AN ADAPTIVE CLASSIFICATION APPROACH TO FILTER SPAM E-MAIL USING VECTOR SPACE MODEL

Nakul Dave*

Uttam Chauhan**

Avani Dave***

Abstract

The majority of previous studies of data mining have been concentrate on structured data, such as relational, transactional and data warehouse data. But, in actuality, an important section of the available information is stored in text databases, which consist of large collections of web documents from various sources, such as news articles, research papers, e-books, digital libraries, e-mails, and Web pages. Moreover, It is in increasing phase and in magnitude of terabytes of size. Among the ample of provisions of internet, e-mail facility is very useful and broadly used. Spam email is the strongly attached issue with email provision. Among various approaches developed to stop spam emails, filtering is an important and popular one. In this paper, to categorize spam and non-span email which arrives to our email id, classification method-KNNC Classification can work for better accuracy using Vector Space Model in adaptive manner. For getting accuracy in spam classification we have used two dataset- personal & Ling Spam Corpus(Lemm dataset) and apply KNNC Classification on them. We got nearly 95% of precision in spam & 86.6% of precision in nonspam and got 83% of accuracy using personal dataset and 80% using Lemm dataset using adaptive approach. We propose our own solution by reviewing the result and related work that adaptive approach using vector space model in KNNC classification method is efficiently provide better accuracy for filtering the spam mail for both smaller and larger dataset.

Key words: Spam, Vector Space Model, KNNC-Classification

* **Gujarat Technological University,** Government Engineering College, Modasa.

** **Department Of Computer Engineering,** Vishwakarma Government Engineering College Chandkheda.

*** **Gujarat Technological University,** Kalol Institute Of Technology & Research Center.

Introduction

Today most of the data-mining researches predict that the information to be “extracted” is already in the form of a relational database. Unluckily, for many applications, available electronic information is in the form of unstructured natural-language documents rather than structured databases. Spam can be definite as unsolicited (unwanted, junk) email for a recipient or any email that the user do not wanted to have in his/her inbox. It is also defined as “Internet Spam is one or more unsolicited messages sent or posted as a part of larger collection of messages, all having substantially identical content [1]". There are severe problems from the spam mails, e.g.; wastage of network resources (bandwidth), wastage of time, damage to the PC’s & laptops due to viruses & the ethical issues such as the spam emails advertising pornographic sites which are harmful to the young generations [1]. The technique for removing spam manually includes; For example, use of white lists, black lists, and gray lists is straightforward. The difficulty with this approach is that the burden on the user can be considerable. For removing spam automatically from user's email, the most efficient task used is spam filtering[3]. Using text mining or machine learning methods which consider spam filtering as two-class text classification that classifies a message as spam or legitimate. Classification using K-NNC is works better with large data sets. Our aim is to evaluate the current state of research and propose own solution to the K-NNC algorithm with the high accuracy and precision for finding the efficient spam from the mail using not only the content of that spam email but also with its subject.

Methods & Results

We have evaluated our proposed system using two modules; Training module and Testing Module.

Training Module

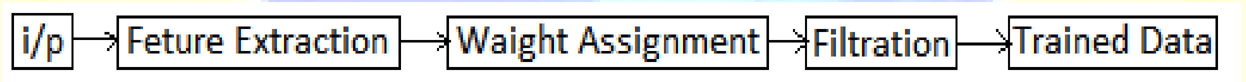


Figure 1 Training Module

In Training Module we will pass any mail and then one by one extract all features of that mail and calculate the weight(Global Weight, Local Weight, Total Weight) as well as probability and occurrence of each feature and based on that create the Vector Space model after filtration on them. Filtration is used for stopping & Stemming.

Feature Extraction: This module is divided into two submodules as Feature Identification and stemming.

Feature Identification: In this module, Mail dataset is given as input. Each E-mail message contains number of words.Features are identified from that E-mail and given it to next module.

Stemming: A single word can be represented by many ways, for example offer, offering, offers|| . This all should be stem and considered as a single word —offer. Stemming reduces the vector space by reducing set of word to one common word. Each word has its own importance in the mail and hence to assign the weight to the feature, features are passed to weight assignment module.

Weight Assignment: There are three components used in weighting scheme to calculate the total weight of feature:

$a_{ij} = g_i * t_{ij} * d_j$ (1)

where gi is global weight of the ith term, tij is the local weight of the ith term in the jth E-mail , dj is the normalization factor for the jth E-mail.

Local Term-Weighting: Logarithmic Term Frequency (TF) This weight depends on the frequency of feature within the Email. When the frequency of any word is high at that time logarithmic term frequency formula is used. This method is to de-emphasize the effect of high frequency. For example if ―Free word appears ten times in the E-mail than it doesn'tmean that this word is ten times more important in Spam Mails. If we don't do this then any time if ―Free word appears in normal mail then also it gets classified as Spam due to high local frequency. The formula for Logarithmic term frequency is
$t_{ij} = \log(f_{ij} + 1)$ (2) ; where fij is frequency of ith feature in jth Spam.
Global Term-Weighting: Inverse Document Frequency (IDF)

Stop-list: It contains words those doesn't having any potential as feature in any document like is, There, Subject, Those, That. Global term weighting is used to remove the necessity of stop-list. We have used IDF formula that is defined as logarithm of the ratio of total number of documents to the number of documents contains that word. Thus common words having low IDF. For example we have total 100 E-mail and out of those only 20 contain a particular feature than Global Weight for that particular feature is log(100/20)=0.698. If all 100 E-mail contain that feature than Global Weight for that feature is log(100/100)=0.

Normalization: Normally spam mail will not be lengthy and normalization is not used.

Storage: Storage module contains Filtration and Database sub-modules.

Filtration: After the completion of Weight assignment there are number of features that have total weight zero or not contains any potential value of total weight. This all features remove out in this phase. This phase is called filtration phase as it removes unnecessary features from the vector space.

Database: Feature those having potential value of total weight are stored in database and form vector space contains total weight, state either spam or non-spam E-mail for each feature.

Testing Module

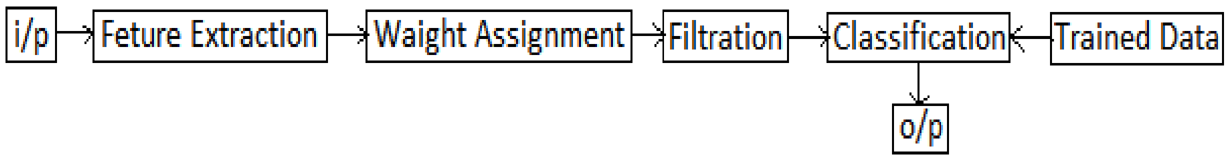


Figure 2 Testing Module

In Testing Module we should test any e-mail ; either Spam or Non-Spam. Based on the input we calculate the Vector Space Model same as in Training phase. Then we apply the classification algorithm on testing mail's Vector space Model on output of Training Module. and Classify the email as spam or non-spam.

Testing is a very important process in any design & development of the software. It uncovers all the bugs generated by the software to make the application a successful product.

A very important criterion for testing is the data set used, i.e., corpus. The corpus used for training and testing is the Ling Spam corpus. In LingSpam, there are four subdirectories, corresponding to 4 versions of the corpus, viz.,

bare: Lemmatiser disabled, stop-list disabled,

lemm: Lemmatiser enabled, stop-list disabled,

lemm_stop: Lemmatiser enabled, stop-list enabled,

stop: Lemmatiser disabled, stop-list enabled,

where lemmatizing is similar to stemming and stop-list tells if stopping is done on the content of the parts or not. Our analysis is done with this all subdirectory. Each one of these 4 directories contains so many parts and devided into the folder of 10,20,30, 40 and 50 message each. In every part, 2/3rd of the content is taken as training data and 1/3rd as the test data. Each one of this 4 subdirectories contains both spam and legitimate messages, one message in each file. There are hundreds of messages which are ready to training and testing.

But except this i also include my dataset also which contains spam messages and nonspam messages taking from our mail.

Vector Space Model

In the vector space model, we represent documents as vectors. The success or failure of the vector space method is based on term weighting [4]. Term weighting is an important aspect of modern text retrieval systems [5,4]. There are three components in a weighting scheme:
$a_{ij} = g_i * t_{ij} * d_j$ (1)

Where gi is the global weight of the ith term, tij is the local weight of the ith term in the jth document, dj is the normalization factor for the jth document. Email documents with their weighted terms shows in Table.

	Term 1	…	Term n
Email 1	W1		Wn
Email 2	:		:
:	:		:
Email n	W1		Wn

Table 1 Email Document represented with their Terms

K-NNC

K-nearest-neighbor (KNN) classification[2] is one of the most fundamental and simple classification methods The K-nearest neighbour classifier is a variant of the nearest neighbour classifier where instead of finding just one nearest neighbour as in the case of nearest neighbour classifier, k nearest neighbours are found. The nearest neighbours are found using the Euclidean distance, ManhattanDistance and ChebyshevDistance.K-nearest neighbor algorithm (KNN) is part of supervised learning that has been used in many applications in the field of data mining, statistical pattern recognition and many others.

Pseudo Code

```
(1) Add or View Classification:-

Input:- Add new classification in SpamClassificationSystem
Output:- Classification is successfully added.

Step1:- Addclassification(classifiction_name, class1_name, class2_name , table_name )
        {
                //-----new classification will be added into table 'classification'
                Insert into classification values ('classifiction_name' , 'class1_name' ,
                'class2_name' , 'table_name' )
        }

(2) Training:-

Input:- Give the spam or non-spam data
Output:- Vector Space Model created

Step 1:- Select the classification on which u want to apply the training.

Step 2:-Training(dataset(body), dataset(Subject), Class_name)
         {
        2.1     do for all dataset in body
                {
```

```
            //---- Extract all the the word and calculate its appropriate feature and
        store it into table which is specified in classification.
            2.1.1   Extract the feature one by one from mail which is in
            dataset(body)
            2.1.2   Calculate the local-weight of each word
            2.1.3   Calculate the global-weight
            2.1.4   Calculate the total-weight
            2.1.5   Calculate the no. of occurance
            2.1.6   Calculate the Probability
            2.1.7   Assign Flag to that as per Class_name
            2.1.8   Insert into classification_table values ('id' , 'feature' , 'local-
            weight', 'global-weight' , 'total-weight' , 'flag', 'occurance',
        'probability' )
        }
    2.2     do for all dataset in subject
        {
            //---- Extract all the the word and calculate its appropriate feature and
        store it into table which is specified in classification.
            2.2.1   Extract the feature one by one from mail which is in
            dataset(subject)
            2.2.2   Calculate the local-weight of each word
            2.2.3   Calculate the global-weight
            2.2.4   Calculate the total-weight
            2.2.5   Calculate the no. of occurance
            2.2.6   Calculate the Probability
            2.2.7   Assign Flag to that as per Class_name
            2.2.8   Insert into classification_table values ('id' , 'feature' , 'local-
            weight', 'global-weight' , 'total-weight' , 'flag', 'occurance',
        'probability' )
        }
        //----------Testing is done.
    }
```

(3) Testing:-

Input:- Give the e-mail which u want to test.
Output:- Tested e-mail is spam or non-spam.

Step 1:- Select the classification on which u want to apply the training.

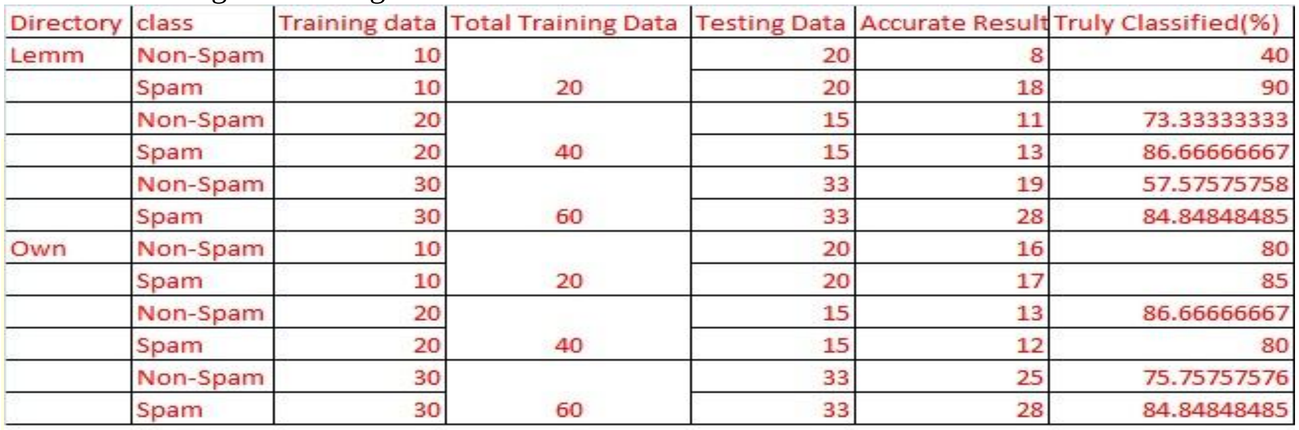
```
Step2:- Testing(new-email)
    {
    2.1     do for new-email
        {
            //---- Extract all the word and calculate its appropriate feature and store
        it into temporary table.
```

```
                    2.2.1   Extract the feature one by one from mail which is in
                    dataset(Subject & Body)
                    2.2.2   Calculate the local-weight of each word
                    2.2.3   Calculate the global-weight
                    2.2.4   Calculate the total-weight
                    2.2.5   Calculate the no. of occurance
                    2.2.6   Calculate the Probability
                    2.2.7   Assign Flag to that as per Class_name
                    2.2.8   Insert into temp_table values ( 'feature' , 'local-weight', 'global-
                    weight' , 'total-weight' , 'flag', 'occurance', 'probability' )
            }
            //---- fetch the data from the classification table and compare that feature with
        temp_table and calculate the distance between features in spam and non-
spam both and calculate the distance using classification method (K-NNC)
    2.2     K-NNC(Trainingdata, Newemaildata)
            {
            2.1     Calculate  the distance of newemail in class1(spam)
            2.2     Calculate  the distance of newemail in class2(nonspam)
            }
    2.3     if(distance(spam)>distance(nonspam)
            {
                    print(your message is spam)
            2.3.1   if(spam)
                    {
                            \\---- Train the newemail with flag spam
              2.3.1.1       Training(newemail)
                            {
                                    \\---- same as training code
                            }
                    }
            }
    2.4     else
            {
                    print(your message is nonspam)
            2.4.1   if(nonspam)
                    {
                            \\---- Train the newemail with flag nonspam
                    2.4.1.2 Training(newemail)
                            {
                                    \\---- same as training code
                            }
                    }
            }
    }
```

Results

In this module i have generated some results based on my dissertation work and make one table representation of that tested data. and also include the graph of accuracy,recall,specificity and precision.

Table 2. Training and Testing on dataset



Directory	class	Training data	Total Training Data	Testing Data	Accurate Result	Truly Classified(%)
Lemm	Non-Spam	10		20	8	40
	Spam	10	20	20	18	90
	Non-Spam	20		15	11	73.33333333
	Spam	20	40	15	13	86.66666667
	Non-Spam	30		33	19	57.57575758
	Spam	30	60	33	28	84.84848485
Own	Non-Spam	10		20	16	80
	Spam	10	20	20	17	85
	Non-Spam	20		15	13	86.66666667
	Spam	20	40	15	12	80
	Non-Spam	30		33	25	75.75757576
	Spam	30	60	33	28	84.84848485

Before we calculate the accuracy or precision, some basic term we need to remember that will show in this table. Evaluation formula is shown below.

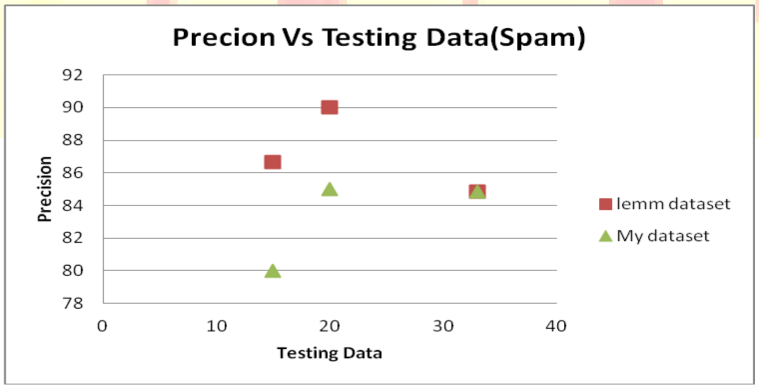
	Test Document	Classified
True Positive	Spam	Spam
True Negative	NonSpam	NonSpam
False Positive	Spam	NonSpam
False Negative	NonSpam	Spam

Precision

The percentage of positive predictions that are correct.

PRECISION(Spam) = TP/(TP+FP)

PRECISION(NonSpam) = TN/(TN+FN)



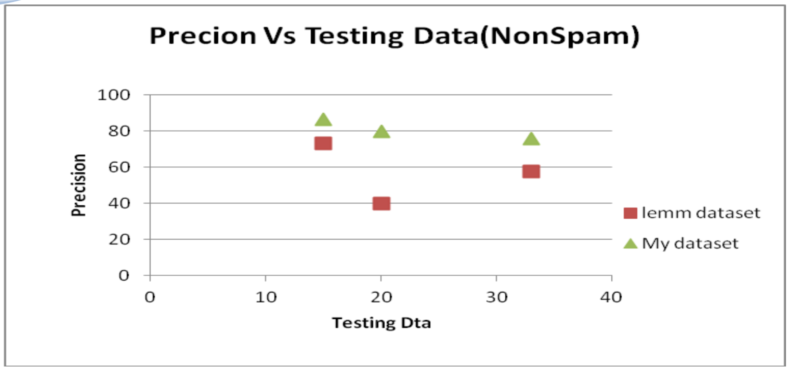


Fig 3. Precision Vs. Testing Data

Accuracy

The percentage of predictions that are correct.

ACCURACY = (Recall*(pos/(pos+neg))+(Specificity*(neg/neg+pos))

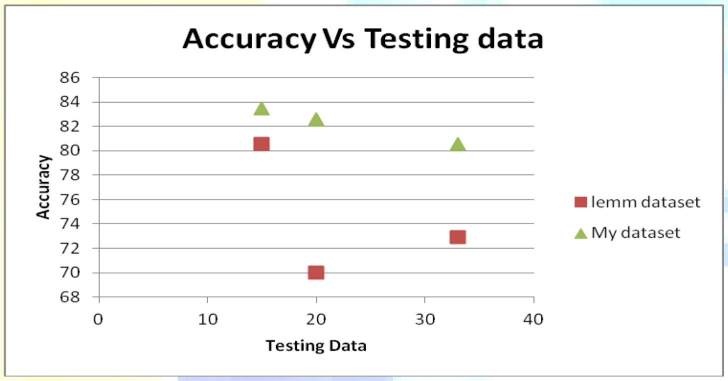


Fig 4. Accuracy Vs. Testing Data

Conclusion

We can conclude by reviewing the result and related work that no single technique can be provided alone to be the ideal solution with 0% fake positive and 0% fake unsatisfactory. In this adaptive approach, vector space model is one of the best model using classification method and it is efficient for filtering the spam mail using K-NNC classification algorithm for both smaller and larger data set. So we conclude that we may be evaluate the current state of research and propose own solution to the categorization with the 84% of accuracy and precision for finding the efficient spam from the E-mail using not only the content of that spam email but also with its subject line using two dataset which is very benificial to classify the spam. Further extended this work in a way of comparision of K-NNC classification algorithm with another classfication algorithm like Naive bayes and provide the different output which provided by them and finding accuracy and precision of them.

References

Journal articles

[1] M. Basavaraju, Dr. R. Prabhakar "A Novel Method of Spam Mail Detection using Text Based Clustering Approach" International Journal of Computer Applications (0975 – 8887) Volume 5– No.4, August 2010.

Book Chapter

[2].Jiawei Han and Micheline Kamber, "Data Mining Concepts and Techniques", Second Edition. In The Classification and Prediction, Lazy Learners (or learning from your neighbor) (pp 348-350). Morgan kaufmann Publishers.
(Jiawei Han and Micheline Kamber, 2006)

Conference Proceedings (Published)

[3] N.T.Mohammad , "A Fuzzy Clustering Approach to Filter Spam E-Mail" , Proceedings of the World Congress on Engineering 2011 Vol III WCE 2011, July 6 - 8, 2011, London, U.K.
(N.T.Mohammad,2011)

[4] Nicola Polettini, " The Vector Space Model in Information Retrieval- Term Weighting Problem" Nicola Polettini Department of Information and Communication Technology, Via Sommarive 14, 38050 Povo (TN) - Italy University of Trento, polettini@itc.it

[5] Chris Buckley. The importance of proper weighting methods. In M. Bates,editor, Human Language Technology. Morgan Kaufman, 1993.