TP1

Natural Language Processing

Experiment 1

In this experiment we implement a naive Bayes classifier that assigns a movie review to a class (here positive or negative) depending on the review's total probabilities of belonging to each of the classes. The review's probability of belonging to a specific class is based on the general prior of each class and its words' probabilities of appearing in a class or another.

In order to run the experiment, we start with the training procedure that computes for each word in the train set conditional probabilities of appearing in each of the classes. All conditional probabilities are based on the relative frequency of words in the respective class.

In the first experiment we do not do any preprocessing of the data except for splitting it at spaces and line breaks. We start by going through the training reviews one by one, extracting the words that appear there and noting them in the training vocabulary. For each word in the training vocabulary, we count how many times it appeared in each of the classes (in positive reviews and negative ones) and then compute its conditional probability of appearing in one class by dividing its respective count with the total number of counts for the words in that class.

$$P(w_i|class = c) = \frac{count(w_i\ in\ c)}{\sum_{j=1}^{V} count(w_j\ in\ c)} \tag{1}$$

where V is the size of the training vocabulary for both classes.

When computing the conditional probability of a word of appearing in a specific class, there are many situations when a word appears in only one class, meaning that its conditional probability for the other class is zero. This is problematic when computing the total conditional probability of a test review that contains that word, as it is based on the product of all conditional probabilities of the words it contains and it would give an overall zero.

$$P(R|class = c) = \prod_{i \in positions} P(w_i|c) \tag{2}$$

In order to avoid such situations, we remove from the training vocabulary any words that do not appear in both classes. This implies a reduction in the size of the vocabulary.[1] Finally, we classify a review as *positive* or *negative* depending on which conditional probability is higher.

The results from experiment 1 are presented in tables 1-2, in which we use words' conditional probabilities and class priors[2] from the train set in order to classify reviews in the test set. We

1. This is ambiguously defined in the assignment but I am guessing that this is expected from us, given that in the second experiment we do a *plus one smoothing* for the words in the training vocabulary, in order to avoid eliminating words that do not appear in both classes. However, I have to admit that dropping the words that do not appear in both classes does not make sense for me, as we impose to have the same vocabulary in both classes whereas in Ch. 4 of the *NAIVE BAYES AND SENTIMENT CLASSIFICATION* book the authors insist with *Note once again that it is crucial that the vocabulary V consists of the union of all the word types in all classes, not just the words in one class c.*

2. The train set consists of 900 positive and negative reviews each. This means that the class priors are equal and uninformative.

can see that the naive Bayes classifier performed quite well in the case of English reviews, with an overall accuracy of 83%, and with both the true positive and true negative rates of 83% (number of test reviews correctly classified). The precision [3] for the English classification is also 83%.

In the case of French reviews, the naive Bayes classifier does not work as well : the true positive rate (also called recall) is only 43% whereas the true negative rate is 98%. This gives an overall accuracy of 71%. The precision is 96%.

TABLE 1 – Confusion matrix for review classification ; Experiment 1

English

		True class	
		Positive	Negative
Predicted	Positive	83	17
Class	Negative	17	83

French

		True class	
		Positive	Negative
Predicted	Positive	43	2
Class	Negative	57	98

TABLE 2 – Statistics for review classification ; Experiment 1

	English	French
True positive rate	0.83	0.43
True negative rate	0.83	0.98
Precision	0.83	0.96
Accuracy	0.83	0.71

Experiment 2

For the second experiment we focus on the English reviews and adapt our train code in such a way that we improve the preprocessing by tokenizing, lowercasing the data and dropping stopwords. For tokenizing we use a list of punctuation symbols that we want to be split from the words. In the list we include the typical punctuation marks {. , ; : ? ! ... } as well as other symbols that are not usually tied to a word { ( ) [ ] { } # _ * }.

The stopwords are words that are not necessarily related to the content and that should be ignored. For example, Google has a list of words that it removes from search queries because they do not improve the search results and create lags or could even add a level of ambiguity in the results. This words are mostly neutral words, such as a, an, the, then, and, because, by, do, for, from..... For this experiment, I use Google's list of stopwords and add to it some tokenized symbols from the previous step. These symbols are the ones that are *not* related to punctuation

3. Precision is the ratio of the true positive value to the total number of reviews predicted positive.

(mainly the second set of symbols given above), because the punctuation ones are important for the meaning of a word set (for example enthusiastic reviews might use many exclamation marks).

Moreover, instead of ignoring the words that do not appear in both classes in the trainset, I apply the *plus one* smoothing, that adds by default one to all word counts in both classes. This means that, for each class, the total counts of words in the vocabulary will increase with the vocabulary size. Otherwise, the conditional probabilities of words to appear in a certain class are computed in the same way, but using the new smoothed counts.

Table 3 shows the confusion matrix and other measures of classification performance, based on the task of classifying the reviews in the test set.[4] We can see that compared to experiment 1, the true positive rate goes slightly down to 81% whereas the true negative rate increases to 89%. Overall, the accuracy is 85% whereas the precision is 88%. The results indicate that tokenizing the data helps improve the classification performance.[5]

TABLE 3 – Statistics for English review classification ; Experiment 2

		True class	
		Positive	Negative
Predicted	Positive	81	11
Class	Negative	19	89

True positive rate	0.81
True negative rate	0.89
Precision	0.88
Accuracy	0.85

4. We also apply tokenizing and stopwords dropping on the test data.
5. Results from applying the smoothing procedure but no tokenizing do not show much difference compared to experiment 1.