Traitement automatique du langage
TP 2 — Classification of Movie Reviews with Naive Bayes

Asheesh Gulati
Exercises prepared by Yves Scherrer
4.10.2018

Submit by 17.10.2018 midnight.
Submit a separate script for each experiment, and name your archive `lastName_firstName_tp2.zip`.

Create a Naive Bayes classifier that learns to classify movie reviews into positive and negative ones, using a bag-of-words model.

Data: We provide two datasets on Chamilo: *movie-reviews-en.zip* contains English movie reviews from the IMDb archive[1], and *movie-reviews-fr.zip* contains French movie reviews from the allocine.com website.[2]

These reviews have been classified as positive or negative reviews based on numerical and star ratings associated with them. Both archives contain a *train* folder with the positive and negative reviews to be used for training your classifier, and a *test* folder with the reviews to be used for evaluating it.

The Naive Bayes algorithm: The basic formula of Naive Bayes is the following, where C represents the class of the document (here: positive or negative) and $f_1 \dots f_n$ represent the features (here: the words in the documents):

$$p(C \mid f_1 \dots f_n) = p(C) \cdot p(f_1 \dots f_n \mid C)$$

[1]The data are extracted from the *polarity dataset v2.0*, available at `http://www.cs.cornell.edu/People/pabo/movie-review-data/`.
[2]This dataset was made available by Hatem Ghorbel, HES-SO St-Imier. Please do not redistribute it without the author's consent.

If we assume that the features are independent, we can rewrite the formula as follows:

$$p(C \mid f_1 \dots f_n) = p(C) \cdot \prod_n p(f_n \mid C)$$

This formula allows us to compute the probability of a class given the words of the document. When testing your algorithm, you'll want to obtain a class label as an answer. This means that you'll have to compute the probability of both classes (positive and negative) of the document, and choose the label of the one whose probability is higher:

$$\hat{C} = \arg\max_C p(C \mid f_1 \dots f_n) = \arg\max_C p(C) \cdot \prod_n p(f_n \mid C)$$

This is the formula that you are going to use for **testing**.

The **training** part will consist of computing the priors $p(C)$ and the likelihoods of the features $p(f \mid C)$. They can be computed as follows using maximum likelihood estimation:

$$p(C) = \frac{\textit{number of training documents in } C}{\sum_{C'} \textit{number of training documents in } C'}$$

$$p(f \mid C) = \frac{\textit{Count}(f, C)}{\sum_{f'} \textit{Count}(f', C)}$$

where $\textit{Count}(f, C)$ refers to the number of occurrences of word f in all training documents of class C.

Dealing with underflow: The Naive Bayes algorithm computes a product whose factors are probabilities between 0 and 1. However, the product of numbers between 0 and 1 may well be smaller than the smallest floating point number that a computer may represent. To resolve this problem of *underflow*, you can simply substitute a probability by its logarithm. Likewise, the product of probabilities is replaced by the sum of their logarithms. So, instead of computing

$$p(C \mid f_1 \dots f_n) = p(C) \cdot \prod_n p(f_n \mid C)$$

you should compute

$$\log p(C \mid f_1 \dots f_n) = \log p(C) + \sum_n \log p(f_n \mid C)$$

Maximizing the product of probabilities is equivalent to maximizing its logarithm.

Experiment 1

We ask you to create a simple model with the following parameters:

- Do not preprocess the data except splitting it at spaces and line breaks.
- If a word has not been seen during training, disregard it.
- Use a uniform prior.

Your code must contain two clearly distinguishable parts (e.g. two procedures): a training part and a testing part. Use this model to create and test a classifier for English reviews, and to create and test a classifier for French reviews.

Report the results of this experiment in the form of a confusion matrix. Also report the overall accuracy of your classifier, as well as precision and recall values for both classes.

Experiment 2

Choose one dataset (English or French) and improve the simple model in the following ways:

- Improve the preprocessing by **tokenizing** (i.e. splitting punctuation marks off from words) and/or lowercasing the data.
- The files have not been cleaned and contain words and other signs that are not useful for the classification (**stopwords**). Try to eliminate these words and signs before using the texts for training or testing. Please describe clearly which tokens you consider irrelevant and why.
- Instead of throwing away unseen words, use **smoothing**:

  It is likely that the test data contains a feature f (here: word) that has not been seen in the training data. Using simple maximum likelihood estimation as above, $P(f \mid C)$ will be 0, and by multiplication, the probability of the whole document as well. To avoid this problem, we will use *add-one smoothing*. We will add 1 to each observation, whether it appears in the training corpus or not. Consequently, the number of events is increased by the total number of observations encountered in the training corpus. So, instead of computing

  $$p(f \mid C) = \frac{Count(f, C)}{\sum_{f'} Count(f', C)}$$

  you should compute

  $$p(f \mid C) = \frac{Count(f, C) + 1}{\sum_{f'} (Count(f', C) + 1)}$$

Report the results as above.

The relevant research paper on English reviews (Pang & Lee, ACL 2004) reports an overall accuracy of 82.8% using a Naive Bayes classifier on the full reviews. No Naive Bayes results are reported on the French reviews, but a comparable strategy reports an overall accuracy of 91.75% (Ghorbel & Jacot, DART 2010). You might get different results due to our data selection and depending on your data cleaning and smoothing strategies.