NATURAL LANGUAGE PROCESSING

UML602

SMS Spam classifier

Submitted by:

Himanshu Amethia 101611023

Submitted to:

Mrs. Ashima Sharma

THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

Computer Science & Engineering Department
Thapar Institute of Engineering & Technology, Patiala-147004

Jan –May 2019

1.0 Introduction

The use of mobile phones has skyrocketed in the last decade leading to a new area for junk promotions from disreptable marketers. People innocently give out their mobile phone numbers while utilizing day to day services and are then flooded with spam promotional messages.

In this Report we will take a look at classifying SMS messages using the TF-IDF algorithm for feature extraction Naive Bayes Machine Learning model, will be used for classification of messages

Need of System

The proposed filter detects and filters out SMS spam messages in a smart manner rather than black/white list approaches that require intervention of phone users. SMS spam messages are automatically filtered out without disturbing the phone user. The proposed detection scheme is evaluated on a large SMS message dataset consisting of spam and legitimate messages.

Application

The idea is to create an independent Software program that helps consumers get a better understanding about the SMS they are reading on various platforms. We examine SMS on the platforms, and then create a report for each SMS by running and an analysis on them. Our reports help inform you about the nature of the SMS. Our tool analyzes SMS and helps improve your experience of distinguish between ham messages and spam messages

2.0 Existing Work

SMS spam detection is comparatively a new research area than email, social tags, and twitter and web Spam detection. There are several established email spam detection techniques. SMS spam detection technique has some challenges over email spam detection. such as restricted message size, use of regional and shortcut words and limited header information. These challenges need to be solved. There is scope of research in this field and some research works have been conducted on it. There are different categories of SMS spam filtering such as white listing and black listing, content-based, non-content based, collaborative approaches and challenge-response technique Several Machine Learning Algorithms such as Naïve Bayes, Support Vector Machine (SVM), Logistic Regression, Decision Trees, K-Nearest Neighbor are used to classify between Spam and legitimate SMS named as Ham.

3.0 Working of the Proposed System

The approach followed in this project involves data preprocessing, data cleaning where we will remove the stopwords from our dataset as they do not play much role in the classifying of messages The model is a Multinomial Navies Bayes classifier which is used to predict the class of message whether it is ham/spam of the text data. The text data is initially cleaned and tokenized for appropriate analysis and then Tfidf Vectorizer is used to convert the text tokenized data into feature vectors so that the classifier can deal with it.

4.0 Data Collection and Data Preparation

Cleaning of data, preprocessing

For any machine learning model raw data is of no use. So, it is imperative to preprocess data before applying any model to it. For this project we need to clean, preprocess the data as for dataset stopwords is not of any use so before apply any machine learning algorithm we need remove stopwords So preprocessing was very important to extract some relevant information from it. Initially we used regular expressions to remove unwanted space breaks, html tags etc. so that we can apply natural language processing on it to make it machine understandable. Here we use NLTK.

NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries, and an active discussion forum.

Thanks to a hands-on guide introducing programming fundamentals alongside topics in computational linguistics, plus comprehensive API documentation, NLTK is suitable for linguists, engineers, students, educators, researchers, and industry users alike. NLTK is available for Windows, Mac OS X, and Linux. Best of all, NLTK is a free, open source, community-driven project.

NLTK has been called “a wonderful tool for teaching, and working in, computational linguistics using Python,” and “an amazing library to play with natural language.”

Natural Language Processing with Python provides a practical introduction to programming for language processing. Written by the creators of NLTK, it guides the reader through the fundamentals of writing Python programs, working with corpora, categorizing text, analyzing linguistic structure, and more.

Firstly we remove stop words. In computer search engines, a stop word is a commonly used word (such as "the") that a search engine has been programmed to ignore, both when indexing entries for searching and when retrieving them as the result of a search query. When

building the index, most engines are programmed to remove certain words from any index entry. The list of words that are not to be added is called a stop list. Stop words are deemed irrelevant for searching purposes because they occur frequently in the language for which the indexing engine has been tuned. In order to save both space and time, these words are dropped at indexing time and then ignored at search time.

This is followed by stemming. Stemming is the process of reducing a word to its word stem that affixes to suffixes and prefixes or to the roots of words known as a lemma. Stemming is important in natural language understanding and natural language processing.

Now once data has been cleaned we apply TF-IDF model to select the keywords.

5.0 Training of the Model

The model is trained on a data-set of SMS spam collection which was downloaded from UCI machine learning . It has around 5.5k SMS of mobile phones For the purpose of our model, we converted the spam/ham collection. Into 1/0 numeric value The data is split in a ratio of 80:20 for training and testing purposes

6.0 Testing of the Model

As mentioned above, the SMS spam collection dataset was used for making the model and split into a 80:20 ratio for training and testing respectively. The classifier finally results in an accuracy of 0.9766 for the test data.

7.0 Results and Discussions

Using the naive classifier we achieved a final prediction accuracy of 97.66%. This means that when we fed the 1114 SMS of the test dataset to our code, we were able to classify about 1088 of them correctly.

Cleaning and parsing of data:

For eg.:

Before cleansing:

Go until jurong point, crazy. Available only .

After cleansing :

-Go go jurong point crazi avail

After cleansing spam and ham message are separated through spam classifier i.e

Rating of a product:

```
pred
```

```
array([1, 1, 1, ..., 1, 0, 1])
```

```
import numpy as np
actual=np.array(y_test)
```

```
actual
```

```
array([1, 0, 1, ..., 1, 0, 1], dtype=object)
```

8.0 Conclusions and Future Scope

Nowadays, SMS classifier is a hot topic in machine learning. In this project we tried to show the basic way of classifying messages into spam or ham category using Multinomial Naive Bayes as baseline and how language models are related to the Naive Bayes and can produce better results. We could further improve our classifier by trying to extract more features from the messages, trying different kinds of features, tuning the parameters of the naive Bayes classifier, or trying another classifier all together.

9.0 References

1. https://www/kaggle.com

2. http://scikit-learn.org/stable/modules/feature_extraction.html#the-bag-of-wordsrepresentation

3. https://en.m.wikipedia.org/wiki/Tf-idf