Exercise Sheet 6

Due: 09.01.2019, 09:00

– This exercise sheet has twice the workload of the other sheets. Start early! –

In this exercise, we implement two Naïve Bayes classifiers for text classification and conduct a comparative scientific experiment on a spam dataset.

Download the following files

- `NaiveBayesTextClassification.ipynb:` ISIS
  - This notebook demonstrates how to implement classes in Python and how to use the CountVectorizer from sklearn for feature extraction from text.
- SMS Spam dataset: https://archive.ics.uci.edu/ml/datasets/sms+spam+collection

Exercise 6.1 (one point for each part, a and b)

Implement the following Naïve Bayes classifiers (for general numerical features, not for text)

a) Bernoulli Naïve Bayes for binary feature vectors $x = (x_1 ..., x_n), \ x_i \in \{0,1\}$

b) Multinomial Naïve Bayes for feature vectors of the form $x = (x_1 ..., x_n), \ x_i \in \mathbb{N}$

For this, complete the Python classes `BernoulliNaiveBayes` and `MultinomialNaiveBayes` in the Jupyter notebook `NaiveBayesTextClassification.ipynb`. The classes and methods can be extended if necessary (for instance, you may want to add class variables or input and output parameters of methods). Part of this task are the `fit()` and `predict()` methods. Make sure to avoid arithmetical underflow.

Exercise 6.2

Extend both implementations by Laplace Smoothing. Why is this important?

Exercise 6.3

The `evaluate()`-methods should return the following evaluation metrics: Accuracy, Precision, Recall.

Exercise 6.4

Implement the following routine.

- take raw train and test data (i.e. unprocessed texts with labels) as input
- extract features from the data (according to Bernoulli or Multinomial Naïve Bayes). For this, you can use the CountVectorizer from Scikit-Learn.
- train the model
- evaluate the model

Exercise 6.5

Estimate generalization performances for accuracy, precision and recall with train-test split or cross validation (what do you think is more appropriate here?) on the dataset for the following models:

a) Bernoulli Naive Bayes with 1-grams.
b) Bernoulli Naive Bayes with 1-grams und 2-grams.
c) Multinomial Naive Bayes with 1-grams.
d) Multinomial Naive Bayes with 1-grams und 2-grams.

An n-gram is a sequence of n consecutive words. Note that in this case, the vocabulary consists of n-grams rather than single words. It is allowed to use Scikit-Learn to generate cross validation splits of the data (consider for example sklearn.cross_validation.KFold).

Exercise Sheet 6

Due: 09.01.2019, 09:00

Exercise 6.6

Estimate accuracy, precision and recall for a classifier that always returns ham as answer and for a classifier that returns random answers (uniformly from {ham,spam}). These classifiers can be used as a baselines.

Exercise 6.7 (Optional / 1 bonus point)

Evaluate further model configurations, for example with removing stopwords, applying stemming, lemmatization or spelling correction, using TFIDF features, higher n-grams, or other natural language processing (NLP) techniques.

Exercise 6.8

Write an experimental section reporting on your experiments. Define your own goals of this study (what could be interesting to investigate?). Make sure that the experimental setup is so precisely described that it is repeatable and that the results are reproducible. Present and discuss your results in an appropriate way.

Using Scikit-Learn

Scikit-Learn should only be used if indicated by the task. You can use Scikit-Learn in another task, A say, (e.g. Naive Bayes implementation) in order to be able to solve further tasks building upon task A. But then you'll not receive a point in task A.

Submission modalities

Submit a single zip file per group. This mini project should be solved in team work. The zip should contain a single python notebook and a single report as pdf, preferably written in Latex. Please name the zip as follows: G<group_number>_E06.zip. Example: G05_E06.zip