# Sentiment and topic classification with Naive-Bayes

**Remy Wang / Richard Scholtens**
s2212781 / s2956586

## Abstract

This research examines if it is possible to create a classifier which can be used for sentiment and topic classifications of reviews. A Naive-Bayes classifier was used with count and tf-idf vectorizers as features. By comparing these results with the results of the Dummy classifier of Scikit Learn as baseline, there could be determined that this was indeed possible. For the sentiment classification the count vectorizer resulted in a F-score of 0,80 and the tf-idf vectorizer resulted in a F-score of 0,79. The baseline resulted in a F-score of 0,51. For the topic classification the count vectorizer resulted in a F-score of 0,89 and the tf-idf vectorizer resulted in a F-score of 0,91. The baseline resulted in a F-score of 0,17.

## 1 Introduction

In the computer era many new technologies have been developed. Some of these technologies are based on comprehensive programming algorithms which can be used as tools for prediction. One of the techniques used is the machine learning technique. Machine learning can best be described as set of statistical computations where input is given, and output is predicted. There are different sorts of machine learning techniques but before one can understand the newest machine learning techniques one must understand the basic principles of machine learning. Therefore, in this report one of the first machine learning classification techniques will be explained with the emphasize on Naive Bayes. This technique will be examined by training a statistical model which can predict the sentiment of a sentence. The same technique will also be used to train a statistical model which can predict the class of a sentence. Specifically, the following research question will be addressed:

- Can a good performing classifier be built for sentiment classification and topic classification for reviews?

## 2 Data

Machine learning models require input data which is used for training. Therefore, it is necessary to obtain a data set. In this inquiry a data set with 6000 reviews is used which is provided by the Rijksuniversiteit Groningen. The data set is distributed as a text file. Each line in the text file represents an instance and is structured in the following order: class, sentiment, document, and sentence. The sentences are classified by a total of six classes which include books, camera, dvd, health, music, and software. An instance can also be classified according to a positive or negative sentiment. The given document of an instance refers to the specific text file to which the sentence belongs. Pre-processing has been applied by tokenizing and lowercasing the data. Finally, the data has been split into a training set(75%) and test set(25%).

| Class | Sentiment | Document | Sentence |
|---|---|---|---|
| music | neg | 575.txt | the cd came ... |
| dvd | neg | 391.txt | this was a very ... |
| health | neg | 848.txt | the braun ... |
| camera | pos | 577.txt | when it comes to ... |

Table 1: Sample data

## 3 Method/Approach

For this experiment the Multinomial Naive-Bayes classifier will be used. This classifier will be used in two tasks. The first task is classifying the sentiment of reviews being either positive or negative.

| Class | Support |
|---|---|
| negative | 3032 |
| positive | 2968 |
| books | 993 |
| camera | 988 |
| dvd | 1012 |
| health | 986 |
| music | 1027 |
| software | 993 |

Table 2: Distribution of labels

The second task is a multi-label classification task where we must classify each instance into one of the following six: books, camera, dvd, health, music, software.

A baseline is required to measure the performance of our classifier. For the baseline we use the dummy classifier provided by Scikit Learn (Pedregosa et al., 2011), where we use the stratified strategy which generates predictions by respecting the training sets class distribution.

Two vectorizers will be used for our classifier on both tasks which are included in Scikit Learn pedregosa2011scikit. The first vectorizer that is used is the term frequency inverted document frequency (tfidf) vectorizer. This vectorizer gives a weight to each word using the equation shown in figure:

$$tfidf_{i,d} = tf_{i,d} \cdot idf_i$$

The second vectorizer that is used is the count vectorizer. This vectorizer simply counts the occurrence of each word.

For the evaluation of the performance of our tasks, precision recall and F-1 score will be used and compared with the baseline. The accuracy will also be calculated in both tasks. A confusion matrix will also be created for both tasks as it visualizes well how your system does.

## 4 Results

The results from classifiers with features for both tasks will be presented in separate tables. In table 4 the results for the sentiment classification task are presented. In table 4 the results for the topic classification task are presented. Also, the results of individual labels will be presented. In table 4 the results per label for the sentiment classification are presented and in table 4 the results per label for the topic classification task are presented.

| Classifier | Precision | Recall | F1 | Acc |
|---|---|---|---|---|
| Baseline | 0.51 | 0.51 | 0.51 | 0.51 |
| NB(tfidf) | 0.81 | 0.79 | 0.79 | 0.79 |
| NB(Count) | 0.80 | 0.80 | 0.80 | 0.80 |

Table 3: Results of the sentiment classifier

| Classifier | Precision | Recall | F1 | Acc |
|---|---|---|---|---|
| Baseline | 0.17 | 0.17 | 0.17 | 0.17 |
| NB(tfidf) | 0.91 | 0.91 | 0.91 | 0.91 |
| NB(Count) | 0.90 | 0.89 | 0.89 | 0.89 |

Table 4: Results of the topic classifier

Shown in table 3 and 4 the features we used for our model does not influence the performance too much. Both classifiers perform about equal on both tasks, whereas the count vectorizer feature performs best in the sentiment classifier with a F1-score of 0.80 and the tfidf feature performs best in the topic classifier with a F1-score of 0.91.

| Label+features | Precision | Recall | F1 |
|---|---|---|---|
| neg(tfidf) | 0.73 | 0.92 | 0.81 |
| pos(tfidf) | 0.90 | 0.67 | 0.77 |
| neg(count) | 0.76 | 0.85 | 0.80 |
| pos(count) | 0.84 | 0.75 | 0.79 |

Table 5: Results of individual labels of the sentiment classifier

| Label+features | Precision | Recall | F1 |
|---|---|---|---|
| books(tfidf) | 0.94 | 0.91 | 0.93 |
| books(count) | 0.91 | 0.91 | 0.91 |
| camera(tfidf) | 0.83 | 0.94 | 0.88 |
| camera(count) | 0.82 | 0.94 | 0.87 |
| dvd(tfidf) | 0.88 | 0.91 | 0.89 |
| dvd(count) | 0.88 | 0.87 | 0.87 |
| health(tfidf) | 0.97 | 0.79 | 0.87 |
| health(count) | 0.98 | 0.79 | 0.88 |
| music(tfidf) | 0.96 | 0.95 | 0.95 |
| music(count) | 0.95 | 0.92 | 0.93 |
| software(tfidf) | 0.89 | 0.93 | 0.91 |
| software(count) | 0.86 | 0.93 | 0.89 |

Table 6: Results of individual labels of the topic classifier

Shown in table 5 and 6 the performance on

one class is not significantly better than the other classes.

## 5 Discussion/Conclusion

This research set out to find an answer whether it was possible to build a good performing classifier for sentiment classification and topic classifications for reviews. By using a Multinominial Naive-Bayes classifier a model was created which used review data as input. The previous section presents the results of the sentiment and topic classifiers in table 2 and 3. For the sentiment classification the count vectorizer resulted in a F-score of 0,80 and the tf-idf vectorizer resulted in a F-score of 0,79. The baseline resulted in a F-score of 0,51. For the topic classification the count vectorizer resulted in a F-score of 0,89 and the tf-idf vectorizer resulted in a F-score of 0,91. The baseline resulted in a F-score of 0,17.

This means it is possible to create a classifier which can perform on both tasks well enough. However, it is still possible to improve the classifier by adding more features or using more data. It is also possible to use high information words as a feature so it can tell which words are most informative for which sentiment or class. In this case the topic classifier outperforms the sentiment classifier. This could be due to the fact this classifier holds more important features on which to base its decisions. A feature can be highly responsible for assigning a label to an instance. In the topic classifier there are more classes then in the sentiment classifier and therefor the chances are higher that the topic classifier possesses such a feature. By identifying this feature, it is possible to improve the classifiers as well.

## 6 Answers to additional questions

**(a) Why should one not look at the test set?**
If one knows what is in the test set the model can be easily programmed to fit with the test set. Because of this the system will most likely perform well on the test set but most likely not on other data sets due to overfitting.

**(b) What happens with cross-validation?**
Cross-validation is a technique where a data set is split up in n-parts. Most commonly the set is split up in ten separate parts. One part will be used as a test set and the rest as training set which will give a result. Every part will be used as a test set once which will result in ten separate results. The average of these results will become the outcome of the model. By doing so all parts of the data set will be tested and therefor give a more all-round result.

**(c) What baselines could you use for the binary sentiment classification and for the six-class topic classification? What would their performance be?**
To create a baseline for the binary sentiment classification one could always guess one label. The labels are evenly distributed and therefore their performance would be around 0.17. This can be achieved by using the dummy classifier with the most frequent strategy. For the six-class topic classification it is possible to use the class with the most observations as baseline for topic classification. It might happen that the number observations are equal for all classes. In this case it is possible to select a specific class and enumerate each class in order to see which gives the better results. In our case the performance will be around 0.17. This can be achieved by using the dummy classifier with the stratified strategy.

**(d) Why is it useful to look at the confusion matrix?**
A confusion matrix allows you to see inside your system more accurate. It visualizes how well the algorithm performs with the help of precision, recall and F-score. By doing so it is easier to determine which classes get confused and therefore can identify more easily how to improve the system. One way doing this is by using more data or different features.

**(e) What features are used in the classification task you ran?**
A count vectorizer is used in order to determine the frequencies of the words in the sentences which is used as feature. Another feature used is the tfidf-vectorizer which creates a score scheme based upon the frequencies of words within specific documents.

**What is the difference between using accuracy and using F-score?**
Accuracy only returns the total number of correctly predicted identified cases. However, the F-score uses precision and recall in order to return

a harmonic average. By doing so it corrects potential outliers.

**(f) What is the difference between macro F-score and micro F-score and what different functions and implications do they have?**
Micro F-score considers the total true positives, false negatives and false positives (no matter of the prediction for each label in the data set). Macro F-score considers each label and returns the average without considering the proportion for each label in the dataset. Macro-averaged metrics are used when we want to evaluate systems performance across on different datasets. Micro-averaged metrics should be used when the size of datasets is variable.

## References

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830.