Spam Detection
Machine Learning for Natural Language Processing 2020

Selma Bouchta
ENSAE
selma.bouchta@ensae.fr

Maxime Berillon
ENSAE
maxime.berillon@ensae.fr

Abstract

The spam issue knows a huge interest since the last 20 years. It could represent more then 72% of the whole mailing or messaging trafic. The intrusive side apart, it can contain viruses or malware, which is why it is interesting and essential to detect it. In this project, we try to create a performing model and to explore more advanced models in the detection of spams.

All the code can be found here [1] and the GitHub repository is accessible here [2].

1 Problem Framing

The dataset we used is from the spam detection research used in several paper ((Almeida et al., 2011)): `https://www.dt.fee.unicamp.br/~tiago/smsspamcollection/`. This dataset holds a collection of SMS in english and their classification as a spam or a ham.

2 Experiments Protocol

We try to answer the following questions: **How can we treat text messages efficiently? What preprocessing can we do? What models can we use for this type of problem? How do we evaluate our models? What vectorizer makes the best scores? What variables can we add to the dataset? What is the most performing model?**

We start by the importation of the dataset, the exploration and preprocessing. We then train a first simple model SVM. After that, we try to create new variables from the text messages and train a Random Forest with these new variables and the SMS. The last trained models developped in this notebook are a RNN and a LSTM. Finally, for all of our models we do a quantitative evaluation and for our last model we do a qualitative evaluation.

[1] https://colab.research.google.com/drive/1VG88IMj-HkYe1MawuwgFEvKmEZT-JaIo?usp=sharing

[2] https://github.com/maximeberillon/Spam$_{Detection}$

3 Results

3.1 Exploratory Data Analysis

We then analyse the dataset the first thing we noticed was the great imbalance between the classes with 86% of **ham**. We therefore have a problem of imbalanced classes which must be taken into account in the models and in the choice of evaluation metrics. This is a classic case for problems of anomaly detection.

3.2 Models

The first model that is implemented is an SVM classifier. The idea of the SVM is to find a hyperplane in the space of variables to separate ”spam” from ”hams”. To obtain a space of variables we need to vectorize the sms and thus the words. We used two embedding methods for this:

- In a first step we train a *Word2Vec* on our own data thanks to the package *Gensim* [3];
- In a second step we use a pre-trained model to compare performances, here a pre-trained model on 6 billion Wikipedia words [4].

The second model studied is a Random Forest. We compare two functions that transform text into matrices:

- *CountVetorizer*: which converts a collection of texts into a matrix of token counts;
- *TfidfTransformer*: equivalent to CountVectorizer followed by TfidfTransformer. The latter transforms a count matrix into a tf-idf representation (the tf-idf allows the importance of a term contained in a document to be evaluated, relatively to a collection)

[3] https://code.google.com/archive/p/word2vec/)

[4] https://nlp.stanford.edu/projects/glove/

We tried to improve our model by adding text-related variables such as the length of the SMS, the number of punctuation marks, and the presence of long or short numbers in the messages. We visualise for each variable the distribution of ham and spam and observe that these are discriminating variables that should be added to the model.

Finally we further increase the complexity of the model and use neural networks. The best for this type of task are recurrent neural networks. We therefore studied a RNN and then an LSTM.

3.3 Performances

We can first see that the SVM shows very good results with high F1-scores (1). This sets the baseline quite high.

We then manage to increase the performance with the RandomForest and in particular the addition of variables.

Finally, the two neural networks show disappointing performances due to overfitting despite several attempts to reduce this phenomenon (addition of *dropout*, reduction of the number of layers, reduction of the number of neurons).

4 Discussion/Conclusion

To conclude, the aim of this project being to explore several methods, we have, through an SVM, a Random Forest, an RNN and an LSTM, tried to predict if an SMS was a spam or a ham. We observed that the best model in terms of performance is the Random Forest with TfidfVectorizer and the addition of discriminating variables. It would have been interesting to try other Deep Learning architectures to improve the scores of our latest models and avoid overfitting.

References

Tiago A Almeida, José María G Hidalgo, and Akebo Yamakami. 2011. Contributions to the study of sms spam filtering: new collection and results. In *Proceedings of the 11th ACM symposium on Document engineering*. pages 259–262.

	Accuracy		F1-Score		Precision		Recall	
	Test	Train	Test	Train	Test	Train	Test	Train
SVM + Word2Vec	0.955	0.962	0.840	0.869	0.778	0.817	0.913	0.928
SVM + GloVe	0.969	0.962	0.882	0.860	0.874	0.867	0.890	0.852
RandomForest + CountVectorizer	0.965	0.973	0.848	0.891	0.994	0.998	0.740	0.805
RandomForest + TfidfVectorizer	0.968	0.978	0.862	0.910	1.0	0.998	0.758	0.837
RandomForest + variables	0.982	0.985	**0.927**	0.942	0.981	1.0	0.878	0.891
RNN	0.836	0.951	0.170	0.782	0.252	0.975	0.128	0.653
LSTM	0.847	0.996	0.225	0.984	0.336	0.998	0.169	0.970

Table 1: Results of the different models