Technical Report on Text Mining

To classify the emails as spam or not spam, several algorithms are to be used. In this report we will analyze the different algorithms, varying their parameters to improve the model accuracy, and choose the best option for this problem. The main.py file will include the selected algorithm with its selected parameters.

We proceed now to study the accuracy of the different classifiers when we use a 30% of the train dataset to train, when we use a 80% and when we use a 90%. Execution time of the algorithm can also be an important factor to decide what algorithm is the best, but in this case it is not necessary, as all the programs have an execution time lower than five minutes

Naïve Bayes

	Gaussian	Laplace	Multinominal (alpha=0.5)	Multinominal (alpha =0.01)	Multinominal (alpha =0.000001)
30% training	0.92133	0.91409	0.94095	0.95980	0.95904
80% training	0.92133	0.93933	0.95066	0.96066	0.95866
90% training	0.91200	0.93600	0.94400	0.95066	0.95200

We can observe in this case that the multinominal NB is the best algorithm. Alpha values do not affect so much the accuracy, but it seems that the best value would be 0.01. Yet, an accuracy close to 96% can be improved with the use of other algorithms.

KNN

	Uniform, 5 neighbors	Weighted distance, 5 neighbors	Weighted distance, 1000 neighbors	Weighted distance, 2 neighbors
30% training	0.83866	0.85200	0.77219	0.89695
80% training	0.87400	0.88866	0.79133	0.92866
90% training	0.88000	0.89866	0.78000	0.93200

Results in this case clearly indicate that Knn is not the best algorithm for text classifying. The distance of the attributes for text mining are not so important, it is the repetition of key words what matters. That makes this algorithm not as accurate as we want it to.

SVM

	C =1	C = 100	C = 3
30% training	0.96076	0.95923	0.96285
80% training	0.97800	0.97266	0.97866
90% training	0.96933	0.96933	0.97200

SVM is an algorithm that can be very accurate with any kind of data, given that the parameters are correctly adjusted. After the experiments, the conclusion we can extract is that the value of C should be between 3 and 4 to get the maximum accuracy, which is close to 98%.

Neural Networks (MLP Classifier)

	Default parameters	Optimized parameters
30% training	0.97600	0.97866
80% training	0.97600	0.97866
90% training	0.97533	0.97800

Neural Networks also work very good with text mining. This is the algorithm that takes longer to execute but it is also the one with the best results out of the non-ensemble algorithms that we have studied.

Ensemble Methods (Random Forest)

	100 trees	1000 trees
30% training	0.95504	0.95561
80% training	0.97466	0.97666
90% training	0.96266	0.96666

Random Forest is an ensemble method that consists of combining the classification of several decision trees. It works well with high dimensional data, as we can observe in the accuracy results.

Ensemble methods (Experiments)

	KNN-SVC-Naïve Bayes	Random Forest-NN-SVC
30% training	0.96066	0.96933
80% training	0.96066	0.98000
90% training	0.96066	0.97066

We are trying now to improve the results obtained by the standard algorithms. The first combination to be done was to try to combine algorithms that do not have very high accuracy, but there is a possibility that each one works accurately in different portions of the dataset. Results indicate that the combination is very robust and does not depend on the training set length, but it is still not as accurate as it should be.

In the second case, we combine three algorithms that already had high accuracy. The nature of these algorithms is very different, which will make the results even more solid. As we expected, this algorithm has reached 98% accuracy layer, but it has also the highest execution time of all.

Conclusion

Although SVC with C = 3, Neural Networks with optimized parameters and the last ensemble experiment have an accuracy near 98%, the final decision is to use the Random Forest-NN-SVC model, as it is a model that studies from three different approaches what should be the classification of an email. Thus, it is more likely that if two algorithms agree on a decision from two different paths, it is very probable that they are right. Furthermore, the execution time is not exaggeratedly larger than the one of the other algorithms, so this will not be a problem.

Comments on the code

The folder where the code is includes two python files. The preprocessing one basically converts the csv into a data frame and eliminates words that do not give information. In every module that has been uploaded, the data frame must be vectorized to be trained. Then there is a selection of data because the dimensions of the data are too big (there are many words in a set of 200 emails). Finally, the classifier(s) are created and trained. Finally, the predictions are made. This is analogous for all the experiment files and also to the main file.

In the main file, the set does not have to be splitted into train and test. The other difference with the rest of the files is that the predictions are extracted and placed in the predictions.txt file and it does not need to print the accuracy. We suppose that there are tools for calculating the accuracy from the predictions.txt file itself and that is the responsibility of the testers.