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

Petr Pošík

April 19, 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.

In case you struggle with creating your own working and successful spam filter, you are allowed to pick a filter from the internet, and describe its principles in the report. In that case, however,

- you have to indicate the filter is not your work (and provide reference to the original source), and
- the filter will not be allowed to take part in the contests.

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, if possible.

1.2 Report

The report can be elaborated in Czech or in English. It shall have a form of a scientific article. You can assume the reader has a general background in machine learning, but she may not know the individual methods. The report shall contain an adequate description of the data, principles, methods used and results achieved in 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 and discuss** 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, i.e. email texts X, and email labels (classes "ham" and "spam") y, shall train the classifier (or a pipeline with preprocessors and a classifier), and shall return the trained classifier (or rather the whole 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 functions will be imported from your **`filter`**`.py` module and used for the comparison.

2.2 Working example

In the supplement module **`filter`**`.py` you obtained 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 is ham or spam.

3 Filter evaluation

The filters shall be evaluated using a modified accuracy score

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

i.e. by an accuracy measure which uses 10 times larger penalty for FP than for FN.

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

4 Model tuning

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

TODO

When using the grid search facility, you may also need some other information about how to construct your own scoring function and use it e.g. in grid search.

5 Comparison of several filters

Compare several types of (preferably tuned) filters using

- ROC curves, and/or
- learning curves, and/or
- area under the curve (AUC score), ...

You can then choose the best one as the final filter to submit.
You can choose several types of classifiers from

- k nearest neighbors,
- logistic regression,
- SVM with linear kernel,
- SVM with polynomial kernel,
- SVM with RBF kernel,
- naive Bayes with multinomial distribution,
- decision trees,
- AdaBoost,
- random forrests,
- ...
- TODO add links

6 Additional ideas

To gain additional points, you can try to elaborate the following ideas:

- Try to modify the tokenization process of the `CountVectorizer`. You can set e.g. a custum analyzer like this:

  ```
  def my_analyzer(s):
      return s.split()
  vec = CountVectorizer(analyzer=my_analyzer)
  ```

  Similarly, you cen set only the preprocessor, or tokenizer.

  You can try e.g.

  - to prevent the vectorizer to make the tokens lowercase,
  - to extract other types of tokens than character sequences,
  - ...

- Try to evaluate the importance of individual tokens for the discrimination of spam.
- Try to choose only a subset of words as features (feature selection).

Component	Points
Code quality (readability, understandability)	0-2
Report in LaTeX	0-1
Adequate description of the final filter chosen for competition	0-2
Filter tuning via grid search + adequate description in the report	0-2 per filter type
Filter comparison + adequate description in the report	0-4
Additional ideas tried + adequate description in the report	0-2 per idea
Subtotal	Max. 16 regular points
Contest on dataset A	0-4
Contest on dataset B	0-4
Subtotal	Max. 4 regular points + max. 4 bonus points
Total	Max. 20 regular points + max. 4 bonus points

7 Scoring

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

To get the assessment, you need to get at least 10 regular points from this project. Since you can get at most 4 regular points from the contests, you need at least 6 points for report and code; but at least 10 points for report and code will bring you the assessment independently of the contests results. If you earn more than 16 points for the code and report, only 16 will be counted.

It is recommended to try to improve your filter score in several ways, 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!