Programming assignment 3: Stance classification

Eric Johansson - Max Sonnelid
johaeric@student.chalmers.se - sonnelid@student.chalmers.se

Abstract

The purpose of this assignment was to learn more about stance classification. This was done by building a classifier with identifies positive or negative connotations of comments on social media. The data was pre-processed and different learning algorithms were evaluated in order to yield the highest performance. Moreover, the hyperparameters of the Naive Bayes Classifier were tweaked using grid search, since that classifier were the most accurate. Finally the trained classifier were tested on a new data set and yielded an accuracy of 85%,

Introduction

Since the initial outbreak of COVID-19 in China in 2019, there has been an intense debate regarding how to prevent further spread and, eventually how to get rid of the virus. As always, introducing a new vaccine is a common mentioned solution. However, during the last year, the vaccine producers have been greeted with a lot of skepticism. A lot of people sees the vaccine as an attempt from the governments to control their people, or as dangerous due to lack of confidence in doctors and scientists (Willsher 2021).

This report aims to develop further knowledge about what opinion people writing online platforms (e.g, Youtube and Facebook) have in the vaccine discussion. To do this, a large data set is used, containing comments that expresses a pro-vaccination or anti-vaccination stance. The data is then slightly tweaked in order to implement a text classifier to predict which sentiment additional comments contains. The used data is collected and annotated by students that are enrolled in the course ”Applied Machine Learning”.

Pre-processing of data

There are three data sets used in this task: a small training data set with 975 entries, a large training data set with 8788 entries and a test data set with 392 entries.

All these data set consists by one column of comments of different lengths expressing either a pro-vaccination or anti-vaccination stance and one binary column where 1 represents pro-vaccination stance and 0 anti-vaccination stance. However, the large training data set included several annotations of either 1 and 0 for the same comment as this data was converted by the method `conv_annotation` into one single annotation for each comment by taking the most frequent annotation for a particular comment.

Firstly, the training data is divided into two new, randomly selected training and test data sets.

Secondly, the data is run in a so called `TfidfVectorizer`, which creates a vocabulary from the input documents and assign a ”TF - IDF” value to each word in the vocabulary. The ”TF - IDF” values is a kind of word frequency score, which aims to highlight words that are frequent in specific documents, but is not frequent in all document. In this case in can be expected that a word as ”being” is assigned a low score, while a word as ”fantastic” is assigned a high score. The choice of hyperparameters for the `TfidfVectorizer` will be explained further in the next section.

It was attempted to run the data through a pipeline of different regular expressions as well as a lemmatizer. In this pipeline, all the special characters are removed, single characters from the start are removed, multiple spaces are substituted with single spaces and eventually all words are run through the WordNetLemmatizer from the Natural Language Toolkit library, which removes inflections from all words. However, after running the data through this pipeline the accuracy score decreased after making predictions by all of the models Random Forest Classifier, Multinomial Naive Bayes and Stochastic Gradient. The decrease in accuracy score also happened when it was attempted to process the data with only individual regular expressions. Therefore, it was chosen not to continue experimenting with this kind of pre-processing, even though it theoretically could be able to improve the predictions. Instead, the original, unprocessed data was used as input to the `TfidfVectorizer`.

Selection of learning algorithm

Several types of classifiers that the report authors have previous experience of using were used as a start for finding the best prediction model for this task. Those classifiers were a Random Forest classifier, Stochastic Gradient Descent classifier, Support Vector Machine as well as a Multinomial Naive Bayes classifier. A more systematic research about suitable prediction models for binary text classification could have been done, but as all of these four classifiers gave rather high and similar accuracy scores, the focus was instead directed towards optimizing these four models.

In order to optimize the hyperparameters of the models, the method GridSearchCV from the Scikit-learn library was used. By selecting a range of possible values for chosen hyperparameters of a model, the method then runs through all possible combinations of hyperparameters and is able to return the set of hyperparameters that optimizes the accuracy score. As this process is rather time-consuming for a bigger data set, GridSearchCV was only run on the small training data set.

Model	Accuracy
Stochastic Gradient Descent	0.759
Multinomial Naive Bayes	0.769
Random Forest	0.692
Support Vector Machine	0.764

Table 1: Accuracy scores for all four models with optimized hyperparameters.

By pre-processing the data and tuning the hyperparameters by GridSearchCV, it was eventually possible to reach a maximal accuracy score of 0.769 with a Multinomial Naive Bayes classifier with hyperparameters described in the following subsection.

Hyperparameters of Multinomial Naive Bayes

The first hyperparameter that was included in the grid search, and later tweaked was `alpha` which is a way of introducing Laplace smoothing. Setting the parameter to 0 means that no smoothing is used, and more smoothing is attained with a higher `alpha` (Scikit-learn 2020a). The grid search proposed a value of 0.5.

The second parameter that was included in the grid search was `fit_prior`, which (if set to true) learns the probabilities for all the priors. If it is set to false, then a uniform prior will be used. (Scikit-learn 2020a)

Hyperparameters of the TfidfVectorizer

Compared to Multinomial Naive bByes, there exist a wide variety of hyperparameters for the TfidfVectorizer that can be tweaked until perfection. Since all permutations of the different parameters will be run when performing a grid search, only a few hyperparameters were tweaked.

The first parameter that was included in the grid search was `max_features` which builds a vocabulary that only takes the top n features into account (Scikit-learn 2020b). This can be useful to include since setting this parameter to ”none” (which is the default choice) will let the classifier take all words into consideration when making a predictions. Even the words that only occur once which are often unnecessary. This parameter was eventually set to 900.

The second parameter included in the grid search was and `max_df` which set a roof of how many times a word can occur in the text before being ignored. For example, the word ”the” is possibly often used in texts that are both negative and positive towards the vaccine. Hence, it is not necessary to take that word in to account when making a classification. (Scikit-learn 2020b). The grid search proposed a value of 0.9.

Thirdly, `ngram_nrange` was included. This parameter is by default set to (1,1). The numbers represents the lower and upper bound of the n-grams that will be used. Default settings imply that only unigrams is used (Scikit-learn 2020b). This parameter was set to (1,1), i.e, only unigrams.

Evaluation of performance

The primary evaluation score in this task is the accuracy score, which is motivated as the two classes in the data set are approximately equally big and no class is more important to classify correctly than another.

A DummyClassifier was introduced for being used as a trivial baseline classifier. Its strategy was chosen as ”most frequent” and the DummyClassifier thus always predicts the most frequent class found in the data set. If the more advanced model is worse than or equally good as the DummyClassifier, it means that the more advanced model is not able to create any more insight from the data than what the most simple rules does. As could be expected from a data set with two approximately equally big classes, the DummyClassifier only reached an accuracy score of 0.530 by always predicting the most frequent class.

As the finally selected model, the Multinomial Naive Bayes classifier, reached an accuracy score of 0.760 with optimized hyperparameters on the small training data set, one can see that this is clearly higher compared to the DummyClassifier and thus it is possible to state with confidence that the model has been able to draw valuable insights from the data.

Training and testing the Multinomial Naive Bayes classifier, with the same hyperparameters, on the large training data set, it was possible to reach an accuracy of 0.794. Testing this already trained model on the final test data set, it was possible to reach an accuracy of 0.878.

That the accuracy is higher for the large training data set compared to the small training data set is not rather surprising considering the big differences in size for the respective data set (975 versus 8788 data points). With a larger amount of training data, it is possible for the model to draw more insights and thus be able to better fit the model to the data.

What is very interesting is that the accuracy is higher on the test data set compared to the training accuracy. Normally the test accuracy is the lower value as it can be expected to contain some data points that do not behave exactly as in the training data set and thus can not be predicted accurately by the model. In this case, it seems like the model knows more about the data set it has not been exposed to before, which is surprising. A possible explanation for this is that most of the comments in the test set have very clear features that define them as either positive or negative comments, and that the training data set contains many ambiguous sentences.

A confusion matrix (Figure 1) was also created on the predictions from the Multinomial Naive Bayes on the large training data set in order to check whether the model is prone to misclassify to a certain class. 83 % of the actual negative comments (0s) were correctly classified, while 75 % of the actual positive comments (1s) were correctly classified. This shows that the model is slightly better at classifying negative comments correctly, but the difference in performance for the two classes is rather minor and should probably not be paid any particular attention.

In some tasks it is more important to classify a certain class correctly. This is for example the case when classifying X-ray pictures as either cancer or not cancer. In this case, the client is probably more likely to want a false alarm of cancer sometimes rather than sometimes missing to detect the cancer. In such cases, metrics as precision and recall could be interesting to calculate from the confusion matrix.

However, in this task it is equally interesting to correctly classify both classes as no more specific objective has been stated. Therefore, accuracy score is still the most suitable metrics for evaluating the performance of the model, as stated in the beginning of this section.

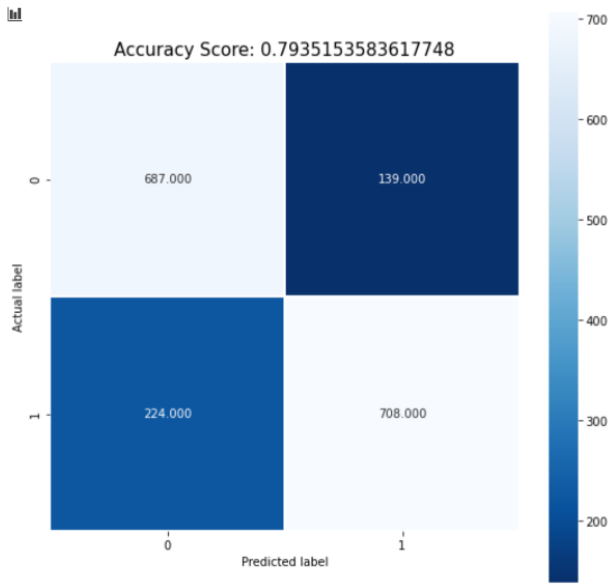

Figure 1: Confusion matrix for Multinomial Naives Bayes model run and tested on the large training data set.

In connection to the confusion matrix, it could be interesting to look at two examples of misclassified comments.

” Imagine spending years and thousands of dollars in education and training to become a doctor only to have somebody say “I don’t trust you I know my child’s health better than you” ” was incorrectly classified as a negative comment, while the correct label was a positive comment.

”One side effect of the vaccine that seems to be cropping up recently is known as “COVID Arm”” was incorrectly classified as a positive comment, but the real label was a negative comment.

What is common for both these sentences is that they only contains very vague nuances of positive respectively negative vaccination stances. Even a human reader must look very carefully to be able to catch the tone in the comments. Thus it is not strange that machine learning-based classifier as Multinomial Naives Bayes, which bases its decision by either linear relationships or rather simple non-linear relationships, is not successful at classifying these two comments. Probably, a neural-network based classifier could be more successful for such ambiguous comments as neural networks are able extract more advanced non-linear relationships among data points.

Feature importance

As we selected a Multinomial Naive Bayes classifier as the final model, it is possible to retrieve the `coef_` attribute from the trained model, which represents the log of the estimated probability of a feature given the positive class. Thus a higher coefficient implies that a feature is more important for the positive class.

A method (`most_informative_feature_for_class` was created for extracting the coefficient together with the feature name and sorting this list with respect to the coefficient. The five features with the highest coefficient can be seen in the table below.

Feature	Importance
the	-3.84
to	-4.12
vaccine	-4.27
and	-4.32
it	-4.46

Table 2: The five most important features together with the feature coefficient for the final model run on the large training data set.

This is a very interesting result as all of these words are equally applicable both to the negative and to the positive class, and, by common sense, should not be able to use for distinguishing between positive and negative comments.

It was attempted to remove so called stop words, common words such as ”in”, ”and” and ”or” that do not bring any value by themselves, in the `TfidfVectorizer`. However, by doing that, the accuracy score decreased slightly and therefore it was decided that the final model would include the stop words. Also, the most important features were still words that are not specific to either category.

In order to understand this strange result for important features, one must probably look deeper in the definition of the feature coefficient and understand exactly why ”the” gets the highest feature coefficient of -3.84. However, we were not successful at that and are therefore not able draw any conclusion about which features that the Multinomial Naive Bayes models considers important. It would be interesting to explore further how feature importance’s are calculated in future assignments.

Conclusion

To conclude, this report has described one possible approach to create a classifier that can determine weather a text expresses a positive or negative opinion towards the COVID-19 vaccination.

The implementation and evaluation of multiple model led to a conclusion that using a Multinomial Naive Bayes Classifier when creating a stance classifier could be beneficial.

The lessons of this project were fourfold. Firstly, the writers got more experience regarding annotating data by themselves. Secondly, approaches to clean data, process and select features and tuning a model were explored. Thirdly, the results of the machine learning test were analyzed which led to increased knowledge about the performance of a model. Finally, the writers learned to describe their project in an academic manner.

References

Scikit-learn. 2020a. Multinomialnb.

Scikit-learn. 2020b. Tfidfvectorizer.

Willsher, K. 2021. Vaccine scepticism in france reflects 'dissatisfaction with political class. *The Guardian* 7.