Johannes Gutenberg-Universität Mainz
Institut für Informatik
Data Mining
Prof. Dr. Stefan Kramer

Machine Learning - Sheet 5

24.05.2018
Deadline: 07.06.2018 - 16:00

Task 1: Naive Bayes algorithm *(20 Points)*

The overall goal of this exercise is to get hands-on experience with the implementation of a popular machine learning scheme and to work on a real-world problem. The task is to implement an improved version of the Naive Bayes algorithm that is able to predict the domain - one of Archaea, Bacteria, Eukaryota or Virus - from the abstract of research papers about proteins taken from the MEDLINE database. The team, which manages to predict the largest fraction of test examples correctly, wins the "Machine Learning JGU Cup 2018" for fame, fortune and glory.

The Data Set

The data set contains the abstracts of research papers that deal with proteins. Each protein is found either in Archaea, Bacteria, Eukaryota, or Virus. The first character in each record specifies the domain of the protein, as given in the following table.

Code	Domain
A	Archaea
B	Bacteria
E	Eucaryota
V	Virus

The second attribute in the record is a character string containing the abstract to be classified. The string is preprocessed: it contains only whitespace, alphanumerical lowercase characters, digits, the dash (–) and the prime ('). Each contiguous sequence of non-whitespace characters framed by whitespace is considered a word. You can download the data set from Moodle: `train3500.txt`. The file contains 3500 training examples with class label. You will then hand in your runnable code that takes two arguments: the path to a test file without class labels and the path to the output file which contains the predictions of your implementation on the test data.

The Challenge

The ultimate goal of this programming project is to come up with an implementation of a (possibly extended or modified) Naive Bayes algorithm, that achieves a high predictive accuracy on the test data. As a minimum requirement your implementation should be at least the same as a vanilla-plain version of Naive Bayes as explained in a standard textbook. You might, however, change some assumptions, representations or models in your implementation. Basically, there are two parts to be solved:

(a) First of all, you need to decide about a suitable representation for the text in the abstract. An easy way to obtain an attribute-value representation is to identify the 1000 most frequently occurring words and generate 0-1 attributes stating whether or not the word occurs in the corresponding example. There are other possible representations, e.g. one could take the occurrence frequency of a word in the abstract into account.

(b) The standard Naive Bayes algorithm as outlined in Mitchell's "Machine Learning" [1] will probably yield comparably poor predictive accuracy, so you need to improve it in order to obtain good predictive accuracy. There has been some research on improving Naive Bayes in general and especifically for text classification (you might want to read the papers provided in Moodle). Of course, you are also welcome to come up with your own improvements, so feel free to be creative.

Your results

You can submit your preliminary model once until 31.5.2018 to get feedback on how well your model is working on test data (the results will be announced on 1.6.2018). For that, you should upload your trained model and its loader that gets two arguments: the path to a test file without class labels and the path to the output file which contains the predictions of your model on the test data. This phase is not mandatory, though. For your final delivery, please hand in:

- all your Java or Python code,
- your predictions for the test set (Your prediction should be stored in a plain ASCII text file containing one line per test example. Each line should contain the character A, B, E, or V, depending on your prediction for the corresponding test example.), and
- a report explaining the design details of your implementation and your rationale for doing so.

References

[1] Tom M. Mitchell. *Machine learning.* McGraw Hill series in computer science. McGraw-Hill, 1997.