Tweet Collection Services
CS6604 Fall 2019 Final Report

Ola Karajeh
Department of Computer Science
Virginia Tech

Chidubem Arachie
Department of Computer Science
Virginia Tech

Edward Powell
Department of Computer Science
Virginia Tech

Eslam Hussein
Department of Computer Science
Virginia Tech

ABSTRACT

The proliferation of data on social media has driven the need for researchers to develop algorithms to filter and process this data into meaningful information. In this project, we consider the task of classifying tweets relative to some topic or event and labeling them as informational or non-informational, using the features in the tweets. We focus on two collections from different domains: a diabetes dataset in the health domain and a Heartbleed dataset in the security domain. We show the performance of our new method for classifying tweets in the different collections. We employ two approaches to generate features for our models: (1) a graph based feature representation and (2) a vector space model, e.g., with TF-IDF weighting or a word embedding. The representations generated are fed into different machine learning algorithms (Logistic Regression, Naïve Bayes and Decision Tree) to perform the classification task. We evaluate these approaches using metrics (accuracy, precision, recall, and F1-score) on a held out test dataset. Our results show that we can generalize our approach with tweets across different domains.

KEYWORDS

Twitter, Machine Learning, TDM, Vector Space Model, Graph Based Model, Word Embedding

ACM Reference format:
Ola Karajeh, Chidubem Arachie, Edward Powell, and Eslam Hussein. 2019. Tweet Collection Services CS6604 Fall 2019 Final Report. In *Proceedings of ACM Conference, Washington, DC, USA, July 2017 (Conference'17),* 11 pages.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Social media sites are considered an important source of data in the modern era [3]. This data comes in different forms (structured and unstructured) and often is about different: incidents, trends, events, impending circumstances, or reactions to hot topics. Timely collection and processing of social media information can help with: providing relevant information to authorities, deciphering public opinions about a topic, forecasting events, and gathering data for academic researchers. Our work is targeted towards the latter in that we build models that classify social media data for use by academic or industry practitioners. We process social media data and develop features for machine learning classifiers. From this, we can create new datasets that have less noise and can allow experts to work with the data more easily. We collect data from Twitter in the form of tweets, process it to generate meaningful representations, and then feed these representations into machine learning models that learn to classify the tweets as informational or non-informational relative to a specific topic. Our task is non-trivial as it addresses many challenges that researchers face when dealing with text data. These challenges are amplified when using short unstructured texts like tweets, hence the need for thorough processing of such a corpus. Moreover, Twitter datasets are usually enormous with at least 500 million tweets sent per day [16]. Thus, manual cleaning of a huge corpus by a human is very time-consuming, hence impractical and not scalable.

The first step in our pipeline is that we sample and annotate some tweets from our collection to be used as training data. We label these tweets as either informational or non-informational by following some annotation guidelines we define. The annotations provide labels used for training our machine learning models. Next, we process the labeled tweets, generating features for them via our graph based approach and also through TF-IDF scores and word embeddings. Lastly, we feed the features into our models to learn to predict informational and non-informational labels. In our experiments, we use standard metrics such as accuracy, precision, and recall to compare the performance of the different models. We analyze the results to show which feature approach works better for our classification task.

In the sections below, we present our literature review, methodology, and experiments. We highlight the work done so far, the detailed steps of our approach, and provide insights on our plans for future work.

2 LITERATURE REVIEW

Social media is not only used as a means of communicating social information but also to seek help, share useful information that would improve the well-being of others, and sometimes save lives in the event of disasters. Researchers have utilized information shared on social media for a variety of purposes. For example, in

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

[17] the authors used a Bayesian approach to classify tweets during Hurricane Sandy into informational or conversational tweets. They manually labeled 1, 086 tweets (139 informational and 943 conversational), then developed a set of 9 features that would identify such tweets based on a concept called the *formality* of the tweet. For example, if a tweet contains emojis (emotions) then it would be classified as conversational. On the other hand, the presence of a phone number reflects credibility, so the tweet is more likely to be informational. Although their approach sounds logical, it has many flaws; for example, a phone number could be for an advertisement of some product. Also, the presence of some emojis could indicate that the person might be stressed or grieving, as in the case of a disaster.

Another study [9] of tweets written during a disaster also used classes of informational and non-informational. The authors use a Convolutional Neural Network (CNN) with a fully connected layer for feature extraction and classification of tweets during Hurricane Harvey in 2017. Their proposed approach outperformed other baselines using different combinations of unigrams, bigrams, and trigrams in terms of accuracy, F1 score, precision, and recall. Other works [2, 6, 8, 12, 18] have also used social media data for different classification tasks in specific domains. Our work is similar to theirs but we aim to use a different approach.

Our graph based approach is inspired by the work in [4] where the authors construct a graph using words in the tweets, and then generate features using the PageRank and Hypertext Induced Topic Search (HITS) algorithms. They used graph features to train their models. Our approach differs from theirs in that we use different graph metrics (features, e.g., graph centrality) that better capture information about the tweets. Since we are not restricting ourselves to one approach we also use word embeddings as features to train our machine learning models. Word embedding models like Word2Vec [11], FastText [7], and context aware models like BERT [5] have been developed to map text into word vectors; they provide richer features for machine learning models. In our experiment, we use GloVe global vectors [15] to generate an embedding for the words in the tweets, and then we add the individual word vectors to obtain a sentence embedding for the full tweet. We give details in the following sub-sections.

2.1 Comparative Models

2.1.1 Graph Based Model-PHD. Blanco and Lioma [1] used graph based models for information retrieval. They note that "related words tend to form a network of connections that approximates the model humans build about a given context" [1]. Their work demonstrated usage of both directed and undirected graphs. Our proposed method utilizes a directed graph for its topical analysis with co-occurrence and the order of a pair of words (grammatical dependence) being taken into account.

Cordobes et al. [4] proposed a graph-based text representation which is used for topic classification. One basic idea is that tweets belonging to the same topic will have similar graph structures. This paper used mainly three graph measures: PageRank, Hyperlink-Induced Topic Search HITS (Hub and Authority), and Graph Density (GD). This approach is referred to as (GB-PHD). Although this solution solved the scalability issues in the previous solutions, it still has room for improvement in terms of accuracy. Our approach use various centrality metrics, and other different metrics such as the weights between edges.

Tweet 1	type 2 diabetes sucks
Tweet 2	diagnosed type 2 diabetes
Tweet 3	health benefit cinnamon

Table 1: Example Tweets

	benefit	cinnamon	diabetes	diagnosed	health	sucks	type	2
Tweet 1	0	0	1	0	0	1	1	1
Tweet 2	0	0	1	1	0	0	1	1
Tweet 3	1	1	0	0	1	0	0	0

Table 2: TDM Example

2.1.2 Term Document Matrix (TDM). TDM cells give the frequency of terms that occur in a collection of documents, in our case, tweets. The rows of the term document matrix represent the tweets while the columns represent the words in the corpus. Consider the example tweets from the diabetes dataset shown in Table 1. The resulting TDM is given in Table 2. Each row represents a tweet, while the columns are for the unique set of terms found in the set of tweets. The value of cell (n_i, m_j) describes the frequency of term m_j in tweet n_i.

A disadvantage of TDM is that if a term is added to the corpus, the entire matrix must be regenerated. This is costly and inefficient. Furthermore, documents with similar content but different ways to express similar ideas are not associated with similar representations, which is another major disadvantage of this method.

2.1.3 Word Embeddings. A word embedding is a form of word representation in which words with similar meaning tend to have similar representations. Word embeddings also can reduce the dimensionality for word representation in the corpus. Thus, we could say it generates a more efficient and expressive word representation by generating small vectors. Generally, a word embedding does a better job of taking into account the context and semantic similarity, improving upon the traditional Term Document Matrix. GloVe [15] is an embedding model available to use in our experiments. This model is an unsupervised learning approach for obtaining vector representations for words. GloVe Vectors are part of an open source project at Stanford that have a handful of pre-trained word vectors, one of which is for Twitter datasets. We will average the individual word vectors to obtain a sentence (tweet) embedding.

The next two sections highlight the work done so far; the appendix gives our plan for future work.

3 METHODOLOGY

Figure 1 shows the major workflow for the first stage of our project, which is classifying the tweets using graph features.

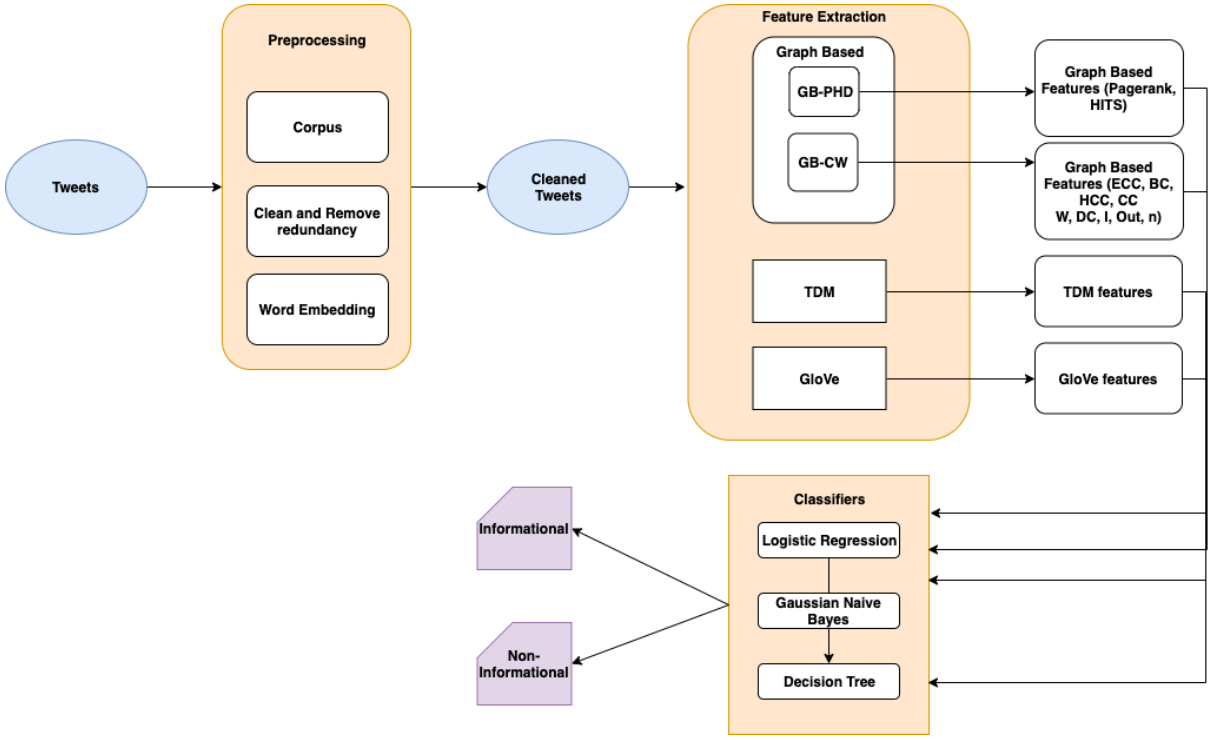


Figure 1: Basic Workflow

3.1 Pre-processing

The pre-processing steps consist of data collection, data annotation, data cleaning, and lastly word embedding. We explain the details for each of these steps below.

3.1.1 Data Collection. An earlier but ongoing study by Ola Karajeh concerns over twenty million tweets about "heart attack"; it was used to pilot the approach reported herein. We utilize two new datasets for our tasks. As with the "heart attack" study, one new collection is in the health domain, i.e., about diabetes. The other new dataset is in the security domain and is a collection of tweets about "Heartbleed," an infamous Internet security bug. As aforementioned, we utilize this dataset to see how our method works in a separate domain.

The tweets were collected using the Social Feed Manager platform, connected with Virginia Tech's Digital Library Research Laboratory. The diabetes dataset has about 12 million tweets while the Heartbleed collection had just over 197K tweets.

3.1.2 Data Annotation. For our supervised training of the machine learning models, we require labeled data. As such we randomly sample 6000 tweets from each dataset for labeling. Each of the four co-authors was responsible for labeling 1500 tweets for each of the two new datasets. Thus, each team member labeled 3000 tweets for a binary classification of either informational or non-informational. The labeled tweets will be split into different segments, being used for training, validation, and testing. To ensure consensus on what constitutes an informational tweet, a random sample of 50 tweets was generated for each of the two datasets, with each team member labeling both of the samples. After all 100 of these tweets were labeled, we discussed the conflicts in the labels and derived a strategy to ensure consistent labeling for the tweets.

Heuristic 1	"#heartbleed affected YouTube, but not @MediaGoblin. 8 days remain to help yourself free your own media!"
Heuristic 2	"#Heartbleed isn't the only #vulnerability of #OpenSSL.. What all are its #weaknesses?"
Heuristic 3	"RT @jolizevette: This #heartbleed bug has made me realize I have WAY too many accounts. I feel like I've spent all day changing passwords."

Table 3: Heartbleed Labeling

Heartbleed: This dataset aimed for tweets about the Heartbleed Internet virus. The Heartbleed bug was created accidentally by Robin Seggelmann in 2012 and revealed to the public in 2014, causing a brief panic wave across the Internet and the world. For this dataset, Table 3 outlines the three major heuristics with examples. These heuristics are utilized to classify tweets as informational: (H1) tweets related to the companies or entities that were affected, (H2) a description or details about the bug, and (H3) the responses of people or entities towards the bug.

Diabetes: This dataset contained tweets that revolved around diabetes, a disease related to blood glucose and sugar levels. For

Heuristic 1	"RT @11streethealth:RT @DrexelMedNews: Symptoms include blurry vision, excess thirst, fatigue, urinating often and weight loss. #Diabetes"
Heuristic 2	"Moral of the story: Don't eat too much sugar if u don't want to get diabetes injection like Hansel got"
Heuristic 3	"#SM Spicy Solution? Cinnamon May Help Diabetes Patients: Cinnamon may improve blood sugar levels for people"
Heuristic 4	"Gestational Diabetes Tied to 7-fold Increase in Sleep Apnea Risk"
Heuristic 5	"Foiegras triggers diseases that people are predisposed to like Alzheimers,CJD and diabetes"

Table 4: Diabetes Labeling

this dataset, there were five main heuristics utilized to classify a tweet as informational. The tweets in Table 4 correspond to the following heuristics: (H1) tweet content about symptoms; (H2) tweet content related to causes; (H3) a tweet referencing medication, vaccines, or remedies; (H4) tweet content about the effects, risks, or consequences of diabetes; and (H5) general scientific information related to diabetes. Self-mention and mention of others having the disease were also labeled as informational if they were combined with one of the five heuristics.

Reflection on labeling: Some tweets were full of sarcasm or malevolent content; however, some of these were still classified as informational because they met a heuristic. In general, sarcastic tweets are difficult to deal with because the fact that a tweet is sarcastic may not necessarily indicate that it is non-informational. At times, sarcasm requires inference to understand the author's intent. Accordingly, we did not consider sarcasm, and stuck close to the heuristics developed for each dataset. Inferring tweets is not scalable and is too subjective to do during manual labeling. If we were to do this during the manual labeling, without a reliable method to detect sarcastic tweets in a dataset, this would most likely lead to a decrease in accuracy.

Originality was not required because many users can receive the same idea or data from multiple Twitter users just like people receive their news from multiple news channels, papers, and blogs. Retweets were also considered and many tweets in the content had overlapping content. If a tweet and its retweets are properly labeled by the learning model, this is great for accuracy as all the retweets were in theory labeled correctly. On the other hand, a mislabeled tweet can really hurt results, especially if a specific retweet is prevalent throughout a dataset. One way to fix this potential issue and improve the results would be to get a unique set of all the retweets and manually label those in addition to the random sample. One major limitation during this phase is that misinformation or false statements are one important aspect that was not taken into account with this project. Furthermore, those who do the manual labelling may not have enough knowledge to decipher misinformation, so it is possible for such content to leak into the final dataset, which is intended for the experts in a domain. Therefore, it is important to point out that for this experiment, any tweet that gave the appearance of being informational based on the developed heuristics was categorized as such.

3.1.3 Data Cleaning. For the cleaning process, we used Stanford's Natural Language Toolkit (NLTK), following standard practice. The tool is built in Python and is open source. The general idea of our processing script is for the tweets to undergo a process that extracts the valuable content. Below are the descriptions of the cleaning process and more details on its usefulness.

Removal of Punctuation and Hashtags: Certain characters in Twitter have unique meaning and weight in the social media realm. The hashtag sign, '#', is used before a relevant keyword in a tweet and can help categorize a tweet. Sometimes, it is also used to summarize or conclude a tweet. The at character, '@' is used to reference a handle name on Twitter. Removal of these characters is standard during pre-processing because these symbols do not typically add meaningful data nor provide greater context. Other punctuation symbols such as the following '.,"!*;' were also removed. It is important to note that the removal of certain punctuation such as periods in an acronym or name can alter the original meaning. One example would be the open source software "f.lux" that changes a computer's display based on the time of day. If the period were removed from this word (flux), then the meaning changes and can reference the more common definition of the flowing of something in and out.

Removal of RT: In the Twitter space, 'RT' is an acronym that stands for "retweet." This is when one Twitter user is tweeting the contents of another user's tweet verbatim, proliferating it to their own followers. For our purposes, we gain no information knowing if the tweet was "original" or not, so we remove the 'RT' which indicates this.

Removal of Stop-Words: In natural language processing, stop words are considered words that do not have much meaning. The following are examples of common stop words: "the," "a," "an," and "in." Removing words such as these save space and time for the feature representation model as they do not provide additional context to the models. Generally, this is standard during a pre-processing phase.

Removal of URLs: Some of the hyperlinks point to meaningful or informational pages. However, these were filtered out because we are looking at the tweet content itself as being informational or non-informational. As a result, many tweets with useful hyperlinks were classified as non-informational. Furthermore, the main purpose of the project was to focus on tweets as a standalone source of information rather than a source which points to another source. Future work may be able to account for this, but that was beyond the scope of this project.

Stemming: The stemming algorithm used is the Porter Algorithm which is a standard algorithm used for English stemming. The primary purpose of the stemming is to remove affixes and revert inflectional forms into a basic form. The algorithm has five phases of word reduction that are applied consecutively. Each phase has special rules and some of the phases use a metric that essentially counts the syllables to see if the rule can apply to a particular affix [10]. The result of this process is a stem, i.e., the essence of a word to which affixes can be attached.

Convert all letters to lowercase: The primary reason for converting all letters in a word to lowercase is to prevent imprecise counting. Without the conversion, the words "Free" and "free" would be counted separately. The placement of a word in a sentence or a user using all caps to provide an emphasis should not lead to a new word being counted. Lower-casing all letters alleviates this issue, allowing for a more genuine frequency count. It is important to note that lower-casing all letters might lead to some data being lost such as a company or entity name with familiar terms but using an uppercase letter or even all uppercase letters to depict it. One common example is if the relevant term is "Windows" referring to the operating system. If this is lower-cased, then it can lose its meaning and falsely appear to take on the meaning of "windows."

3.2 Feature Extraction

Now that we have pre-processed the datasets, the next step in our workflow is to feed the clean datasets into the graph models to generate features that we will use to train our machine learning models.

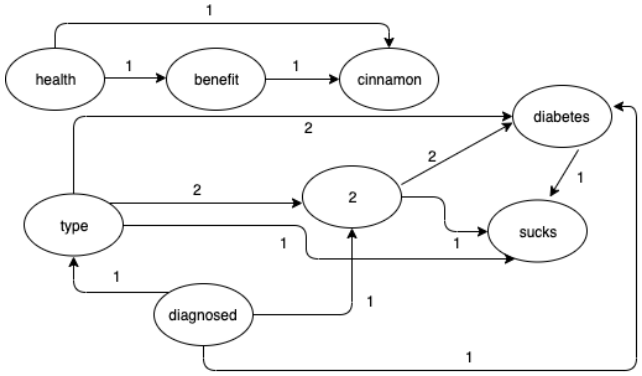


Figure 2: Graph Based Example

3.2.1 Graph-Based Text Representation - CW. Figure 2 shows the graph based example using the tweets from Table 1. The nine features that were used for our graph based model were: Number of nodes per tweet, Eccentricity (ECC), Betweenness Centrality, Harmonic Closeness Centrality, Closeness Centrality, Edge Weight, Node Degree, In Degree, and Out Degree. GB-CW utilizes the similarity of the graphs and these attributes to help classify the tweets by topic.

The number of nodes per tweet is the number of words contained in the tweet. For 'Tweet 3' from Table 1, the number of nodes depicted is 3. With Twitter data, it is important to acknowledge and take into account the sparsity problem of working with small documents [13].

The Edge Weight between two nodes, node v and node w, represents the co-occurence of the pair (v,w) in that order. In the context of our graphs, it indicates that a word, v, appeared x times before another word, w in a tweet. From Table 1, the edge weight between the term 'type' and '2' is two.

The In and Out Degrees consider the order of co-occurrences of words. The In Degree of a particular node specifies how many times a particular word has appeared after other words. Similarly, the Out Degree of a particular node depicts how many times a word appears before other words. From the example in Table 1, the out degree of the node 'type' is 2 and the in degree is 1.

The Node Degree is simply adding the In and Out degree values together. This helps to give an overall picture of the total flow from a particular word or node to other words in the network. Continuing this example, the node degree for the node 'type' from Table 1 is three.

$$ecc(v) = \frac{1}{\max_i d(v, x_i)} \; \forall i \in \{1, \ldots, n\} \tag{1}$$

Eccentricity is a metric that depicts the importance of a node by capturing the reciprocal of the maximum distance a node v has to other nodes in the network, where x_i is any other node in the network. The formula is shown in Equation 1 and the smaller that distance, the more central the node. In Table 1, the eccentricity for the node 'diagnosed' has a .25 value.

$$closeness(v) = \frac{n}{\sum_i d(v, x_i)} \; \forall i \in \{1, \ldots, n\} \tag{2}$$

The Closeness Centrality is depicted in Equation 2. We take the reciprocal of the sum of the distances from a single node, v, to all other nodes that are in the network. We use the normalized form which multiplies the reciprocal by the number of nodes n. Similar to eccentricity, a larger closeness value indicates that a node is more centralized.

$$harmonic(v) = \sum_{v \neq x} \frac{1}{d(v, x)} \tag{3}$$

The Harmonic Closeness Centrality is mostly intended for directed graphs and is similar to Equation 2. The main difference is that we sum up the reciprocal of the distances between the nodes in the graph.

$$betweeness(v) = \sum_{s \neq v \neq t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}} \tag{4}$$

Betweenness Centrality is the sum of the ratio of the shortest paths that pass through the node v to all the shortest paths in the graph, where σ_{st} is total number of shortest paths from node s to node t and $\sigma_{st}(v)$ is the number of those paths that pass through v.

A major aspect for developing the graph from the words in the tweets is reducing the number of dimensions representing the data in the tweet corpus. We achieve this by training a Word2Vec embedding model on our corpus, then replacing very similar word vectors with a base word. For example, the words dad, father, and daddy are related. Hence the cosine similarity of their word vectors reflects their closeness. This is similar to using a thesaurus and replacing every instance of dad and daddy in the corpus with father in order to reduce the space of words in our data corpus.

4 EXPERIMENTS

We used Scikit-learn [14] to implement the machine learning models we use in our experiments. The framework is open-sourced, coded in Python, and provides off the shelf models for use.

	GB-CW	TDM	Word Embedding	GB-PHD
Logistic Regression	0.802	0.831	0.825	0.794
Gaussian Naive Bayes	0.772	0.683	0.718	0.776
Decision Trees	0.709	0.822	0.777	0.693

Table 5: Accuracy of the different feature models trained on the Diabetes dataset

	GB-CW	TDM	Word Embedding	GB-PHD
Logistic Regression	0.558	0.648	0.620	0.429
Gaussian Naive Bayes	0.412	0.373	0.410	0.365
Decision Trees	0.412	0.571	0.460	0.282

Table 6: Precision of the different feature models trained on the diabetes dataset

	GB-CW	TDM	Word Embedding	GB-PHD
Logistic Regression	0.161	0.378	0.380	0.017
Gaussian Naive Bayes	0.261	0.806	0.920	0.127
Decision Trees	0.356	0.517	0.520	0.322

Table 7: Recall of the different feature models trained on the diabetes dataset

	GB-CW	TDM	Word Embedding	GB-PHD
Logistic Regression	0.250	0.477	0.470	0.030
Gaussian Naive Bayes	0.319	0.510	0.570	0.189
Decision Trees	0.333	0.540	0.490	0.301

Table 8: F1-score of the different feature models trained on the diabetes dataset

	GB-CW	TDM	Word Embedding	GB-PHD
Logistic Regression	0.738	0.769	0.755	0.735
Gaussian Naive Bayes	0.677	0.632	0.694	0.702
Decision Trees	0.657	0.730	0.651	0.626

Table 9: Accuracy of the different feature models trained on the Heartbleed dataset

	GB-CW	TDM	Word Embedding	GB-PHD
Logistic Regression	0.741	0.755	0.720	0.696
Gaussian Naive Bayes	0.623	0.535	0.590	0.651
Decision Trees	0.589	0.688	0.580	0.550

Table 10: Precision of the different feature models trained on the Heartbleed dataset

	GB-CW	TDM	Word Embedding	GB-PHD
Logistic Regression	0.573	0.662	0.670	0.649
Gaussian Naive Bayes	0.575	0.905	0.880	0.616
Decision Trees	0.589	0.643	0.590	0.567

Table 11: Recall of the different feature models trained on the Heartbleed dataset

	GB-CW	TDM	Word Embedding	GB-PHD
Logistic Regression	0.646	0.705	0.700	0.671
Gaussian Naive Bayes	0.598	0.672	0.710	0.633
Decision Trees	0.589	0.665	0.580	0.560

Table 12: F1-score of the different feature models trained on the Heartbleed dataset

4.1 Models

We next provide a short description of the models we used for our experiments.

4.1.1 Logistic regression. Logistic regression is a discriminative model that estimates the probability of a class given a data point. It uses a logistic function to estimate probabilities for the classes.

4.1.2 Naive Bayes. The Naive Bayes classifier arises from an evidence based theorem that essentially provides insights into how much the evidence for a given dataset can be trusted to give predictions. The model is based on Bayes theorem. Naive Bayes is a generative model but makes the assumption that features of a dataset are conditionally independent given the class. This simplifying assumption makes it easier to implement, and reduces the model complexity.

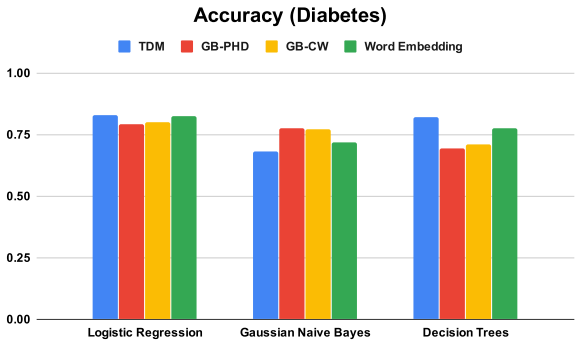
4.1.3 Decision Tree. Decision tree is a non-parametric discriminative model that iteratively uses rules called decision stumps to build a classifier. Decision trees can be used for both classification and regression.

4.2 Evaluation and Results

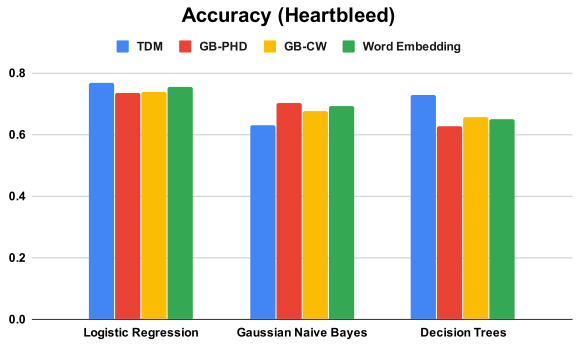
We train the three machine learning algorithms described above on each of the feature extraction models. We compare results from our graph features (GB-CW) against the comparative models: term document matrix (TDM), word embedding (GloVe vectors), and graph based model (GB-PHD). We present results in Tables 5-12 showing the accuracy, precision, recall, and F1-score of the different models on the diabetes and Heartbleed datasets. For each dataset, we train on 80% of the data and test on the remaining 20% of the data. The features for the graph models were scaled to have zero mean and unit variance.

For the diabetes dataset, as can be seen from Table 5, all the feature models achieve good accuracies on all the machine learning models with the TDM baseline having the highest accuracy on Logistic Regression overall. While the results are close, it is worth noting that accuracy is not a good measure of performance for this dataset as nearly 80% of the test data is not relevant and only 20% is relevant. Hence a naive classifier that predicts all zeros on the test data will have an accuracy of 80%. For this reason, we consider the precision, recall, and F1-scores as the more important metrics for evaluating the dataset. Looking at Table 8, we observe that word embedding feature model has a higher F1-score for Gaussian Naive Bayes algorithm meaning that its weighted precision and recall value is higher than competing models. Our graph based features outperform the baseline GB-PHD for all models but will need more tuning to match the results of word embedding and TDM.

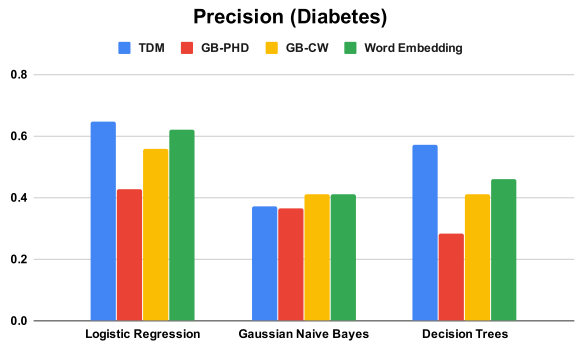
For the Heartbleed datasets, looking at Table 12, we see that word embedding still trains the best model on Gaussian Naive Bayes. However, other models perform almost as well and do better on this dataset than with the diabetes dataset. One reason for this



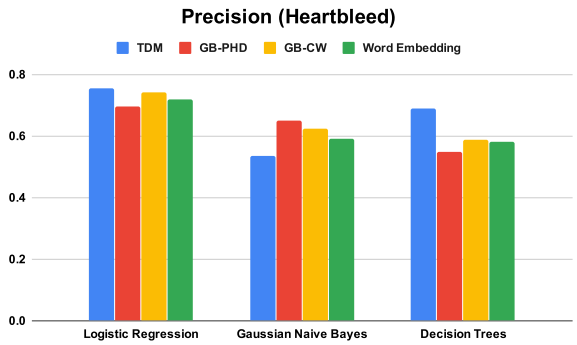
(a) Accuracy (Diabetes)



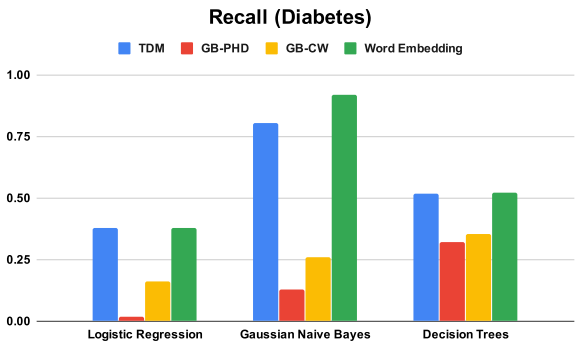
(b) Accuracy (Heartbleed)



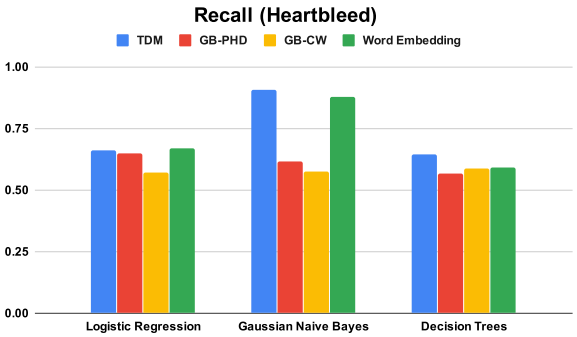
(c) Precision (Diabetes)



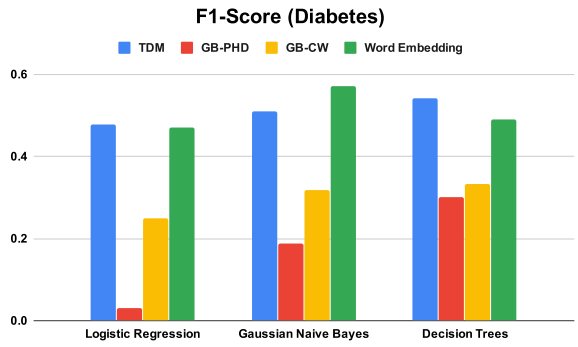
(d) Precision (Heartbleed)



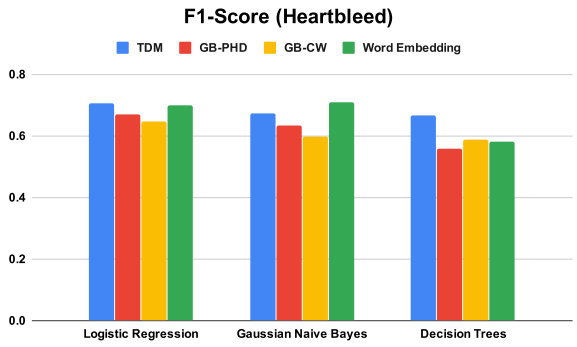
(e) Recall (Diabetes)



(f) Recall (Heartbleed)



(g) F1-Score (Diabetes)



(h) F1-Score (Heartbleed)

Figure 3: Evaluation metrics (Accuracy, Precision, Recall, and F1-Score) of three classifiers (Logistic Regression, Gaussian Naive Bayes and Decision Trees) trained and tested for both the diabetes and Heartbleed datasets using the four features sets (GB-CW, GB-PHD, TDM, and Word Embedding).

is that the Heartbleed dataset is less skewed in class distribution than the diabetes dataset.

While our graph based feature model didn't outperform some of the baseline models, we can see that it shows promising results, and with some modifications will match and possibly outperform the baseline models.

5 CONCLUSION

We have developed a model for classifying tweets. While we have shown results of testing our model on a small corpus of labeled data, the purpose of our model is to predict relevance to a topic for a larger unlabeled corpus. A user of our system will train our model on a small annotated corpus, validate its accuracy on a test data, tune the model to perform better, and then use it to make predictions on a larger corpus. The test results can be interpreted as confidence bounds on the machine learning algorithm. Furthermore, it is worth noting that when training machine learning algorithms with skewed classes, special provisions should be made for the model to accurately learn the skewness of the classes. This can be done by weighting the classes by their ratio in the dataset or by using a loss function that penalizes the dominant class more. While we haven't done any of the above, since we wanted to show a general approach for our methodology, users of our system should be mindful of this and incorporate it for their work.

In the future, we plan to go beyond tweet classification and extend our methodology to classify documents as to relevance to a corpus topic. This is particularly useful for digital libraries as we can classify documents in stored archives as to relevance to a category. Additionally, we plan to extend our methodology to go beyond binary classification of tweets, as relevant or not relevant, and instead enable multi-classification. A successful implementation of this will be vital to researchers who use digital libraries and are working in more complex settings, or in particular specializations.

REFERENCES

[1] Blanco, R., and Lioma, C. Graph-based term weighting for information retrieval. *Information retrieval 15*, 1 (2012), 54–92.

[2] Caragea, C., Silvescu, A., and Tapia, A. Identifying informative messages in disaster events using Convolutional Neural Networks. In *ISCRAM 2016 Conference Proceedings - 13th International Conference on Information Systems for Crisis Response and Management* (1 2016), P. Antunes, V. Banuls Silvera, J. Porto de Albuquerque, K. Moore, and A. Tapia, Eds., Proceedings of the International ISCRAM Conference, Information Systems for Crisis Response and Management, ISCRAM.

[3] Castillo, C. *Big crisis data: social media in disasters and time-critical situations*. Cambridge University Press, 2016.

[4] Cordobés, H., Anta, A. F., Chiroque, L. F., García, F. P., Redondo, T., and Santos, A. Graph-based techniques for topic classification of tweets in Spanish. *IJIMAI 2*, 5 (2014), 32–38.

[5] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[6] Giannakopoulos, K. Informative vs. non-informative short message detection in social networks. In *2017 3rd International Conference on Big Data Computing and Communications (BIGCOM)* (Chengdu, China, Aug 2017), pp. 165–171.

[7] Joulin, A., Grave, E., Bojanowski, P., Douze, M., Jégou, H., and Mikolov, T. FastText. zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651* (2016).

[8] Longhini, J., Rossi, C., Casetti, C., and Angaramo, F. A language-agnostic approach to exact informative tweets during emergency situations. In *2017 IEEE International Conference on Big Data (Big Data)* (Boston, MA, USA, Dec 2017), pp. 3739–3475.

[9] Madichetty, S., and Sridevi, M. Detecting informative tweets during disaster using deep neural networks. In *2019 11th International Conference on Communication Systems Networks (COMSNETS)* (Bengaluru, India, India, Jan 2019), pp. 709–713.

[10] Manning, C., Raghavan, P., and Schütze, H. Introduction to information retrieval. *Natural Language Engineering 16*, 1 (2010), 100–103.

[11] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds. Curran Associates, Inc., Stateline, Nevada, 2013, pp. 3111–3119.

[12] Nguyen, T. D., Al-Mannai, K., Joty, S. R., Sajjad, H., Imran, M., and Mitra, P. Rapid classification of crisis-related data on social networks using convolutional neural networks. *ArXiv abs/1608.03902* (2016).

[13] Nugroho, R., Yang, J., Zhong, Y., Paris, C., and Nepal, S. Deriving topics in Twitter by exploiting tweet interactions. In *2015 IEEE International Congress on Big Data* (New York, NY, USA, 2015), IEEE, pp. 87–94.

[14] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. Scikit-learn: Machine learning in Python. *Journal of machine learning research 12*, Oct (2011), 2825–2830.

[15] Pennington, J., Socher, R., and Manning, C. GloVe: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (Doha, Qatar, 2014), pp. 1532–1543.

[16] Project, R. T. S. Twitter usage statistics. https://www.internetlivestats.com/twitter-statistics/, 2019. Accessed: 2019-10-29.

[17] Truong, B., Caragea, C., Squicciarini, A., and Tapia, A. H. Identifying valuable information from Twitter during natural disasters. *Proceedings of the American Society for Information Science and Technology 51*, 1 (2014), 1–4.

[18] Zhang, S., Zhang, X., and Chan, J. Language-independent Twitter classification using character-based convolutional networks. In *Advanced Data Mining and Applications* (Cham, 2017), G. Cong, W.-C. Peng, W. E. Zhang, C. Li, and A. Sun, Eds., Springer International Publishing, pp. 413–425.

Appendices

A DEVELOPER NOTES

We have stored our relevant files on Code Ocean in 4 capsules that follow the project's workflow.[1] Our method is a supervised learning approach and takes manual labeling. With a new dataset, heuristics need to be developed to accurately train the classifier as best as possible. After the heuristics are developed a random sample should be taken and tweets can be manually classified as informational or non-informational to the particular dataset. It is best that experts do the labeling themselves, especially if they will be working with the informational dataset afterwards. The specifics for the sample size and number to label can vary by datasets, and should be analyzed on a case by case basis. Our pre-processing and cleaning code are available on capsule 1. The process is as follows.

(1) Select or use a Twitter dataset that is focused on a specific domain.
(2) Develop heuristics so experts or curators can accurately manually label a tweet for a dataset as informational or non-informational.
(3) Label a random sample of the tweets.
(4) Clean the dataset using our Python script and remove any possible redundancies.[2]
(5) A word embedding is determined, and graph-based features are computed.
(6) Tune the model for best results.
(7) After the feature extraction, vectors of feature values for the tweets are sent to various classifiers. These can be context-dependent and the user's choice. Our work utilizes Naive Bayes, Logistic Regression, and a Decision Tree for demonstrative purposes.

It is important to note that the four computing capsules all include further instructions and details that describe a particular module.

B ANNOTATION EXAMPLES

This section shows two tables of annotation examples from our dataset, where Table 21 shows 10 (5 informative/relevant and 5 non-informative/non-relevant) examples from the diabetes dataset and Table 22 shows 10 (5 informative/relevant and 5 non-informative/non-relevant) examples from the Heartbleed dataset.

C EXTENDED RESULTS

In this section we show some variation of our results using different k-cross values.

	GB-CW	TDM	GB-PHD
Logistic Regression	0.807	0.841	0.794
Gaussian Naive Bayes	0.749	0.683	0.775
Decision Trees	0.725	0.819	0.736

Table 13: Accuracy of the different feature models trained on the Diabetes dataset using 3-folds

	GB-CW	TDM	GB-PHD
Logistic Regression	0.586	0.685	0.429
Gaussian Naive Bayes	0.409	0.364	0.359
Decision Trees	0.333	0.57	0.353

Table 14: Precision of the different feature models trained on the Diabetes dataset using 3-folds

	GB-CW	TDM	GB-PHD
Logistic Regression	0.189	0.411	0.017
Gaussian Naive Bayes	0.511	0.733	0.128
Decision Trees	0.344	0.478	0.35

Table 15: Recall of the different feature models trained on the Diabetes dataset using 3-folds

	GB-CW	TDM	GB-PHD
Logistic Regression	0.286	0.514	0.032
Gaussian Naive Bayes	0.454	0.486	0.189
Decision Trees	0.339	0.52	0.352

Table 16: F1-score of the different feature models trained on the Diabetes dataset using 3-folds

	GB-CW	TDM	GB-PHD
Logistic Regression	0.802	0.845	0.794
Gaussian Naive Bayes	0.768	0.685	0.778
Decision Trees	0.722	0.815	0.707

Table 17: Accuracy of the different feature models trained on the Diabetes dataset using 5-folds

	GB-CW	TDM	GB-PHD
Logistic Regression	0.568	0.7	0.429
Gaussian Naive Bayes	0.427	0.368	0.373
Decision Trees	0.326	0.559	0.309

Table 18: Precision of the different feature models trained on the Diabetes dataset using 5-folds

	GB-CW	TDM	GB-PHD
Logistic Regression	0.139	0.428	0.017
Gaussian Naive Bayes	0.389	0.75	0.122
Decision Trees	0.339	0.444	0.35

Table 19: Recall of the different feature models trained on the Diabetes dataset using 5-folds

	GB-CW	TDM	GB-PHD
Logistic Regression	0.223	0.531	0.032
Gaussian Naive Bayes	0.407	0.494	0.184
Decision Trees	0.332	0.495	0.328

Table 20: F1-score of the different feature models trained on the Diabetes dataset using 5-folds

[1] 1. Cleaning and Stemming Capsule: https://codeocean.com/capsule/7766507/tree
2. Graph Representation Capsule: https://codeocean.com/capsule/2325826/tree
3. Extracting Graph Features Capsule: https://codeocean.com/capsule/7522921/tree
4. TDM Capsule: https://codeocean.com/capsule/5162588/tree

[2] Retweets are one common way redundancies arise for datasets. The cleaning script removes the 'RT' keyword so it looks like the original tweet, containing just the cleaned text.

Tweet	Class
After diabetes during pregnancy, healthy diet linked to reduced type 2 ... http://t.co/TOrAU2Db	Informative
Crusty foods may worsen heart problems associated with diabetes http://t.co/0FOUdpRw	Informative
RT @UberFacts: One alcoholic drink a day can reduce your risk of type 2 diabetes by up to 30 percent.	Informative
Study: Obesity doubles risk of gestational diabetes - OCRegister: Study: Obesity doubles risk of gestational dia... http://t.co/Aoo2NGpA	Informative
Type 2 diabetes hits hardest in communities of color. Learn more. http://t.co/VVt78C3a	Informative
Diabetes Fatigue: 6 Steps to Take http://t.co/KyFU2cE1	Non-Informative
Obesity #america #diabetes #scooters http://t.co/35U5SUFP	Non-Informative
@missebby16 go get tested for diabetes.	Non-Informative
Le diagnosticaron diabetes... y vendi?Ÿ?? su flauta dulce. Jaj?Ÿ?ja	Non-Informative
@daCPlanet I read that fast and thought you said "I'm sick of these diabetes"	Non-Informative

Table 21: 10 annotation examples from the diabetes dataset

Tweet	Class
@EFF so #Heartbleed exploits #OpenSSL 1.0.1 & 1.0.2-beta releases if I have the mod version.9.8 but not really using it am I clear?	Informative
Cisco products affected by #heartbleed > http://t.co/YcNuk8nSi1	Informative
Heartbleed Hit List - areas that may be affected and passwords you want to change: https://t.co/FKkpAT0qnk #heartbleed #HeartbleedVirus	Informative
We are #Heartbleed patched, our engineers rock. Proud of @shapeways for being so transparent: http://t.co/vz7Hzxvidn http://t.co/vPBiQguzaa	Informative
A useful list of the sites that have patched the Heartbleed bug.. http://t.co/bL300410Hq #security #heartbleed	Informative
Now is not the time to change passwords, in case you're planning to. #Heartbleed	Non-Informative
come vendors n pitchmen, please heed the call donÃ¢â‚¬â„¢t sell in the doorway donÃ¢â‚¬â„¢t flog up the hall #Heartbleed	Non-Informative
#Heartbleed = pretty much how I felt throughout middle school. Why do computer bugs get such poignant names?	Non-Informative
RT @SecurityHumor: This #heartbleed #OpenSSL bug walks into a bar. Repeatedly. And walks out with everything.	Non-Informative
RT @SZ: #Heartbleed-SicherheitslÃƒÂ¼cke: Bei welchen Diensten Sie jetzt Ihr Passwort ÃƒÂ¤ndern sollten. http://t.co/nxbX588I5V	Non-Informative

Table 22: 10 annotation examples from the Heartbleed dataset

	GB-CW	TDM	GB-PHD
Logistic Regression	0.732	0.745	0.741
Gaussian Naive Bayes	0.678	0.647	0.703
Decision Trees	0.665	0.725	0.64

Table 23: Accuracy of the different feature models trained on the Heartbleed dataset using 3-folds

	GB-CW	TDM	GB-PHD
Logistic Regression	0.731	0.73	0.707
Gaussian Naive Bayes	0.621	0.549	0.653
Decision Trees	0.596	0.673	0.563

Table 24: Precision of the different feature models trained on the Heartbleed dataset using 3-folds

	GB-CW	TDM	GB-PHD
Logistic Regression	0.567	0.619	0.646
Gaussian Naive Bayes	0.589	0.872	0.616
Decision Trees	0.62	0.662	0.608

Table 25: Recall of the different feature models trained on the Heartbleed dataset using 3-folds

	GB-CW	TDM	GB-PHD
Logistic Regression	0.639	0.67	0.675
Gaussian Naive Bayes	0.605	0.674	0.634
Decision Trees	0.604	0.668	0.585

Table 26: F1-score of the different feature models trained on the Heartbleed dataset using 3-folds

	GB-CW	TDM	GB-PHD
Logistic Regression	0.738	0.768	0.735
Gaussian Naive Bayes	0.678	0.636	0.692
Decision Trees	0.658	0.74	0.652

Table 27: Accuracy of the different feature models trained on the Heartbleed dataset using 5-folds

	GB-CW	TDM	GB-PHD
Logistic Regression	0.739	0.752	0.696
Gaussian Naive Bayes	0.627	0.539	0.644
Decision Trees	0.592	0.713	0.58

Table 28: Precision of the different feature models trained on the Heartbleed dataset using 5-folds

	GB-CW	TDM	GB-PHD
Logistic Regression	0.575	0.662	0.649
Gaussian Naive Bayes	0.567	0.882	0.586
Decision Trees	0.581	0.629	0.605

Table 29: Recall of the different feature models trained on the Heartbleed dataset using 5-folds

	GB-CW	TDM	GB-PHD
Logistic Regression	0.647	0.704	0.671
Gaussian Naive Bayes	0.596	0.669	0.613
Decision Trees	0.586	0.669	0.592

Table 30: F1-score of the different feature models trained on the Heartbleed dataset using 5-folds