Analysis (EDA)

After data cleaning, the dataset consisted of 7724 rows and 14 columns as described in the data dictionary (Table 1).

The author of this technical report was previously a data curator with the HealthMap from 2016 – 2017 and has previous knowledge about what kind of information was brought into the "Waterborne Illness" alerts. Most alerts relating to boil-orders are classified as "Warning" under the "Alert Tag." Value counts of the type of alert tags are graphed as shown in Figure 1.

Data Dictionary	
Column	**Description (Example)**
“Location”	Location where the alert is referencing (Delaware, United States)
“Country”	Country where the alert is referencing (United States)
“Disease”	Disease the alert is referencing (Waterborne Illness)
“HM Alert”	Unique HealthMap (HM) alert ID (5407609)
“Headline”	The Headline of the news article (‘Kutcher tells Council about water situation - Murray Ledger and Times’)
“URL”	URL pulled by the HM referencing the original article at the date it was pulled (http://news.google.com/news/url?sa=t&fd=R&ct2=us&usg=AFQjCNGmA0yAoTj-LEPjIRxm5NB9akVvVw&clid=c3a7d30bb8a4878e06b80cf16b898331&cid=52779651867230&ei=Cc7yWaC_H9G7zAKcpws&url=http://murrayledger.com/news/kutcher-tells-council-about-water-situation/article_2020569a-bac6-11e7-82f8-1f4b918f38ae.html)
“Issue Date”	The date and time the web article was published (10/27/17 1:11)
“Alert Tag”	The classification of the type of alert. This ranks the alert in importance or whether it references information relating to a disease but not necessarily an outbreak (Breaking, Warning, Content, No Tag)
“Dup Count”	The number of duplicate articles referencing this alert
“Long”	Longitude of the location of the alert
“Lat”	Latitude of the location of the alert
“Smooshed Parser Extract”	Automated text extracted from the article (‘ XXXXXX Murray Ledger and Times)
“Place Category”	An extra categorization column to help give more information about the details of the alert (‘Natural disaster related’)

Table 1: A description of the different columns in the original dataset after initial cleaning.

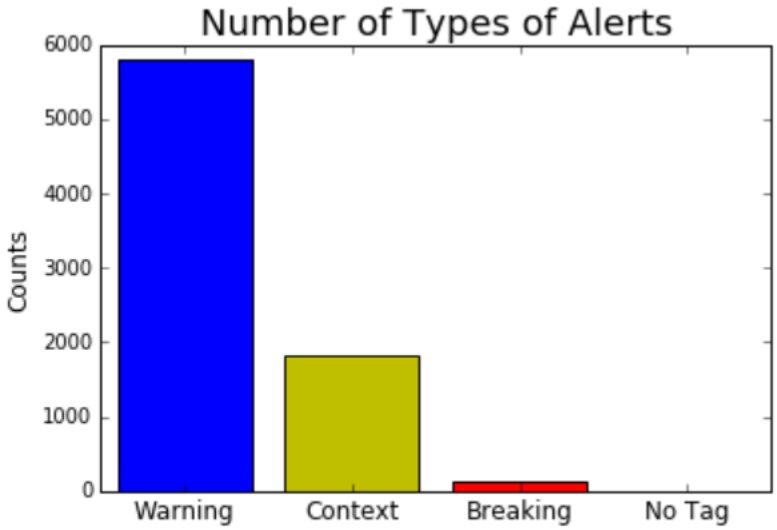

Figure 1: Counts of alerts grouped by “Alert Tag”

Initially, the “Smooshed Parser Extract” column was thought to be the text of the articles in the alerts but the data turned out to be missing many rows with 5,171/7,725 (66.9%) having non-null values. Furthermore, the values present were of questionable use for predicting boil-orders. To get the text of the articles, each URL was scraped to get the raw HTML/CSS and the results for each row were appended to the dataset. Each successful request that returned HTML/CSS results was then ran through custom built regular expression parser to target HTML paragraph tags and then remove undesirable text that was left, such as HTML tags and strings of non-predictive value (ex: LinkedIn, ‘’&amp’). The resulting text was used to make new features for modeling.

Only 4,849/7725 (62.8%) returned text while 2348/7725(30.4%) sites were deleted (404 or 410 errors with requests), 299/7725 (3.8%) had bad requests (400 or 500 errors other than 404 or 410), and 228/7725 (3.0%) sites had a successful request but retuned no text. The results of the web scrapping are showed in Figure 2 and the number of usable websites by the year of “Issue Date” are shown in Figure 3.

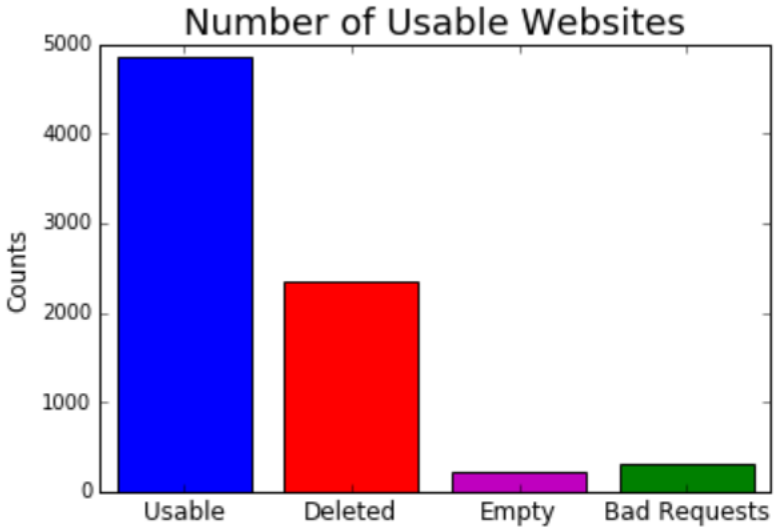

Figure 3: Results of scraping useable text from the dataset

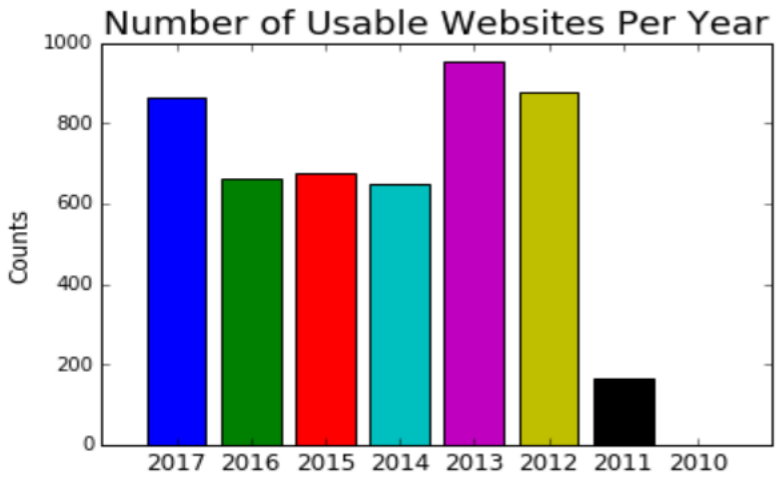

Figure 3: Results of scraping usable text from the dataset grouped by year of "Issue Date"

Feature Creation and Extraction

The results of the parsed text from the web scraping was preprocessed to remove stop words, remove punctuation, remove numbers, stem the words, and then the resulting text was put into a count vectorizer with a minimum document frequency (min_df) of five articles. This was set low to avoid removing potential important words that could differentiate between boil-orders and non boil-orders. 10,407 words were returned including non-descriptive and nonsensical text. This was used to run a Latent Dirichlet Allocation (LDA) unsupervised machine learning technique.

Because the results of the web scraping yielded a substantial amount of missing data and non-descriptive text, another set of features were extracted for a separate model building pipeline using the Headline text of the entire dataset. The same preprocessing steps were applied as above and then ran through a count vectorizer with a min_df of two articles to remove non-descriptive text. The resulting text was used for a separate LDA.

LDA was performed on the web-scraped text and the headlines. Wordclouds were created for each topic. These, along with the top 25 most frequent words per topic were used to infer a category based on the author's domain knowledge of what kind of articles are pulled into the "Waterborne Illness" section of the HealthMap. This process was repeated, each time increasing the number of computer generated topics by one, until the author could infer multiple nonsensical topics. The final number of topics was the minimum number resulting in the least number of nonsensical topics. Figure 4 outlines this process. The resulting number of topics was four with the web scraped data and five for the headline data. The process assumes that LDA should be able to generate topics associated with boil-orders and other commonly reported topics relating to "Waterborne Illness" until there are only nonsensical topics left.

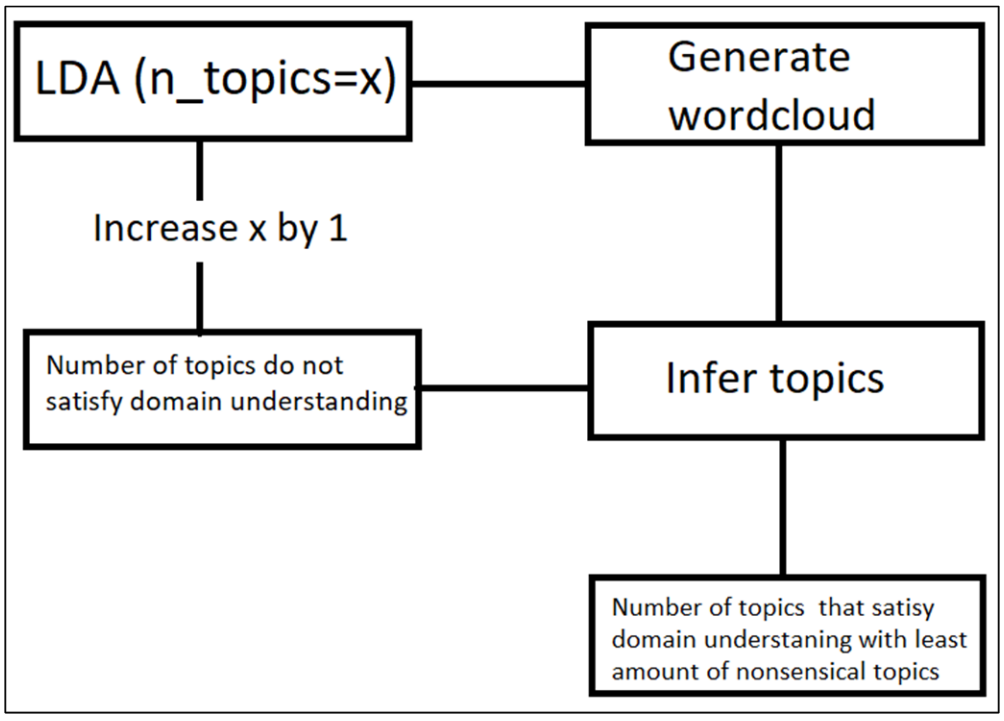

Figure 4: Schema to infer topics with LDA from text

The results of the LDA was used to transform the text data into an array of probabilities. The results are that each data point of scraped text or headline has a list of probabilities of belonging to each LDA generated topic. The data point with the highest probability of each topic was extracted, read, and compared with the topic inferred by the author to validate the LDA generated topic. The headline and the text with the highest probabilities should be congruent with the topic inferred by the author to pass this validation step.

The extracted text data did not pass this validation test but the headline data did. The headline data was further validated by appending each inferred topic to the dataset based on the highest probability of belonging to the associated LDA generated topic. Randomly selected headlines were read and compared with each associated inferred topic. Many headlines were congruent with the generated/inferred topics. This was used as a ground truth for a classification model.

Classification Model

The five topics inferred with LDA and domain knowledge, along with examples, are shown in Table 2. New boil-orders and existing boil-orders are difficult to distinguish just from a headline alone but they share the same words that indicate a boil order is present. These two topics were combined and a classification model was built to detect these combined topics against the rest of the topics. This yielded 6,408 boil-orders to classify against 1,316 non-boil orders, giving a baseline accuracy of 83% if a model were to classify the most frequent class.

Topic # - Topic Description	Example Headline	Count
0 - “New boil-order issued or repaired”	“Boil water advisory for parts of Summerville -ABC NEWS 4”	3428
1 – “Beach/Lake/Park closed from bacteria in water”	“Beaches closed due to elevated bacteria levels – Northwest Herald”	661
2 – “Hard to classify”	“Officials investigating after several fall ill at Yale School of Medicine – WTNH Connecticut News (press release)”	365
3 – “Drinking water contaminated”	“Puerto Ricans at Risk of Waterborne Disease Outbreak in Wake of Hurricane Maria – NBC Chicago”	290
4 – “Existing boil-order continuing or lifted”	“Boil Order in Silvis – cbs4qc.com”	2980
Total		**7725**

Table 2: Topic inferred from LDA with examples and counts across the dataset.

The data was split with a stratification, with a 5,175 (66.7%) data points used for training, and 2,549 (33.3%) used for testing. These separate datasets were put through a count vectorizer with the same preprocessing steps as listed above but with no min_df set. 4,560 columns were used to train multiple models. These models were validated across the test set, scoring for overall accuracy. Confusion matrices and classification reports were generated, which showed results for precision, recall, and f-1 score for both classes.

Multiple baseline models were tested with default parameters set with scikit-learn library, including random forest classifier (RFC), logistic regression with cross-validation (LogRegCV), K nearest neighbors (KNN), and four support vector machine (SVM) classifiers, each one with a different kernel (radial basis function[rbf], linear, a two-degree polynomial, and three degree-polynomial). The rbf-SVM and the polynomial SVM’s performed with baseline accuracy and were dropped as viable algorithms. The other four algorithms (RFC, LogRegCV,

KNN, and linear-SVM) performed better than baseline and were all very similar. These were then put into a bagging technique, which yielded comparable results. An adaptive boosting (AdaBoost) decision tree algorithm and AdaBoost LogRegCV were also performed with comparable results to the standard and bagged algorithms. Some hyperparameter optimization with some algorithms yielded comparable results as well.

Feature selection was changed to attempt to improve modeling results. Count vectorizers with a ngram range of (1, 2) with and without a min_df of two were used as feature extractions. These were used to train and test with bagged LogRegCV and bagged RFC. A count vectorizer with ngram range of (1, 2) was used along with a truncated single value decomposition (truncatedSVD) down to 3,000 components (covering ~95% of the variance). This was also used for the two bagged algorithms. Lastly, a Term Frequency Inverse Document Frequency vectorizer (Tfidf vectorizer) was used with/without a min_df of two and with/without an ngram range of (1,2). These feature extractors were used on the two bagged algorithms. Each feature selection technique used the same preprocessor as with the baseline modeling algorithms. All models performed very similar to each other across all different variations of feature selection techniques and algorithms.

A bagged LogRegCV and non-bagged LogRegCV were used to perform a grid search of different regularization values (Cs), cross-validations(cv), and maximum iterations (max_iter) over a standard count vectorizer with the same preprocessor. The best estimators performed similarly with the grid-searched non-bagged LogRegCV, however the standard LogRegCV had a better accuracy. This estimator was used for a Tfidf vectorizer with an ngram range of (1,2) and the results less accurate than the previous model.

As a final test, the best estimator from a non-bagged grid-searched LogRegCV, a base LogRegCV model, and a linear SVM was tested together on a standard count vectorizer with the same preprocessor. Again, all performed the similar.

Results

The lowest accuracy score for the training set was 93.53% with an AdaBoost Decision Tree with scikit-learn default parameters. 225 false negatives were classified out of a total of 882 and 100 false positives were classified out of total of 4,293. The highest was 99.96% with a LogRegCV (Cs = 10, cv = 3, max_iter = 10). One false negative was classified out of a total of 882 and one false positive was classified out of total of 4,293.

The lowest class 0 (non-boil order) precision, recall, and f1 score on the training set was with the boosted decision tree with 0.86, 0.74, and 0.80 respectfully. The highest was with the LogRegCV (Cs = 10, cv = 3, max_iter = 10) model with 0.9989 for class 0 precision, recall, and f1 score for the training set.

The lowest class 1 (boil-order) precision, recall, and f1 score on the training set was with 0.95, 0.97, 0.96 respectfully with the boosted decision tree. The highest was 0.9998 for precision, recall, and f1 score with the LogRegCV (Cs = 10, cv =3, max_iter = 10) model. These results do not include models that classified at baseline level nor does it include models that gave perfect accuracy, as this indicates overfit.

For the testing set, the lowest accuracy was 92.47% with the boosted decision tree. 129 false negatives were classified out of a total 434 and 63 false positives were classified out of a total of 2,115. The highest accuracy for the test set was 95.56% with a LogRegCV(Cs = 100, cv =

5, max_iter = 100). 74 false negatives were classified out of a total of 434 and 32 false positives were classified out of a total of 2115.

The lowest class 0 precision, recall, and f1 score on the testing set was with a KNN model with base parameters set by scikit learn. Precision was 0.90, recall was 0.67, and the f1 score was 0.77 for class 0. Across the models, however, a linear SVM with base parameters set by scikit learn had the lowest precision of 0.87 for class 0 and a bagged RFC with 15 bagged estimators and 15 trees per forest yielded the lowest recall of 0.66 for class 0. Three models performed equally well across all classification metrics of precision, recall, and f1 score. A bagged LogRegCV with 15 estimators and base parameters from scikit learn for the regression yielded a class 0 precision of 0.92, a recall of 0.79, and f1 score of 0.86. A boosted LogRegCV(Cs = 100, cv = 5, max_iter = 100) with two estimators for boosting yielded a class 0 precision of 0.91, a recall of 0.81, and a f1 score of 0.86. The same LogRegCV model without the boost yielded a class 0 precision of 0.90, a recall of 0.83, and a f1 score of 0.86. The highest precision for class 0 was 0.96 with the bagged RFC model with 15 estimators and 15 trees per forest. The highest class 0 recall was 0.85 with the linear SVM model with default scikit learn parameters.

The lowest class 1 precision, recall, and f1 score on the testing set was 0.94, 0.97, and 0.96 respectfully with the boosted decision tree. The two models with highest class 1 precision, recall, and f1 score was 0.97, 0.99, and 0.97 respectfully was with a LogRegCV(Cs = 100, cv = 5, max_iter = 100) and 0.96, 0.99, and 0.97 respectfully with a bagged LogRegCV with 15 estimators and scikit learn default parameters for the LogRegCV.

All model scores were validated over multiple stratified splits into training and testing sets. The models with lowest and highest values for all metrics were done with a standard count vectorizer preprocessed with the same cleaner as described in the Methods section.

For the final comparison between two LogRegCV models and a SVM model, a base LogRegCV was decided as the best model for potential deployment because it classified one more true positive for an adjusted LogRegCV (Cs = 10, cv = 5, max_iter = 10). Bagged LogRegCV did not give much better results over a non-bagged LogRegCV.

Discussion

Nearly all models outperformed the baseline model by nearly ten percentage points, with the maximum classification accuracy of 95.5%. The slight variations in model accuracy, recall, precision, and f1 score are miniscule. An increase in accuracy or other classification metric can be attainted through hyperparameter optimization but the work and time required may not be worth it if the models are performing with nearly 95% accuracy. Also, the randomness of the training/testing data split, despite stratification, may be more of a factor in the difference in accuracy and/or classification metrics between the models than the hyperparameters.

While the classification model does have a high accuracy in predicting classes obtained from using a LDA technique, caution must be exercised with these results. Articles about boil-orders may not be evident solely on the article headline. An example is the headline "Kutcher tells Council about water situation – Murray Ledger and Times." This article is about a boil-order but neither a computer nor a human would be able to infer that without reading the

article text. Unfortunately, precision web scrapping for article text across many different news sites is a challenging task due to unstructured HTML formats. Further work in the future should try the same process on article text.

Webs-scraping data also has its own challenges concerning how websites can permanently move or go down and articles can be deleted. Luckily, the HealthMap saves the raw HTML that it scrapes but this data was not accessible through the .csv. The author does not know if an API exists or if there is a way to access these stored HTMLs.

There may be a concern in the value of using unstructured LDA clustering to assign a "ground truth" to a dataset. This technique would not be useful if there was no domain knowledge about the articles in the dataset. However, the domain knowledge of the author, along with the validation process, shows that LDA can be a useful technique to separate a single class dataset into multiple classes given adequate domain understanding.

Conclusion

Transparent information on boil-orders are in need as shown in previous disasters with Moore Bend, Missouri, and Flint, Michigan. The flow of transparent information regarding water quality issues, including boil-orders, can be difficult because of the nature of private companies controlling the water supplies and disinfection protocols. Using non-traditional disease outbreak tools, like the HealthMap, can be useful in obtaining transparent information. With an adequate domain understanding and a technique like LDA, multiclass information can be gathered from a single class data source and then be used to build a predictive model for future articles.

References

1. Adler, P. (2017). 4-year Boil Water Order frustrates subdivision in Taney County. Retrieved December 15, 2017, from http://www.ky3.com/content/news/Ozarks-Town-spends-4-years-under-Boil-Water-Order-415694043.html. Updated 3/9/17