Machine Learning Engineer Nanodegree

Capstone Proposal

Ramkumar Singh
August 7th, 2017

ML Spam Filter

Domain Background:

The 1990s marked the beginning of internet with Hotmail being the first web-based email provider. Standards such as sender authentication, whitelisting, etc. were unknown. Marketers exploited this opportunity. As a result, our inboxes get flooded with junk emails every day.

In late 1990 to early 2000, with email abuse rising, ISPs started to use crude filters based on keywords, patterns or special characters. Blacklists began to emerge, listing the IPs of known bad senders. Misuse continued as the ISPs often still didn't have enough data to prove the claims of their senders to be legitimate and the process was still based on human decision

During the mid-to-late 2000s, next major development in email deliverability included the standardization of reputation scores. False positives were further reduced by adding sender authentication mechanisms like SPF and SenderID, and following them also Domainkeys and DKIM (Domainkeys Identified Mail).

2010 onwards we have seen important changes to email spam. Instead of relying on their own judgement, email clients have put the power firmly in the hands of their users. This in turn has given ISPs the opportunity to embrace 'Big Data' and "Machine Learning", to record all the inbox actions of all their users to determine what really constitutes "wanted" or "unwanted" emails. For example below are the few action which could be use to measure a sender's engagement:

- Moving an email to spam or trash is a negative reaction of a recipient and reflects badly on the sender and help in classifying the mail content as spam.
- Replying or forwarding shows positive engagement.
- Adding the sender to the address book is the best indication of all that an email is genuinely wanted.

The data collected through above feedback is used to further train the model and hence making the Spam Filter more efficient.

Problem Statement:

The project is to implement a Spam Filter with the help of a ML classifier which would classify a given mail as spam or ham.

Data Source :

Apache SpamAssasin public corpus has spam and ham mails data. Below is the overview of data

URL:

- http://spamassassin.apache.org/old/publiccorpus/

Content Folder:

- spam: 500 spam messages, all received from non-spam-trap sources.
- easy_ham: 2500 non-spam messages. These are typically quite easy to differentiate from spam, since they frequently do not contain any spammish signatures (like HTML etc).
- hard_ham: 250 non-spam messages which are closer in many respects to typical spam: use of HTML, unusual HTML markup, coloured text, "spammish-sounding" phrases etc.
- easy_ham_2: 1400 non-spam messages. A more recent addition to the set.
- spam_2: 1397 spam messages. Again, more recent.

Total count: **6047** messages, with about a **31%** spam ratio.

Solution Statement:

To create solution of the above problem we would be performing below steps:

- Pre-processing the mail - In this step we will process the mail content with following changes:
    - Remove html tags
    - Normalize numbers, Urls, email address and Dollar sign (Replace 0-9 with ‘number, hyperlinks with 'httpaddr', any email address with 'emailaddr' and $ with 'dollar')
    - Tokenize the words
    - Take only alphanumeric words
    - Lemmatize and change case to lower to reduce the words to their stemmed form.
    - Filter stop words (words which do not have significance like : to, the, a etc)

- Extracting the features: Create a training set from above processed mail. From this set calculate frequency of each word and select high frequency word as feature for be used for learning.
- Training different classifier: Train the classifiers like Logistic Regression, K-Neighbour Classifier, SVM , Naïve Bayes etc
- Evaluating the classifiers: The above classifiers will be evaluated against the dummy classifier, by comparing their prediction time and f1score.
- Fine Tune the best classifier : Use GridSearch technique to fine tune the best classifier using different parameters like C, gamma and other applicable parameters

Benchmark Model :

In this project we will use the sklearn dummy classifier as a simple baseline to compare with. DummyClassifier is a classifier that makes predictions using simple rules, so the goal is to perform better that this.

Evaluation Metrics:

In real world scenario volume of ham males are generally higher that the spam mails. For such problem f1score is a better metric to analyse performance of a classifier. In the given dataset also the spam ratio is 31% only, so f1score is applicable for the given dataset as well. The solution will evaluate f1score of ML model against the benchmark f1score

Project Design:

Implement a supervised learning classifier model using the SpamAssasin public corpus data to train and test the model. Project design would consist of following sections

1. Data Pre Processing
    a. Remove html tags and Normalize numbers, Urls, email address and Dollar sign with regex rules
    b. Tokenize the words using nltk word_tokenize
    c. Change to lower case and take only alphanumeric words.
    d. Lemmatize the words to their stemmed form using nltk WordNetLemmatizer
    e. Filter stop words using nltk.corpus stopwords

2. Data Visualization: using matplotlib and WordCloud
    a. Plot spam mail words and its frequency using histogram and word cloud
    b. Plot ham mail words and its frequency using histogram and word cloud

3. Train- Test Split : Split Data in Training and Testing Set using sklearn.model_selection train_test_split
4. Feature Extraction: From the training set take high frequency spam and ham mail words to be used as feature.
5. Create Feature Data : using the processesed mail (in step 1) and extracted feature (in step 4), create feature data for train and test set
6. Train different classifiers : Train different classifiers like Logistic Regression, K-Neighbour Classifier, SVM , Naïve Bayes etc
7. Evaluate classifiers and create evaluation matrix : The above classifiers will be evaluated against the dummy classifier, by comparing their prediction time and f1score.
8. Fine Tune the best classifier: Use GridSearch technique to fine tune the best classifier using different parameters like C, gamma and other applicable parameters