# Aritificial Intelligence

## Spam Filter

Eslam Elgourany
Department of cybernetics and robotics.
Prague , Czech Republic
eslamelgourany@hotmail.com

***Abstract*— Text mining is a wide field which has gained popularity with the huge text data being generated. Automation of a number of applications like sentiment analysis, document classification, topic classification, text summarization, machine translation, .. etc . All of these have been done using machine learning models. It is the process of deriving high quality information from the text. High-quality information is typically derived through the devising of patterns and trends through means such as statistical pattern learning. Text mining usually involves the process of structuring the input text (usually parsing, along with the addition of some derived linguistic features and the removal of others, and subsequent insertion into a database), deriving patterns within the structured data, and finally evaluation and interpretation of the output. 'High quality' in text mining usually refers to some combination of relevance, novelty, and interestingness. In this paper we will design a spam filter based on learning a certain data set using modified machine learning method.**

### I. INTRODUCTION

A spam filter is a quit good application for the text mining or text classification. It is a program that is used to detect unsolicited and unwanted email and prevent those messages from getting to a user's inbox. Like other types of filtering programs, a spam filter looks for certain criteria on which it bases judgments. For example, the simplest and earliest versions (such as the one available with Microsoft's Hotmail) can be set to watch for particular words in the subject line of messages and to exclude these from the user's inbox. This method is not especially effective, too often omitting perfectly legitimate messages (these are called false positives) and letting actual spam through. More sophisticated programs, such as Bayesian filters or Adaboost other heuristic filters, attempt to identify spam through suspicious word patterns or word frequency.

For instance, some spam-filtering methods run a series of checks on each message to determine the likelihood that it is spam. Other spam-filtering techniques simply block all email transmissions from known spammers or only allow email from certain senders. And while some spam-filtering methods are completely transparent to both the sender and recipient, others require some degree of user interaction.

### II. PROJECT APPROACH

We will create a filtering algorithm and make some testing on some data set. We will implement some features in the (filter.py) module. We will implement train_filter() and predict() commands as well. So that by running the train_filter it will run a pipe line which includes the classifier. While in predict () the command will run in which it classifies the test data and it will return the value of the evaluated predication.

### III. SPAM FILTER PREPARATION

#### *A. Adaboost*

Adaboost classifier is a meta-estimator that begins by fitting a classifier on the original dataset and then fits additional copies of the classifier on the same dataset but where the weights of incorrectly classified instances are adjusted such that subsequent classifiers focus more on difficult cases. This model is characterized that it has n_estimators and learning rate as parameters.

N_estimators is the maximum number of estimators at which boosting is terminated. In case of perfect fit, the learning procedure is stopped early. While learning rate shrinks the contribution of each classifier by learning_rate and n_estimators.

#### *B. Test and training set*

To let the classifier be more effective and working with higher efficiency we use this function train_test_split in which it will make some random mixing into a given ratio on a training and test set. That will lead that every time we will test the classifier the data will be randomly changed. We split the data in which 70% will be for the training set and 30% will be for the testing set.

#### *C. Vectorizer*

In order to recognize and to be able to identify the text classification in a good way for machine learning; total vectorization using the Tfidf Vectorizer () function is used. Based on the counting of the occurrence of the individual words in the text, comparison is made This feature makes

tuning for the text document into numerical feature vectors, This strategy helps in describing the word occurrences while completely ignoring the relative position information of the words in the text.

### D. *Cross validation*

Learning the parameters of a prediction function and testing it on the same data is a methodological mistake: a model that would just repeat the labels of the samples that it has just seen would have a perfect score but would fail to predict anything useful on yet-unseen data. When evaluating different settings hyper parameters for estimators, such as the n_estimators, there is still a risk of over fitting on the test set because the parameters can be tweaked until the estimator performs optimally. A solution to this problem is a procedure called cross validation. The training set is split into k smaller sets (other approaches are described below, but generally follow the same principles). I have chosen in the command to compute the score 5 consecutive times (with different splits each time).

### E. *Grid search*

I have tuned the grid search in order to identify the best parameter in the classifier. I did the range from 30 to 40 with step of 1 as a fast trial and it wasn't good enough for the classifier. I expanded the search range from 1 to 300 with 1 step to search in much more parameters. It is an exhaustive search over specified parameter values for an estimator.

## IV. Results and risks

The risk which may cause problems to that model is that the if the data is more larger than the given testing data set it will be more accurate for the vectorizer function to do more wider dictionary or index of the most common words or characters which will give the classifier better evaluation.

The classifier is then evaluated by a certain formula

$$m_{acc} = \frac{n_{TP} + n_{TN}}{n_{TP} + n_{TN} + 10n_{FP} + n_{FN}} \quad (1)$$

After running the code several times with different trials of either grid search or the scores of the cross validation as well. It gives results from 70 to 75 percentages, but the highest accuracy is achieved under the condition of 1 to 300 searched estimators in which the best n = 77 and scores range (5 slots).

## V. conclusion

The aim of this work is to implement a certain features with different parameters to a classifier and identify how it will work with a given data set. To do so we made use of some algorithms, that are supposed to give the accuracy of the classifier on the testing data. Vectorization using Tfidf and subsequent classification on the test set gave the results of the interval 0.70-0.75.

## VI. References


1. scikit-learn
2. Dive into python , Mark Pilgrim
3. Python for engineers , Shantu Tiwari