Spam detection

Filip Volarić

SW, FTN Novi Sad

Introduction

Spam detection is a machine learning problem possibly best explained with detecion of spam comments on Youtube. Youtube is a platform that billions of people use and express their opinion in the comment section. Sometimes people like to promote their product or youtube channel and it would be nice if this comments could be separated from those that might be helpful.

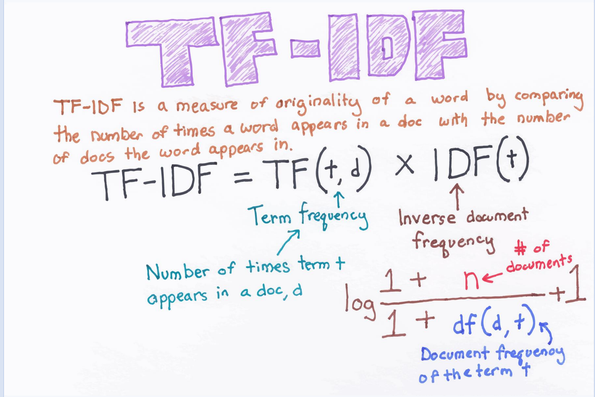
Data

Fist step in solving this problem is retreiving the data set that will be used for training and testing of model. Once data set is obtained we need to create models for text representation. Computers don't understand words in a way humans do, so we need to give these words numeric value. That is what Bag of words and TF-IDF models are for.

- Bag of words
  Bag of words is a simple model. It's a matrix NxM where M = number of unique words in data set and N = number of comments. One value in this huge matrix is interpreted as how many times does word M appear in comment N.

	about	bird	heard	is	the	word	you
About the bird, the bird, bird bird bird	1	5	0	0	2	0	0
You heard about the bird	1	1	1	0	1	0	1
The bird is the word	0	1	0	1	2	1	0

- TF-IDF
  Unlike bag of words tf-idf model does not value only single comment, rather takes account for all appearances in corpus of comments and creates a numeric value.Although rows and columns are same as in BOF numeric value is equal to TF * IDF (term frequency, inverse document frequency)



Algorithms

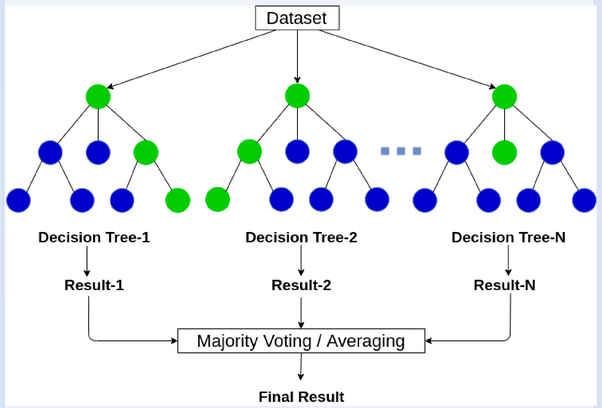
1. Naive Bayes

Naive bayes algorithm is one of the simplest algorithms in text classification yet very effective. We count probabilities that a word of a comment is in set of comments marked as spam and vice-versa for comments marked as ham.

$$P(si|T) = \prod t \in T\ P(t|si)\ P(t)\ P(si)$$

2.Random Forest

Random forest is an algorithm, rather ensemble of decision trees. Each tree is trained on random subset of the attributes. Each desition tree gives a result on test data set and final result is calculated based on majority voting.



Implementation

- Implementation started by loading data from csv files into dataframe.
- We split that dataframe into two, first containing only classes of comments(1-SPAM 0-HAM) and second one containing other information such as date, user and of course content of a comment.
- Next we split the acquired set into two, one for training(70%) and one for testing(30%).
- Then we created Bag of words and TF-IDF models with preprocessed text.
- In file main.py we used sklearn library but in spam_detection.py you can se my personal take on creating models.
- We created training and testing models for both Bag of words and TF-IDF representation.

Results

We converted dataset into 80 different trees and we fit the training set so that we can score its performance on the testing set. Using Random forest algorithm we got the output accuracy of 0.959(Bag of words) and 0.9522(TF-IDF). Here are confusion matrices for one run.

BOF

	Spam	Ham
Spam	300	14
Ham	10	263

TF-IDF

	Spam	Ham
Spam	299	17
Ham	11	260

Similarly, we fit previosly created training set so we can score performance on testing set. Using Naive Bayes algorithm we get the output accuracy of 0.91 for both Bag of words and TF-IDF. Here are confusion matrices for one run.

BOF

	Spam	Ham
Spam	267	9
Ham	43	268

TF-IDF

	Spam	Ham
Spam	268	9
Ham	42	268

Predicting input

To play with model a little bit more, we created four pipelines for predicting the class of input comment. Once user types a comment he gets result from each pipeline.

- Pipeline1 – Bag of words + Random forest
- Pipeline2 – TF-IDF + Random forest
- Pipeline3 – Bag of words + Naive Bayes
- Pipeline4 – TF-IDF + Naive Bayes

In most cases they print out same results, however in some weirdly constructed comments not meant to be spam, NB algorithm much more often than RFC predicts them as spam(False positive).

Conclusion

Acquired results are decent and with average accuracy od 95% using Random forest classifier it is safe to say that it is predicting very good.

Use of pipelines allowed for very simple comparison of classification with different text models and different algorithms.

With my own constructed models, we acquired a little bit worse accuracy -0.01 on average. However perfomance wise, creating of models is much slower than with sk-learn library, specially when it comes to text processing.

Also in file NaiveBayesClassifier.py is my own implementation of Naïve Bayes algorithm that managed to put out maximum accuracy of 88%, compared to consistent 90-91%. However that classifier has much room for impovement.