A(E)3M33UI — Semestral project 1: Spam filter

Petr Pošík

April 1, 2014

1 Introduction

The goal of this semestral task is to employ the knowledge of the lectures and previous exercises, and apply it to a text classification task, namely spam filtering. Your task is to create as good spam filter as possible, and describe its principle in a report.

You shall submit the Python scripts which demonstrate what you have achieved, and a report describing the methods and results. *Remember to submit your own work! Do not commit plagiarism! The plagiarism detection feature will be on for this task, and any confirmed plagiat may be a reason to assign 0 points from this task, i.e. you will not receive the assessment!* This warning applies to both the PDF reports and Python scripts.

You can look at an example of text processing using scikit-learn.

1.1 Code

Your code shall be clean, readable, and understandable. This will form one criterion of the evaluation. Please, take the time to come up with meaningful names of variables, functions, constants, etc. Keep the functions short, spanning several lines only.

1.2 Report

The report can be elaborated in Czech or in English. It shall have a form of a scientific article. It shall contain an adequate description of the data, principles, methods and results used and achieved by your work. By adequate description we mean a **concise description** sufficient to **understand and replicate** the work done by you. You should not only present the results, but also try to **interpret** them.

2 Spam filter: mandatory part

The only mandatory part of this task is to create a spam filtering algorithm, describe it in the report, and submit it for the contests.

2.1 Specifications

In module **`filter`**`.py`, implement functions `train_filter()` and `predict()`.

1. `train_filter(X,y)` shall take the training data (email texts X, and email class y), shall train the classifier (or a pipeline with preprocessors and a classifier), and shall return the trained classifier (or the pipeline).

2. `predict(`**`filter`**`,X)` shall take the filter (classifier, pipeline) and the email texts X, and shall return the predictions of the filter for this data.

During contests, these two function will then be imported from your **`filter`**`.py` module and used for the comparison.

2.2 Working example

In the supplement module **`filter`**`.py` you obtain with this assignment, you can see an example implementantion of a dummy spam filter:

1. The filter first transforms the training emails into the *bag of words* representation, using the `CountVectorizer` class.

2. Then it uses the `DummyClassifier` to decide if an email as ham or spam.

3 Filter evaluation

The filters shall be evaluated using a modified accuracy score:

$$macc = \frac{n_{TP} + n_{TN}}{n_{TP} + n_{TN} + 10n_{FP} + n_{FN}} \tag{1}$$

It is already implemented in the supplied **`filter`**`.py`. It uses the confusion matrix feature to compute TP, FP, TN, FN.

You may also need other information how to construct own scoring function and use it e.g. in grid search.

4 Model tuning

Use grid search feature to search for optimal settings of each filter.

5 Comparison of the filters

Compare several types of (preferably tuned) filters using

- ROC curves, and/or
- learning curves.

You can then choose the best one as the final filter to submit.

6 Additional ideas

- Try to modify the tokenization process of the `CountVectorizer`.

```
def my_analyzer(s):
    return s.split()

vec = CountVectorizer(analyzer=my_analyzer)
```

- Try to choose only a subset of words as features (feature selection).

7 Scoring

The final score for the task will be composed of the following components:

Component	Points
Report in LaTeX	0-1
Adequate description of the final chosen filter in the report	2
Contest on dataset A	0-4
Contest on dataset B	0-4
Filter tuning via grid search + adequate description in the report	0-4
Filter comparison + adequate description in the report	0-4
Additional ideas tried + adequate description in the report	0-2+
Total	0-20+

It is theoretically possible to get the required 10 points by just sumbitting the filter and its short (but adequate description, see sec. 1.2) given that the filter will rank in the top 20 % of filters in both contests (see sec. 7.1). Using this strategy, however, you may also end up with less than 2 points.

It is thus strongly recommended to try also the other suggested ways (filter tuning, filter comparison, additional ideas) how to improve your filter score, and describe them adequately in the report. This way, you will

1. build a better filter which will score higher in the contests (thus bringing you more points from the contests), and
2. have the chance to get additional points for this additional effort.

7.1 Contests scoring

In each contest, the filters will be sorted according the modified accuracy (Eq. 1). The first 20 % of filters will get 4 points, the second 20 % of filters will get 3 points, . . . , and the last 20 % of filters will get 0 points.

8 Have fun!