International Journal of Scientific Research in Computer Science, Engineering and Information Technology
ISSN : 2456-3307 (www.ijsrcseit.com)
doi : https://doi.org/10.32628/CSEIT2173151

Spam Text Detection

Dharmik Timbadia, Niraj Vesaokar
Computer Engineering, Rajiv Gandhi Institute of Technology, Mumbai, Maharashtra, India

Article Info
Volume 7, Issue 3
Page Number: 698-704

Publication Issue :
May-June-2021

Article History
Accepted : 12 June 2021
Published : 20 June 2021

ABSTRACT

Spam Detection is the process to classify text which contains irrelevant or unsolicited messages sent over the internet, typically to a large number of users, for the purposes of advertising, phishing, spreading malware, etc. Text summarization is the technique of converting long text to short. The intention is to make a coherent and fluent summary having only the most points outlined within the document. A USA based machine learning expert which had 13 years of experience and currently teaches people his skills, states his technique has proved to be critical in quickly and accurately summarizing voluminous texts, something which might be expensive and time-consuming if avoided machines. Machine learning models are usually trained to know documents and distil the useful information before outputting the specified summarized texts.

Keywords: Spam Detection, unsolicited commercial email, SMS, NLP

I. INTRODUCTION

Spam filtering is that the process of detecting unsolicited commercial email (UCE) messages on behalf of a private recipient or a gaggle of recipients. Machine learning applied to the present problem is employed to make discriminating models supported by labelled and unlabeled samples of spam and non spam. Such models can serve populations of users (e.g., departments, corporations, ISP customers) or they will be personalized to reflect the judgments of a private. a crucial aspect of spam detection is how textual information contained within the email is extracted and used for discrimination.

II. LITERATURE REVIEW

In the literature, the review spam has been divided into three groups, which was proposed by Dixit et al. [2]: (1) Untruthful Reviews -- the most concern of this paper, (2) Reviews on Brands -- where the comments are only concerned with the brand or the vendor of the merchandise and fail to review the merchandise, and (3) Non-Reviews -- those reviews that contain either unrelated text or advertisements. the primary category, untruthful reviews, is of most concern as they undermine the integrity of the online review system. Detection of type 1 review spam may be a challenging task because it is difficult, if not impossible, to differentiate between fake and real reviews by manually reading them. for instance the problem of this task, we consider a true & fake example from the dataset created by Ott et al. [3]. As

a person's judge, it's difficult to confidently ascertain which review is fake and which is authentic.

There are not any clear indications or signals from the text of the 2 reviews that illustrate to the casual reader that the primary review is real while the second is fake. Nevertheless, guides provided by the ConsumeristFootnote2 and

MoneyTalksNewsFootnote3 websites offer tips to assist consumers spot fake reviews. A scientist might seek to utilize this logic when training data processing and machine learning algorithms to seek out these features within the review which will determine if it's real or fake.

III. PROPOSED SYSTEMS

A.Analysis/Framework/Algorithm:

We aim to use text classification on unstructured text present in SMS and check for spam with the assistance of NLP which has proven to be an excellent alternative to structure textual data in a fast, cost-effective, and scalable way.

IV. DETAILED DESIGN

Like the initiative, we import all the required libraries we'd like to create the model.

NLTK- Natural Language Toolkit may be a leading platform for building Python programs to figure with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources like WordNet, along side a collection of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries.

sklearn- Scikit-learn is perhaps the foremost useful library for machine learning in Python. The sklearn library contains tools for machine learning and statistical modelling including classification, regression, clustering and dimensionality reduction.

The loading dataset part follows after this step. the information set we'll be using comes from the UCI Machine Learning Repository. It contains over 5000 SMS labelled messages that are collected for mobile spam research. After loading the dataset we extract some useful information like the column information & class distributions.

The "pre-processed" subdirectory contains the messages within the pre-processed format. Each message is during a separate document. the amount at the start of every filename is that the "order of arrival".

The 'raw' subdirectory which have the messages in their original form. Spam messages in non-Latin encodings and ham messages sent by the owners of the mailboxes to themselves (sender in "To:", "Cc:", or "Bcc" field), and also some couple of virus-infected messages are removed, but no other modification has been made. The messages within the "raw" subdirectory are quite the corresponding messages within the "pre-processed" subdirectory, because: (a) duplicates are preserved within the "raw" form, and (b) during the pre-processing, ham and/or spam messages were randomly sub sampled to get the specified ham: spam ratios.

V. FEATURE EXTRACTION

As a first feature selection step, we ignore any tokens that appear in but 5 different training messages. Information Gain scores are then computed for the remaining tokens (in their Boolean form) as in previous work [5, 1], and therefore the 3000 of them with the very best scores are utilized in the feature vectors of the messages.

The Algorithms we used are stated below:

K Nearest Neighbors: KNN are often used for both classification and regression predictive problems. Though, it's widely utilized in classification problems

within the industry, to judge any technique we generally consider 3 important aspects they are:

1. Ease to interpret the output
2. Calculation time
3. Predictive Power

K Nearest Neighbors: KNN are often used for both classification and regression predictive problems. Though, it's widely utilized in classification problems within the industry, to judge any technique we generally consider 3 important aspects they are:

1. Ease to interpret the output
2. Calculation time
3. Predictive Power

Random Forest: Random forest is a supervised learning algorithm that is used for both classification as well as regression. However, it is mainly used for classification problems. Random forest algorithm creates decision trees on data samples & then gets the prediction from them then finally selects the best solution through voting. It is an ensemble method that is better than a single decision tree because it reduces over-fitting by averaging the result.

Logistic Regression: Logistic regression is a supervised learning algorithm that is mostly used to solve binary classification tasks. The basis of logistic regression is that the logistic function also called the sigmoid function, which takes in any real-valued number and maps it to worth between 0 and 1. The logistic regression model takes a linear equation as input and use logistic function and log odds to perform a binary classification task. Before going into detail on logistic regression, it is better to review some concepts in the scope of probability.

SGD Classifiers: Stochastic Gradient Descent Classifier is a linear classifier (Support Vector Machines, logistic regression, a.o.) optimized by the SGD. These are two different concepts. While SGD is an optimization method, Logistic Regression or linear SVM may be a machine learning algorithm, you'll think that a machine learning model defines a loss function, and therefore the optimization method minimizes/maximizes it.

Naïve Bayes: Naive Bayes classifiers are a group of classification algorithms supported by Bayes' Theorem. It's not one single algorithm but a family of algorithms where all of them share a standard principle, i.e. every pair of features being classified is independent of every other. Naive Bayes classifiers are heavily used for text classification and text analysis machine learning problems.

VI. MODEL OVERVIEW

Input: Any sentence in English.

Output: It displays whether the given text is spam text or not.

A variety of approaches to spam filtering are utilized in the past. Machine learning classification algorithms have proven to perform alright during this task. Naive Bayes (nb) classifiers especially are found to perform reasonably well, despite their simplicity. The simplicity of nb classifiers and their low computational requirements have made them particularly appealing for commercial spam filters. However, commercial filters rely also on a spread of other indicators, so as to enhance their spam detection performance. Our purpose here is to place simple Naive Bayes text classifiers to a sensible test against more elaborate filters. Thereby, we expect to realize a sign of the potential contribution of nb text classifiers to spam filtering.

We compared variants of nb on the task of spam filtering, using also a spread of datasets. In those experiments, we assessed not only the performance of the classifiers but also their computational efficiency, which is important for real-time adaptive filtering. Based on that, we have chosen to test two variants of nb: multinomial nb with transformed term frequency attributes (to-nb), and multinomial nb with Boolean

attributes. Our first choice (to-nb), which achieved the best performance, uses a transformation of attributes that are based on the paper of Rennie et al. Our second choice (bool-nb) has proven to perform quite well, despite its simplicity, both in our experiments and in related work.

We need to prepare data for further processing. This intermediate preparation stage is called a Preprocessing step which is a structured representation of the original text. In the preprocessing step of our proposed technique the input text obtained from the text file first split into sentences using segmentation method, then sentences are further split into words using tokenization and then stopwords are removed to clean the original text. Our proposed pre- processing step consist of four steps

1) Lemmatizer
2) Tokenization
3) Stop-Words Removal

1) Lemmatizer:

Lemmatization is that the process of converting a word to its base form. The difference between stemming and lemmatization is, lemmatization considers the context and converts the word to its meaningful base form, whereas stemming just removes the previous couple of characters, often resulting in incorrect meanings and spelling errors.

2) Tokenization:

In tokenization, the sentences are uneven into discrete bits or tokens(words). It omits certain characters, like punctuation, spaces and special symbols between words. Punctuations (Viram Chinha) within English consists of question mark (?), full stop(.), exclamation mark(!)etc.

3) Stop-Words

Removal Stop-Words include function words, articles, prepositions, conjunctions, prefix, postfix, etc. i.e. common words that carry less significant meaning than keywords. So these sorts of words should be far away from the input text document, otherwise, the sentence having ore no of stop-words could have a better weight. We analyzed that each Hindi text document contains a minimum of 25-30%. Also, they create the text to look heavier and are insignificant. Hence should be eliminated.

Naive Bayes variants used:

Each message j is represented as a vector hx1j,...,xmji, where xij is the value of attribute Xi in message j, and m is the number of tokens we use as attributes (3,000). In bool-nb, xij = 1 if the token that corresponds to attribute Xi occurs in the message; otherwise xij = 0. For tf-nb, the value of xij is initially equal to the frequency (number of occurrences) of the corresponding token in message j. Thereafter, it is transformed following the next three steps:

xij ← log(xij + 1) (tf transform),
xij ← xij ·log(Pk 1 Pk δik) (idf transform),
xij ←xij qPl (dlj)2 (length normalization),

where k ranges over the training messages, δik is 1 if the token that corresponds to attribute Xi occurs in message k and 0 otherwise, and dlj is the initial (before the first transformation) value of xlj. These steps are fully explained by Rennie et al. [4]. It is worth mentioning that we do not use the weight normalization that Rennie et al. propose, as it led to worse results in our previous experiments.

From Bayes's theorem, the probability of message j with vector ˜x = hx1j,...,xmji to belong in category c is:

P(c|˜x) = P(c)·P(˜x|c) P(˜x)

In order to classify message j in the ham (ch) or spam (cs) category, we use the following formula:

scorej = P(cs)·P(˜x|cs) P(cs)·P(˜x|cs) + P(ch)·P(˜x|ch)

which indicates how sure our filter is that message j is spam.

For values close to 1, we are very confident that it is spam; for values near 0, we are very confident it is ham. Therefore, we can introduce a threshold on scorej in order to make the final decision about the message's category. The value of the threshold controls the tradeoff between false positives and false negatives. In the ceas challenge, we set the threshold to 0.5, i.e., we clasify message j as spam if scorej > 0.5. Our filter, however, also returns scorej, making it easy to experiment with other threshold values. The probability P(c) is estimated by dividing the number of training messages of category c by the the total number of training messages.

The probability P(˜x|c) is estimated as Qm i=1 P(ti|c)xij, where xij is computed as described above for each form of nb, and ti is the token that corresponds to attribute Xi.

For bool-nb, P(t|c) is estimated as: p(t|c) = 1 + Mt,c 2 + Mc ,

where Mt,c is the number of training messages in category c that contain token t, and Mc is the total number of training messages of category c. For tf-nb, xij is estimated as:

p(t|c) = 1 + Nt,c m + Nc

where Nt,c is the number of occurrences of token t in the training messages of category c, and

Nc =Pm i=1 Nti,c.

The two filters that we submitted to the competition were already trained on 500 messages that we picked randomly from the SpamAssassin corpus; the latter is provided with the trec 2006 Spam Evaluation Kit. We maintained in the 500 messages the spam to ham ratio of the SpamAssassin corpus. We also created a different sample (dubbed Active Learning Set) of 500 messages from the SpamAssassin corpus, with the same spam to ham ratio, in order to use it in the active learning procedure that is described below.

Active Learning:

During the active learning, category labels for the incoming messages are only available to the filter upon request. Each filter is allowed a fixed number Tr of requests per run; it is also given the total number Cl of incoming messages it will have to classify during the run.

For each incoming message, the filter has to decide if it will request the message's true category to be revealed, so that the message can be used for training, or not. We define a ratio K equal to:

K = Tr Cl

This is the ratio of incoming messages that we want to be used for training. The main idea behind our method is that the probability of an incoming message being useful for training is proportional to the uncertainty of our filter about the message's category. It should be stressed here that all incoming messages could potentially be selected by our active learning method as training examples (i.e., have their true categories revealed), some with larger probability than others. This additional randomness addresses to some extent potential mistakes when assessing the usefulness of the incoming messages as training examples. We assume that the probability of a message j to be useful for training follows a normal distribution over scorej with mean μ = 0.5 (where the classifier's uncertainty is highest) and standard deviation σ.1 In other words, scores near μ = 0.5 are more likely to correspond to useful training messages. We use K to set the standard deviation of the distribution.

Initially, we use a sample of TM (500) messages from the Active Learning Set , in order to select the value of σ so that T of the TM messages have scores within one standard deviation from μ = 0.5, where T is estimated as: T = round(TM ·K).

This way, the interval from μ–σ to μ+σ contains the scores of the sample's messages that we would have wanted to have been selected for training, i.e., the T messages with scores closest to μ = 0.5. Note that setting σ so that the scores of the T messages fall within μ–3·σ to μ+3·σ would guarantee that the probabilistic selection that we use (described below) would almost always pick messages whose scores fall within the interval that contained the scores of the T messages, provided that the messages of the contest follow the normal distribution we assume. Instead, by setting σ so that the scores of the T messages fall from μ–σ to μ + σ, we allow the probabilistic selection to use (with lower probability) as training examples messages whose scores are outside the interval that contained the T messages of the sample. Provided that the total number of selected training examples has not already reached Tr, we select an incoming message with scorej as a training example with the following probability:

P(train|scorej) =(cdf(scorej;μ,σ),
if
scorej ≤ 0.5 cdf(1–scorej;μ,σ),
otherwise

where cdf(x;μ,σ) is the cumulative distribution function of the normal distribution, μ = 0.5, and σ is estimated as above. Formula 10 assigns the same selection probability to messages whose scorej is at the same distance from μ = 0.5, regardless of whether scorej is smaller or larger than 0.5; furthermore, the probability increases as scorej approaches 1Note that scorej ∈ [0,1], whereas the normal distribution that we assume assigns non-zero probability to values of scorej in (–∞,0) and (1,+∞). For simplicity, we overlook this mismatch.

VII. RESULTS AND DISCUSSIONS

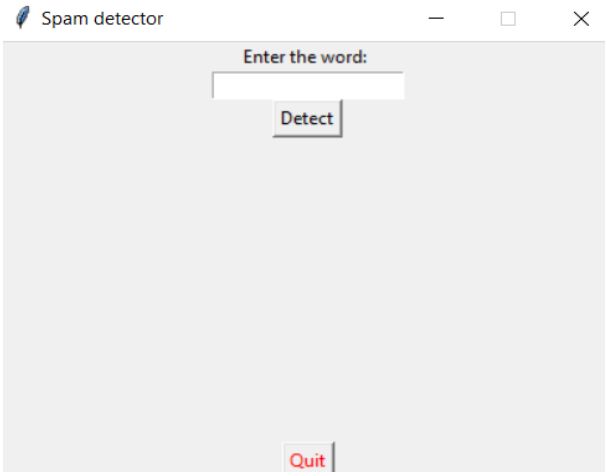

Fig.1

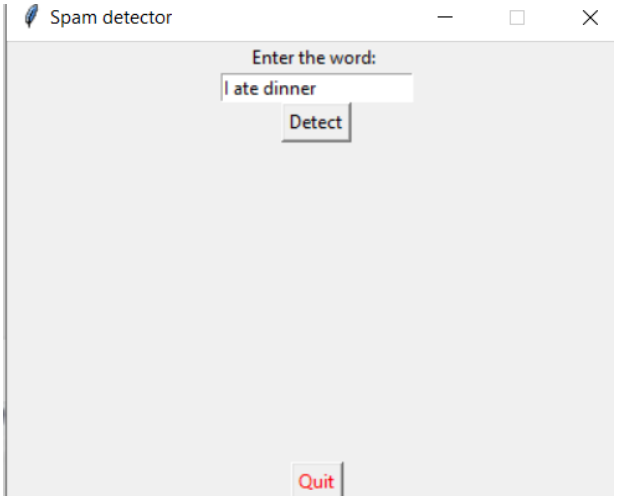

Fig.2

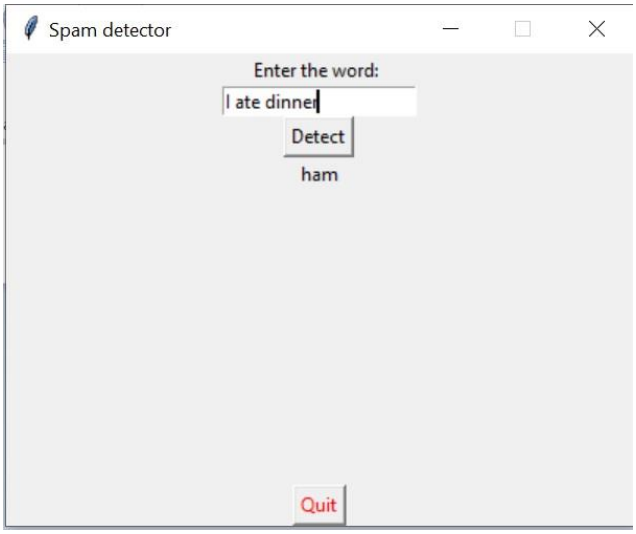

Fig.3

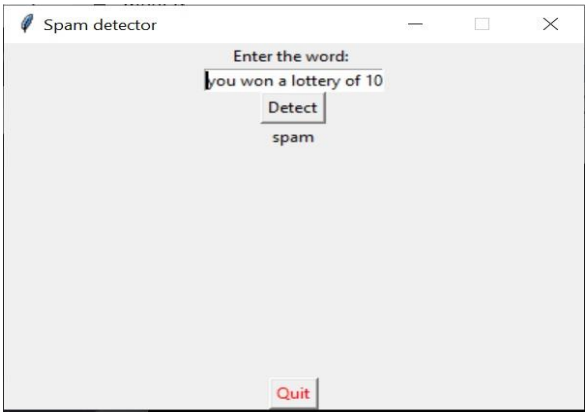

Fig.4

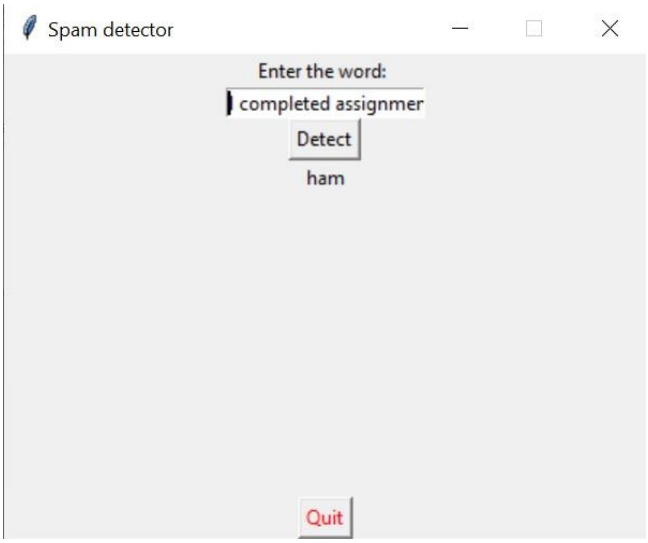

Fig.5

VIII. CONCLUSION

In this short paper we described briefly two email spam filters that employ different forms of the Naive Bayes classifier and focus on the text of the messages. The main criteria for the choice of the two Naive Bayes forms were their good performance in a series of experiments with different data sets and their computational efficiency. The ultimate goal of this effort is to measure the value added by non-textual features and more elaborate classifiers, by comparing our simple text classifiers. Our immediate plans are to study the results of our filters and their competitors and draw potentially interesting conclusions about the spam filtering process. Additionally, we can work on providing our overall filter as a plug-in for a well known free email client, in order to allow measuring the filter's effectiveness under real circumstances, by real users.

IX. REFERENCES

[1]. I. Androutsopoulos, J. Koutsias, K. Chandrinos, and C. Spyropoulos. An experimental comparison of Naive Bayesian and keyword-based anti-spam filtering with encrypted personal e-mail messages. In 23rd ACM SIGIR Conference, pages 160–167, Athens, Greece, 2000.

[2]. A. McCallum and K. Nigam. A comparison of event models for Naive Bayes text classification. In AAAI Workshop on Learning for Text Categorization, pages 41–48, Madison, Wisconsin, 1998.

[3]. V. Metsis, I. Androutsopoulos, and G. Paliouras. Spam Filtering with Naive Bayes – Which Naive Bayes? In Proceedings of 3rd Conference on E-mail and Anti-Spam (CEAS 2006), Mountain View, CA, USA, 2006

[4]. J. D. M. Rennie, L. Shih, and D. R. Karger. Tackling the Poor Assumptions of Naive Bayes text Classifiers. In 20th International Conference on Machine Learning, Washington DC, 2003.

[5]. M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz. A Bayesian approach to filtering junk e-mail. In Learning for Text Categorization – Papers from the AAAI Workshop, pages 550–62, Madison, Wisconsin, 1998

[6]. K.-M. Schneider. A comparison of event models for Naive Bayes anti-spam e-mail filtering. In 10th Conference of the European Chapter of the ACL, pages 307–314, Budapest, Hungary, 2003

Cite this article as :

Dharmik Timbadia, Niraj Vesaokar, "Spam Text Detection", International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT), ISSN : 2456-3307, Volume 7, Issue 3, pp.698-704, May-June-2021. Available at
doi : https://doi.org/10.32628/CSEIT2173151
Journal URL : https://ijsrcseit.com/CSEIT2173151