# Spam Filter

Kakhanouski Aliaksandr

Chicago

December 2019

## Goal

Use machine learning techniques to determine whether an email is spam or not by looking at the email body. Additional techniques, like email header validation can be used to improve classification accuracy, but for the purpose of this project we will concentrate only on email content.

## Dataset

For this project we use SpamAssassin Public Corpus that was downloaded from the following website: http://spamassassin.apache.org/old/publiccorpus/, where file *20050311_spam_2.tar.bz2* contains 1397 spam messages and file *20030228_easy_ham_2.tar.bz2* contains 1400 non-spam messages.

## Preprocessing

One of the most important steps is the preprocessing. To get better feeling we decided to write our own pre-processor and tokenizer instead of using nltk methods. Overview of performed steps:

1. Lower-case
2. Strip HTML tags
3. Normalize URLs
   All URLs are replaced with the text “httpaddr”
4. Normalize Email Addresses
   All email addresses are replaced with the text “emailaddr”
5. Normalize Numbers
   All numbers are replaced with the text “number”
6. Normalize Dollars
   All dollar signs ($) are replaced with the text “dollar”
7. Word Stemming
   Words are reduced to their stemmed form. For example, “discount”, “discounts”, “discounted” and “discounting” are all replaced with “discount”)
8. Remove non-words
9. Vectorize text using Bag of Words algorithm.
10. Add weights to words using TF–IDF algorithm.
    This will help to filter out common terms (words).

**Classifier**

There are several classifiers that can be a good choice for spam classification task - Random Forest, Naive Bayes, Support vector machine (SVM), Logistic Model and Decision Tree. To simplify and speedup project process we decided to try SVM classifier. We could also create a sklearn Pipeline that will try all possible classifiers with different hyperparameters.

**Data split**

Because we use Cross-Validation split validation we split our dataset into training and test sets. 70% training set and 30% test set.

**Scoring**

We have slightly more non-spam emails, so we decided to use F1 score as a more general technique that works well even for unbalanced datasets.

**Grid Search**

To get the best model we used sklearn Pipeline with GridSearch functions and ran it with the following parameters:

- Data reduction - PCA
- N of PCA components - [100, 300, 500]
- Classifier - LinearSVC
- SVC param C - [1, 10, 100, 1000]
- Max number of iterations - 2000
- Scoring - F1
- Cross-validation split folds - 5

### Results

When we ran our program we get the following results:

```
==> Grid scores on development set:
0.975 (+/-0.011) for {'pca__n_components': 100, 'svm__C': 1}
0.977 (+/-0.013) for {'pca__n_components': 100, 'svm__C': 10}
0.982 (+/-0.010) for {'pca__n_components': 100, 'svm__C': 100}
0.972 (+/-0.015) for {'pca__n_components': 100, 'svm__C': 1000}
0.979 (+/-0.010) for {'pca__n_components': 300, 'svm__C': 1}
0.984 (+/-0.009) for {'pca__n_components': 300, 'svm__C': 10}
0.978 (+/-0.007) for {'pca__n_components': 300, 'svm__C': 100}
0.971 (+/-0.011) for {'pca__n_components': 300, 'svm__C': 1000}
0.980 (+/-0.007) for {'pca__n_components': 500, 'svm__C': 1}
0.982 (+/-0.013) for {'pca__n_components': 500, 'svm__C': 10}
0.974 (+/-0.012) for {'pca__n_components': 500, 'svm__C': 100}
0.971 (+/-0.009) for {'pca__n_components': 500, 'svm__C': 1000}

==> Best parameters set found on development set:
{'pca__n_components': 300, 'svm__C': 10}

==> F-1 score for the TEST set with the best model: 0.977
```

From the program output we see that the best number of Principal Components is 300 and best C value for SVM classifier is 10. F-1 Score for the DEV set is 0.984, while for the TEST set it's 0.977. This is a very good result.