Haterade Detector

Yael Romero, Quinn Favilla, Linda L. Li

Abstract

The purpose of this project is to detect abusive speech in discussion comments from Wikipedia. The dataset used for this project was created by Ellery Wulczyn, Nithum Thain, and Lucas Dixon, and contained approximately 100,000 annotated discussion comments. Each comment in the dataset was annotated by crowd-workers to determine if it contained a personal attack. The problem of labeling text as abusive or not is a text classification problem, and using a Support Vector Machine solves this problem. A Support Vector Machine is a discriminative classifier that separates two categories by a linear function called a hyperplane. Since the training data was labeled as 1 if an attack was present and 0 if not, feature words were extracted from the text and then used by the Support Vector Machine algorithm. The motivation behind this project is to build a model that can automatically flag comments that violate a conduct code by detecting whether it contains certain features. The idea was inspired by an implementation of sentiment analysis on Twitter comments by Ravikiran Janardhana.

Introduction

In many real life situations, the assignment of an object to a category based on specific characteristics is desired. For example, based on multiple medical tests, a doctor wants to be able to conclude if someone has a certain disease or not. In this case, the goal is to detect whether a comment contains abusive speech or not. Situations like these are considered classification problems. SVM is one of the most popular classifying methods, often times used in applications for text classification, speech recognition, facial expression recognition, and many others. Support Vector Machines work by mapping data to a high dimensional feature space so that data points can be categorized, even when data is not linearly separable. It works on the principle of fitting a boundary to a region of points that are all alike, meaning, they belong to the same class. They focus on the points that are most difficult to tell apart, whereas other classifiers focus on all of the points. Once the boundary is fitted on the training data, each new point in the test data that needs to be classified is checked to see whether or not they lie inside the boundary. Support vectors help identify and set the boundary. Each data point/observation is a vector, which is, a row of data that contains numbers for specific attributes. In this project, the comments are the vectors, which are labeled with either a 1, meaning the comment contains abusive speech, or a 0. When dealing with text classification, it's normal to have a large feature set. In this project,

a feature set of almost 10,000 words that were both positive and negative was used. SVMs are equipped to handle a large feature set since they use overfitting protection. This is one of the major advantages of using an SVM. To summarize the implementation of classifying speech as either abusive or nonabusive using SVM, a dataset with comments that were annotated with a 1 or 0 was acquired first. A clear gap/boundary line which can be referred to as a hyperplane was created through the process of separating these data points by category. There can be multiple hyperplanes, but the one that cleanly separates the data the best is the best hyperplane. Additionally, a boundary line that ensures that the average geometric distance between the two classes is maximized is even better. This distance can be referred to as a margin, which the SVM tries to maximize. New examples can then be mapped based on which side of the gap they fall on.

SVM Versus Other Classification Methods:

Presently, SVM is considered one of the more popular text classification methods compared to Decision Tree Method, KNN method, and Naive Bayesian method. "Decision Tree method is a process that using information gain in the information theory to find the properties of the field with the greatest amount of information in the sample database to build a decision tree node, and build a branch of a tree according to the properties of different values of the field; and then repeat building the next nodes and branches for each branch(Liu, Liu, Lv, Shi)." The Decision Tree method is a fast and accurate classifier, but there are unresolved issues such as vacancies in data processing and the discretization of continuous attributes. "KNN method is a theoretically more mature method. Suppose each class contains several sample data, and each data has a unique type of data standard. KNN is to obtain the k-nearest sample data from the data to be classified by calculating the distance from each sample data to the data to be classified, and which type of the k sample data occupies the majority, then the data to be classified belongs to that category(Liu, Liu, Lv, Shi)." However, due to the large number of calculations, the efficiency of this method is reduced when training samples have a large amount of attributes to be classified. "Naive Bayesian method is a kind of learning method based on Bayesian theorem, and it can be used to predict the possibility of class membership, and it gives the probability of the text belonging to a particular class. According to forecast results, the sample is assigned to the highest probability of category when classified. In theory, the application premise of naive Bayesian classifier is that the sample attribute value is independent of the classification properties of the sample, but its attribute independence assumption affects its classification performance. (Liu, Liu, Lv, Shi)." The SVM method is suitable for large test samples, as mentioned before. Its algorithm is based on the "structural risk minimization principle (Liu, Liu, Lv, Shi)." The original data is first compressed to a support vector set, then new knowledge is gained when using this subset and the rules decided by the support

vectors. Overall, SVM is a great classifier — it has superior performance and supports a wide range of applications.

Methods

Introduction to Supervised Learning: Supervised models are concerned with a set of rules for each specific case. Unlike unsupervised learning models, supervised models require expansive training data so it can develop a pattern-like recognition for any input data. For example, if the subject were to be extracted from a sentence such as "She was a great leader", regular expressions could be used to identify that the subject is at the beginning of the sentence. In the case of supervised learning, in order for the model to correctly identify subjects, all input sentences should have subjects at the beginning. Figure 1 shows the concept of supervised learning.

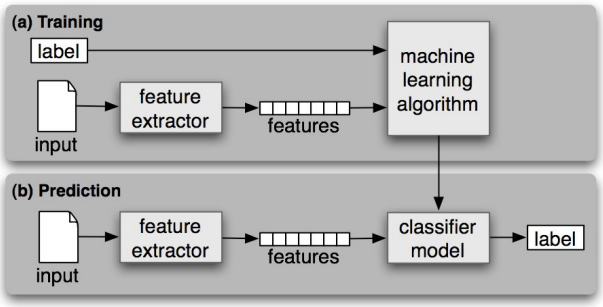

Figure 1
Source: NLTK.org

In the context of this project, a label is generated to determine if it contains an "attack" or not. Then, using a feature extractor, certain words are extracted from the training data and labeled as words that suggest abusive speech. The array of feature words are fed into the Support Vector Machine and then used to generate predictions on test data (the organization of our train and test data will be discussed in the later sections). Figure 2 represents a more concrete illustration of the Support Vector Machine.

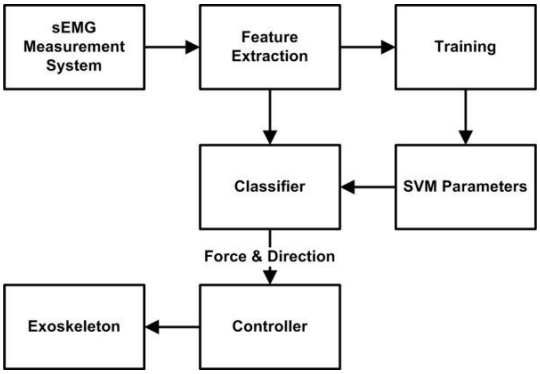

Figure 2
Source: Science Open

Evaluation Setup:

The dataset used for this project contained over 100,000 labeled discussion comments from English Wikipedia. Each comment was annotated by multiple (7-10) crowd-workers with either a 0 or a 1 to indicate whether they believed the comment contained an attack, or abusive speech. The discussion comments were labeled on specific forms of attack as well, such as whether or not there was an attack directed at a third party, or the recipient of the comment, etc. These extra labels were not used in the implementation, although it is important to note that if a crowd-worker labeled a comment with a 1 for at least one of the subsections related to attack, the label for whether or not the comment contained an attack was 1 as well.

Three files composed the dataset: one that contained the comments with a unique ID, and other properties such as the year the comment was posted, whether the user was logged in, a tag with domain values {train, dev, test} and more. Another file contained the unique ID of the comment along with the labels corresponding to each subsection of attack and finally the label corresponding to whether or not the comment contained any form of attack. The third file contained demographic data for each crowd-worker that labeled the comments. This third file was not used in the implementation of this project.

The first step in building the model was to join the files that contained the comments and their labels. However, each comment was labeled by multiple crowd-workers so first one of these labels needed to be chosen per comment. In order to obtain the most training data for abusive speech, if one of the crowd-workers detected an attack in a comment, that label was used for training. That is, if a 1 existed as a label for a given comment, the 1 label was kept and the rest were ignored. If none of the crowd-workers labeled a comment with a 1, then the comment was labeled with a 0. After filtering out the extra labels for each comment, the comments and their labels were joined on their unique ID. Then, since the original dataset contained over 100,000 comments, a

random sample of 1,100 comments of the ones labeled as “train” by the creators were used as training data. This random sample was obtained by using a random number generator in Excel. Another random sample of 1,100 comments of the ones labeled as “test” by the creators were used as test data.

Once the training data was obtained, each comment was preprocessed. The original dataset included tokens such as NEWLINE_TOKEN and TAB_TOKEN, as well as miscellaneous symbols such as colons, equal signs, \n, and quotation marks mapped to `. The preprocessing step consisted of converting the comment to all lowercase, removing the tokens, removing trailing and and leading spaces, removing extra spaces within the comment, and mapping quotation marks back to “ and ‘. In addition, any url found in a comment was mapped to the word URL.

The next step was to obtain a feature vector consisting of individual words (the unigram approach). Each comment was broken into its own feature vector. In this implementation, extra processing was done in order to obtain a robust set of features. For example, for each word in the sentence, if a pattern was repeated more than twice, the pattern was replaced with one occurrence of the pattern. That is, if a comment contained the word “heyyyyy,” this was mapped simply to “hey.” In addition, punctuation was removed from each of the words so as to not have multiple of the same word followed by a punctuation mark. Lastly, some words were filtered out based on their inability to indicate abusive speech or not. These words contained words such as “the,” “in”, “between.”

Each feature vector obtained from the comments in the training data was joined to create one feature vector to build the model. The idea behind using this feature vector to build the model is that for each comment, a pattern of ones and zeros (zero if a given feature word is not present or one if it is) is obtained and these patterns are how the model learns to classify new comments.

The libsvm library, implemented by Chih-Chung Chang and Chih-Jen Lin was used in this project to instantiate SVM. First, the pattern of ones and zeros for each comment (as discussed in the previous paragraph) was calculated and placed in a map. That is, each word in the feature vector was labeled as present or not present in each comment. This map was then used to train the SVM. Once the classifier was obtained from the SVM, the next step was to test it on a test set of comments.

As previously mentioned, the test set of comments was a random sample of the original set of comments labeled as “test” comments by the creators of the dataset. For each of these comments, a map was also created using the feature vector obtained from the training set. Then, using this map and the classifier, the SVM predicted labels for the test comments.

Results & Analysis

The labels predicted by the SVM were compared to the actual labels given by the crowd-workers to obtain an accuracy of 62.909%. This could be for multiple reasons, beginning with the size of the training set. Since language is complex and structure varies from person to person, a set size of 1,100 comments may have been too small to encapsulate the variety. Perhaps better results were to be obtained by using the complete dataset of over 100,000 comments. Since a randomized sample of the dataset was used, it is quite possible that the random data chosen was not as consistent as intended. Or, maybe there was not enough variation in the data. However, SVMs do not perform well on highly skewed data, meaning, training data sets in which the number of samples that fall in one of the classes far outnumber those that are a member of the other class.

Another reason why the accuracy could have been low is because the unigram approach was used to create features that were used by the SVM. This favors the detection of abusive speech when curse words are used, for example, because there is little to no ambiguity when these words are used. That is, a sentence that uses a curse word can usually be labeled as abusive. A bigram approach to extracting features could have been more effective in detecting abusive language directed towards someone. An example of this could be the difference between “hate” and “hate you.” Although the presence of “hate” alone may not indicate abuse, “hate you” does.

The system built works well with predicting the presence of abuse if the comment contained a curse word. On the other hand, the system does not work as well with predicting the presence of abuse if the comment contained abuse through potentially sarcastic tone. An example of this was found in the test data. A crowd-worker labeled the following comment as “containing an attack”: “It's no surprise. This is wikipedia, they glorify fascist dictators here - it's the dream of every WikiAdmin to become one someday and to act like it here until they make it elsewhere!” However, the model does not label this comment as abusive. This may have occurred because the features were not relevant enough to sarcasm, so naturally, the model would not perform well with sarcastic comments.

In order to obtain better results, a larger set of data with less features should be used. This is because SVMs are not efficient if the number of features are very large in number compared to the training samples. The number of features extracted in this implementation outweighed the number of comments in the dataset, but this was learned afterwards, so this could also be a reason as to why the accuracy was not as high as was hoped.

We have learned that the more data used, the more accurate our results will be. SVMs work better when you have plenty of samples for each class, but if you only have a few samples for some classes then KNN might work better than SVM. “KNN can learn very

quickly a frontier while SVM takes a little longer but learns better more robust frontiers because of the ability to maximize margin (Quora)."

Conclusion

With an accuracy of only 62.91 percent, a larger set of training data may help encompass more variety and size in our feature set. Using bigrams in the feature set may have also positively affected the accuracy of the detection of abusive speech. For example, the unigram word "kill" might not be deemed as a member of the feature set because it is just a vague verb, but if "kill you" was the bigram, then it is more evident that it is a form of abusive speech. Lastly, using a smaller number of features compared to the number of observations could have also improved accuracy. In summation, we learned that feature sets are extremely important in the SVM algorithm and also the format of how we choose our features can help determine the accuracy of running a set of test data against the training.

References

Deshpande, Bala. "When Do Support Vector Machines Trump Other Classification Methods." Analytics Made Accessible: For Small and Medium Business and beyond. Simafore, 28 Jan. 2013. Web. 01 May 2017.

Janardhana, Ravikiran. Twitter Sentiment Analyzer, (2014), GitHub repository, https://github.com/ravikiranj/twitter-sentiment-analyzer

Liu, Zhijie, Xueqiang Lv, Kun Liu, and Shuicai Shi. "Study on SVM Compared with the Other Text Classification Methods." 2010 Second International Workshop on Education Technology and Computer Science (2010): 219-22. Web.

Quora. N.p., 2013. Web. 1 May 2017. <https://www.quora.com/For-what-kind-of-classification-problems-is-SVM-a-bad-approach>.