SPAM CLASSIFICATION BASED ON SEMANTIC SIMILARITY AND CORPUS BASED THESAURUS

by

AJAY KANNAN 2014103618
ADITYA NATARAJAN 2014103619
SHRUTTE MURALI 2014103023

A project report submitted to the

FACULTY OF COMPUTER SCIENCE AND ENGINEERING

in partial fulfillment of the requirements for

the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

ANNA UNIVERSITY, CHENNAI – 25

BONAFIDE CERTIFICATE

Certified that this project report titled **SPAM CLASSIFICATION BASED ON SEMANTIC SIMILARITY AND CORPUS BASED THESAURUS** is the *bonafide* work of **AJAY KANNAN (2014103618)**, **ADITYA NATARAJAN (2014103619)** and **SHRUTTE MURALI (2014103023)** who carried out the project work under my supervision, for the fulfillment of the requirements for the award of the degree of Bachelor of Engineering in Computer Science and Engineering. Certified further that to the best of my knowledge, the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or an award was conferred on an earlier occasion on these or any other candidates.

Place: Chennai

Date:

Rajeswari Sridhar
Senior Assistant Professor
Department of Computer Science and Engineering
Anna University, Chennai – 25

ACKNOWLEDGEMENT

We would like to express our thanks to Dr. Rajeswari Sridhar for allowing us to proceed with this project and guiding us for the same. We also would like to thank the Department of Computer Science and Engineering for allowing us to complete the Creative and Innovative Project.

Ajay Kannan **Aditya Natarajan** **Shrutte Murali**

ABSTRACT

The primary objective of the project is to build a Spam Classifier based on Semantic Similarity. The project involves a modified Latent Semantic Analysis for constructing a Latent Semantic Feature Space. As an alternative, we also construct a Corpus based Thesaurus for maintaining contextual similarity. Using these feature spaces as templates, the documents are vectorized and pass through a Feed Forward Neural Network for accurate and efficient classification into the binary classes of Spam or Ham.

TABLE OF CONTENTS

ABSTRACT – ENGLISH iii

LIST OF FIGURES ix

LIST OF TABLES x

1 INTRODUCTION 1

1.1 PROBLEM DOMAIN 1

1.2 PROBLEM STATEMENT 2

1.3 SCOPE 2

1.4 CONTRIBUTION 3

1.5 OVERVIEW OF THE THESIS 3

2 RELATED WORK 4

2.1 NAIVE BASED APPROACH 4

2.2 K-NEAREST NEIGHBOURS 5

2.3 ENSEMBLE FILTERING WITH ACTIVE LEARNING 7

2.4 BOOSTING TREE APPROACH 7

2.5 OBSERVATIONS FROM THE SURVEY 8

3 REQUIREMENTS ANALYSIS 9

3.1 FUNCTIONAL REEQUIREMENTS 9

3.2 NON-FUNCTIONAL REQUIREMENTS 9

3.2.1 USER INTERFACE 9

3.2.2 HARDWARE CONSTRAINTS 9

3.2.3 SOFTWARE DEVELOPMENT 10

3.2.4 PERFORMANCE 10

3.3 CONSTRAINTS AND ASSUMPTIONS 10
3.3.1 CONSTRAINTS 10
3.3.2 ASSUMPTIONS 11
3.4 SYSTEM MODEL 11
3.4.1 USE CASE DIAGRAM 11
3.4.2 SEQUENCE DIAGRAM 11
4 SYSTEM DESIGN 13
4.1 SYSTEM DATA STRUCTURES 13
4.1.1 TEXT ARRAY 13
4.1.2 MODIFIED TEXT ARRAY 13
4.1.3 TERM BY DOCUMENT MATRIX 13
4.1.4 TRIPLET MATRIX ¡U,S,V¿ 14
4.1.5 REDUCE TERM BY DOCUMENT MATRIX . 14
4.1.6 FEATURE SET 14
4.1.7 TRAINING AND TESTING SET 14
4.2 MODULE DESIGN 15
4.2.1 TF*ID VECTORIZER TOOL 15
4.2.2 SINGLE VALUE DECOMPOSITION 16
4.2.3 TERM BY DOCUMENT MATRIX 16
4.2.4 CREATE DATASET 17
4.2.5 NEURAL NETWORK 17
4.3 CLASS DIAGRAM 17
4.4 COMPLEXITY ANALYSIS 17
4.4.1 TIME COMPLEXITY 17
4.4.2 COMPLEXITY OF THE PROJECT 17

5 SYSTEM DEVELOPMENT 19
5.1 SYSTEM PROTOTYPES 19
5.1.1 SEMANTIC FEATURE SPACE 19
5.1.2 Corpus Based Thesaurus Approach 20
5.1.3 NEURAL NETWORK DESIGN 20
5.1.4 SYSTEM DEPLOYMENT 21
6 RESULTS AND DISCUSSION 22
6.1 DATASET FOR TESTING 22
6.2 OUTPUT OBTAINED IN VARIOUS STAGES 22
6.2.1 INPUT 22
6.2.2 TOKENIZING AND STOP WORD REMOVAL 22
6.2.3 TF-IDF VECTORIZATION 22
6.2.4 SINGULAR VALUE DECOMPOSITION 23
6.3 SAMPLE SCREENSHOTS DURING TESTING 23
6.4 PERFORMANCE EVALUATION 25
6.4.1 PRECISION 25
6.4.2 RECALL 26
6.4.3 F-MEASURE 27
6.4.4 ACCURACY 28
6.5 SCORES OBTAINED AND DISCUSSION 28
6.6 DOMAIN WISE ANALYSIS 28
7 CONCLUSIONS 31
7.1 SUMMARY 31
7.2 CRITICISMS 31
7.3 FUTURE WORK 32

A Test Cases for Each Module 33
A.1 SNOWBALL STEMMER 33
A.1.1 Test Pre-requisites 33
A.2 Description 33
A.3 TEST CASES 33
A.3.1 TC ID: 01 33
A.3.2 TC ID: 02 34
A.4 TOKENIZER 34
A.4.1 Test Pre-requisite 34
A.4.2 Description 34
A.4.3 Test Cases 34
A.5 TF-IDF VECTORIZER 35
A.5.1 Test Pre-requisite 35
A.5.2 Description 35
A.5.3 Test Cases 35
A.6 SINGLE VALUE DECOMPOSITION 35
A.6.1 Test Pre-requisite 35
A.6.2 Description 35
A.6.3 Test Cases 35
A.7 MODIFIED DATASET CREATION 36
A.7.1 Test Pre-requisite 36
A.7.2 Description 36
A.7.3 Test Cases 36
A.8 NEURAL NETWORK FITTING AND CLASSIFICATION 36

A.8.1 Test Pre-requisite 36
A.8.2 Description 36
A.8.3 Test Cases 36

LIST OF FIGURES

2.1 An example showcasing the K-Nearest Neighbours and their polling 6
3.1 Use Case Diagram for the Spam Classification System . . . 12
3.2 Sequence Diagram for the Spam Classification System . . . 12
4.1 Vectorizer 15
4.2 Find K largest Rows 16
4.3 Class Diagram 18
6.1 Screenshot of an email 23
6.2 Screenshot of Term of Document Matrix 23
6.3 Screenshot of SVD 24
6.4 Screenshot of training 25
6.5 Domain Wise Analysis(a) 29
6.6 Domain Wise Analysis(b) 30

LIST OF TABLES

4.1 Time Complexity 18

6.1 Overall Classification of a test-batch 28

6.2 Personal Domain -Spam classification of personal mails . . . 29

6.3 Work Domain -Spam classification of work mails 29

6.4 Academia Domain -Spam classification of academia mails . 29

CHAPTER 1

INTRODUCTION

1.1 PROBLEM DOMAIN

Natural Language Processing is a field of computer science and linguistics concerned with the interactions between computers and human (natural) languages. It is a process through which meaning can be extracted in the context of the sentence written. Statistical Natural Language Processing (NLP) is a field of Natural Language Processing which uses stochastic, probabilistic and statistical methods for problem solving.It is used for processing the contents of an email using various algorithms and also eliminating the tedious job of processing unnecessary parts of the email body.

Machine learning is closely related to computational statistics, which also focuses in prediction-making from training a set of data.Many users receive numerous unwanted emails each day which is why a spam filter has been created. The spam filter needs to sort incoming mail into wanted and unwanted. This can be tricky because the filter could allow too much spam into the inbox or could label some legitimate emails as spam. Machine learning can help solve this problem. The email client can be trained to learn where to put each email.A machine learning algorithm for email filtering will take in a set of labeled messages as the input and will output correct labels for the testing data.

The various machine learning approaches used in spam classification are Nave Bayesian classifier that is based on Bayes theorem and as-

sumes that all features are statistically independent,the SVM classifier has a good generalization performance and its ability to handle high dimensional data by using its kernels are very effective in a wide range of bioinformatics problems.In the KNN approach emails are classified based on the class of their nearest neighbors.

Semantic similarity approach uses vector space model to represent each document which is implemented by creating the term-document matrix and a vector of email document.The latent semantic analysis uses the singular value decomposition (SVD)technique to decompose a large term-document matrix into a set of k orthogonal factors.

1.2 PROBLEM STATEMENT

Given the flooding of large unsolicited and unwarranted Spam mails circulating in the web which cause both those naive to fall into the malicious traps encapsulated in the mails and the cluttering of fake advertisements in the email-box, there is a need to create a Spam-classification system which can efficiently and intelligently classify spams thus saving the trouble of millions.

The paper proceeds to elaborate or instantiate a system that performs spam classification using both semantic similarity and the less used corpus based thesaurus approach which is implemented using a LING SPAM corpus. The generated feature space along with the input messages are used to form the input for a feed-forward neural network which classifies the input as either a Spam or a Ham.

1.3 SCOPE

According to Commtouchs report in first quarter of 2010, there are average 183 billion spam messages sent everyday. Such mails not only waste a lot of bandwidth, but can also cause serious damages to per-

sonal computers in the form of viruses. Spammers are getting smarter, continuously trying to come up with approaches that enable them to circumnavigate spam-filtering schemes. Furthermore, active research in emerging approaches shows that there is a continued effort to come up with better and alternative anti-spam schemes. Given that the operational landscape of spam ecosystems is continuously changing, different techniques that can be applied to old and new problem areas are being sought. Social networks, for example, have become a major breeding ground for spam-related activities. Research in these specific areas, including the behavioral model space has also intensifies with continual advancement.

1.4 CONTRIBUTION

This system uses two approaches in the semantic similarity , one of which involves the construction of a Corpus Based Thesaurus. One of the main advantages of this approach is that it considers the contextual similarity between terms. The system can classify email as Spam or Ham with a reasonably high degree of accuracy. However, this approach is high dependent on memory, and requires heavy computational power. This system can further be refined with a more powerful system

1.5 OVERVIEW OF THE THESIS

Chapter 2 discusses the existing approaches of spam filtering in greater detail. It also analyses the advantages and disadvantages of each approach. Chapter 3 gives the requirements analysis of the system. It explains the functional and non-functional requirements, constraints and assumptions made in the implementation of the system and the various UML diagrams.

CHAPTER 2

RELATED WORK

This chapter gives a survey of the possible approaches to Spam Filtering. We proposed to use a semantic similarity approach, which involved the construction of a latent semantic feature space and a corpus based thesaurus. The output of each email must contain a classification- either Spam or Ham. This survey helped us analyse the various existing approaches to Spam Filtering and decide on the one which would best cater to our needs.

2.1 NAIVE BASED APPROACH

This approach involves the application of the famous Bayes Probability Theorem in the domain of Spam Filtering [**?**]. Bayes Theorem can be stated as follows:

$P(H|E) = (P(H) * P(E|H))/P(E)$

In the above equation,

$P(H|E)$: Posterior probability of H , given evidence E

$P(H)$: Prior probability

$P(E|H)$: Likelihood of evidence E if hypothesis H is true

$P(E)$: Priori probability that the evidence itself is true

The above formula is used to calculate the probability that an item belongs to a certain class. Suppose that we knew exactly, that the word DISCOUNT could never occur in a legitimate message. Then when we saw a message containing this word, we could tell for sure that it was spam. This simple idea can be generalized using some probability

theory. We consider two categories classes: Spam(S) and Ham (H). For a given message x,

If $(S|x) > P(H|x)$, (that is, if the a-posteriori probability that x is Spam is greater than the a-posteriori probability that x is non-spam), classify x as Spam, otherwise classify it as Ham.

This can be transformed to the form: If $((P(x|S)/P(x|H)) > (P(H)/P(S)))$, classify x as Spam, otherwise classify x as Ham. Consider a message m containing a feature vector x. Let us try the simplest case of a feature vector with a single binary attribute that denotes the presence of a certain word w in the message. The messages feature vector x is set to 1 if the word is present in the message, 0 otherwise.

The decision rules are then used to classify a given message as Spam or Ham. This approach however assumes predictors are independent 2. In real life, it is almost impossible that we get a set of predictors which are completely independent.

2.2 K-NEAREST NEIGHBOURS

The k-nearest neighbour (K-NN) classifier is considered an example-based classifier, which means that the training documents are used for comparison rather than an explicit category representation, as in the case of the Nave Bayes approach. This approach is generally considered as amongst the simpler machine learning algorithms. When a new object needs to be categorized, the k most similar objects (neighbours) are found and if a large enough proportion of them have been assigned to a certain category, the new object is also assigned to this category, otherwise not.3 If k = 1, then the object is simply assigned to the class of that single nearest neighbour.

The test sample (green circle) should be classified either to the first class of blue squares or to the second class of red triangles. If k = 3

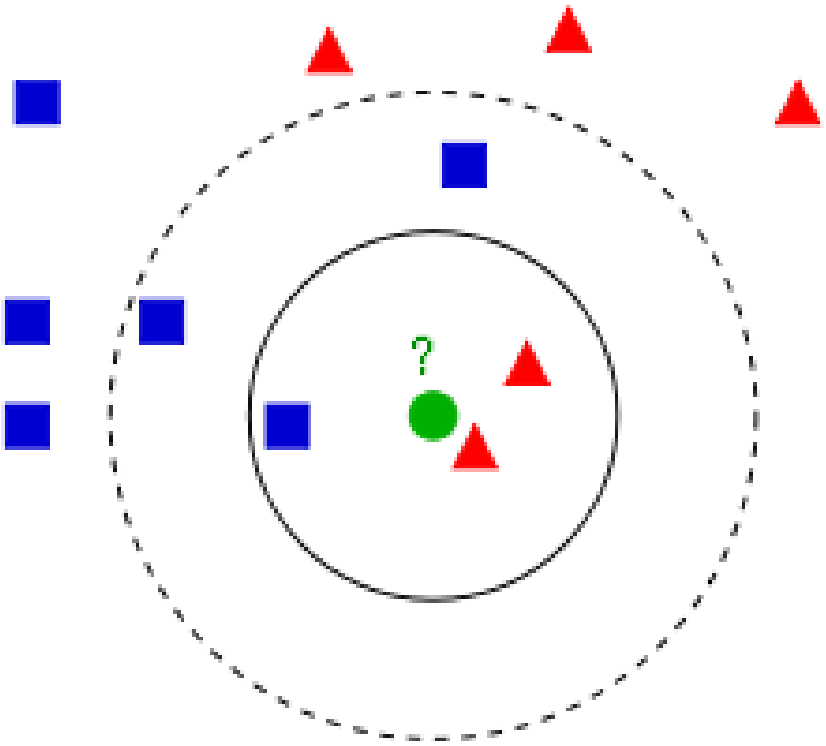

Figure 2.1 An example showcasing the K-Nearest Neighbours and their polling

(solid line circle) it is assigned to the second class because there are 2 triangles and only 1 square inside the inner circle. If k = 5 (dashed line circle) it is assigned to the first class (3 squares vs. 2 triangles inside the outer circle). In the context of Spam Filtering, given a message x, we need to first determine its k nearest neighbours among the messages in the training set. If there are more spam's among these neighbours, the message is classified as Spam. Otherwise, it is Ham. One of the drawbacks of the presented algorithm is that there seems to be no parameter that we could tune to reduce the number of false positives. This problem is easily solved by changing the classification rule to the following l/k-rule:

If l or more messages among the k nearest neighbours of x are spam, classify x as spam, otherwise classify it as legitimate mail.

2.3 ENSEMBLE FILTERING WITH ACTIVE LEARNING

Ensemble learning methods typically use a large number of classifiers. When a new sample is to be classified, a combination of results produced by each single classifier governs the overall classification of the sample. In active learning, a small number of labelled training samples are used to build an initial classifier. The basic idea of selecting samples takes into account the roles of different samples in classification, namely, the greater the amount of information contained in samples, the more important they are in the determination of the classification surface. That is to say, for a large number of unlabelled samples, only a few of them need to be labelled and used to train the model with strong classification performance. Pool-based active learning methods have achieved good results in spam filtering with only a few e-mails tagging. A Pool-based active learning method assumes that the learner can access to n unlabelled samples and at most m (m ¡¡ n) of them can be requested when it needs user tagging 4. Researchers have made a wide range of approaches to obtain m samples, among which there are two typical methods: uncertainty sampling based method and query by committee based method 5. The former one is to select samples, in which the filter is unable to clearly distinguish to join the training set and then use these samples to train classifier. The latter one is to build two or more classifiers based on history labelled samples as committee, then make use of the committee to vote on incoming samples and the samples with most inconsistent votes are selected as candidates for training.

2.4 BOOSTING TREE APPROACH

Boosting is an ensemble technique that attempts to create a strong classifier from a number of weak classifiers. This is done by building a

model from the training data, then creating a second model that attempts to correct the errors from the first model. Models are added until the training set is predicted perfectly or a maximum number of models are added.6

2.5 OBSERVATIONS FROM THE SURVEY

We propose to incorporate a semantic- similarity based approach to spam filtering. Therefore, it no single approach mentioned above will fit the given bill. Other traditional and common spam filtering methods, such as Backpropagation Neural Networks (BPNN) and Vector Space Modelling (VSM) also have their own drawbacks. For example, neural networks have the problems of slow training speed and easy to trap into local minima. Vector space model has three challenges: high dimensionality, sparse representation and semantic concern. For our approach, we have chosen to implement a Latent Sematic Feature Space (LSFS) and a Corpus Based Thesaurus (CBT). We use the standard implementations of Multi-Layer Perceptrons (MLPs) and Support Vector Machines (SVMs) for our classification purposes.

CHAPTER 3

REQUIREMENTS ANALYSIS

3.1 FUNCTIONAL REEQUIREMENTS

The functional requirements of the system are quite synonymous with the objective of the paper. The system must efficiently and accurately classify any given e-mail (Digital communication message) as either Spam or Ham. The classification must be based on both the message and the body of the Message thus giving no leeway for erroneous classifications (Spam being classified as Ham and vice-versa).

3.2 NON-FUNCTIONAL REQUIREMENTS

3.2.1 USER INTERFACE

The user interface is kept simplistic that allows a user/agent to load a message body as input. The result of classification(spam/ham) is displayed. While testing, the UI displays the efficiency of classification on the screen, when a test data set is loaded.

3.2.2 HARDWARE CONSTRAINTS

Since, the system uses Machine Learning classifiers and Natural Language processing tool, it requires a certain level of processing power. A machine with 8Gb RAM and an intel 7th Gen series processor is used to train the system and test the same (considering the LING SPAM data set used). Alternative approach could be to use a GPU with enough memory based on the data sets.

3.2.3 SOFTWARE DEVELOPMENT

OPERATING SYSTEM

The System is independent to the OS used, yet preferably Linux.

PROGRAMMING LANGUAGE

The system is programmed using Python in an Anaconda package based environment.

TOOLS

The system mainly makes use of 2 tools. The NLTK library (Natural Language Toolkit) is used for processing the natural language text into vectors. Which is the subjected to a neural network as input which is designed and built using Tensor Flow (Keras toolkit for design and architecture of the neural network). The neural network can also be created using a scikit-learn package toolkit. The subordinate tools used as Numpy, Scipy and other basic python libraries.

3.2.4 PERFORMANCE

The expected performance of the system must be synonymous with efficient instantaneous classification of the message as either a spam or ham with almost zero latency and 100% accuracy. The system architecture designed is trained and tested with the above mentioned data-sets.

3.3 CONSTRAINTS AND ASSUMPTIONS

3.3.1 CONSTRAINTS

The system design is posed with quite a few constraints. The fundamental of it being, the limited amount of data-set for training and testing which may not give the accurate efficiency of the system. Also,

the limited computing power does not allow the design to be expansive in nature, which may prove to be both ineffective and time consuming even for small data-sets. Yet, the hardware constraints can be overcome by implementing an effective and an optimal design to replicate the deployment if done in large scale.

3.3.2 ASSUMPTIONS

The system, though designed in a generic manner it tends to become data set oriented (overfitting of the dataset). Hence, it is assumed that the data set chosen is generic in nature and that it does not follow a traceable pattern. The corpus based thesaurus approach is also based out a fundamental approach that the test emails can be classified with feature space generated from the training data alone.

Some of the technical assumptions is that term-document feature space is representative of the original labeling of the message. This fundamental assumption is further validated in the testing phase which is elaborated in the further sections.

3.4 SYSTEM MODEL

3.4.1 USE CASE DIAGRAM

The System is designed as a uni-goal system with the core motif to classify spam messages from the ham emais. The server running the core algorithm is responsible for recieving an email, it's classificaiton and providing a UI to the user.

3.4.2 SEQUENCE DIAGRAM

The sequence of steps involved in spam classification is shown in the figure. The sequence diagram shown showcases the various func-

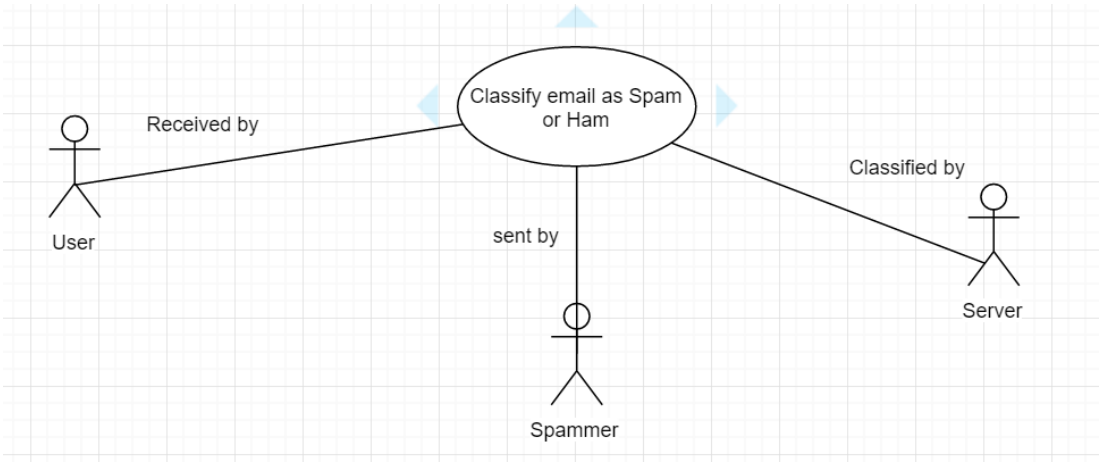

Figure 3.1 Use Case Diagram for the Spam Classification System

tional classes in the system which perform various processes to process the data and develop the feature space/feature set. The features are then classified using a feed forward neural network.

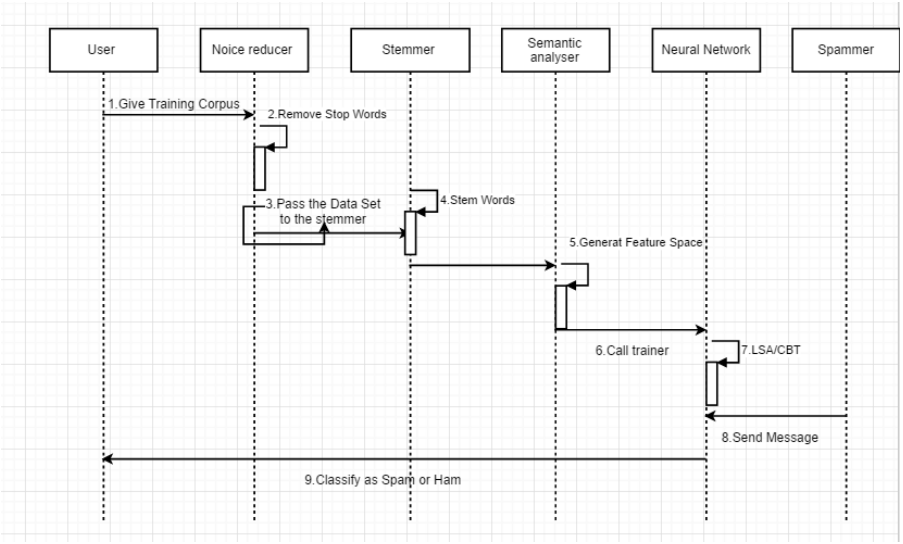

Figure 3.2 Sequence Diagram for the Spam Classification System

CHAPTER 4

SYSTEM DESIGN

4.1 SYSTEM DATA STRUCTURES

The system makes use of certain data structures for efficient performance and understandable abstraction. The email texts are periodically stored, processed and transformed in this architecture, as follows.

4.1.1 TEXT ARRAY

The Spam and Ham email bodies are loaded into vectors of data type char, namely $txt-array$, which is used for initialising the input. These $txt-arrays$ are sent as input into the preprocessing software.

4.1.2 MODIFIED TEXT ARRAY

The content of this array is the result of modification of $txt-array$. The content of each $txt-array$ is converted to lowercase letters while also removing the punctuation marks. This new text, is loaded into the $modified-text-array$.

4.1.3 TERM BY DOCUMENT MATRIX

All $modified-text-arrays$ are passed through a tfidf vectorizer and the resultant vectors are stored in the form of a matrix in a Term By Document Matrix. The Matrix contains vectorized form of the text which shall be used for further processing, as processing numbers are easier than text.

4.1.4 TRIPLET MATRIX ¡U,S,V¿

This composition is generated by passing the Term by Document Matrix through a Single-Value-Decomposition factoring method. This generates three matrices, namely, U, S and V matrices, with matrix S being a diagonal matrix. These constructs are used for creating the feature set.

4.1.5 REDUCE TERM BY DOCUMENT MATRIX

Latent Semantic Feature Space is a method which is slightly different from the Latent Semantic Analysis, where each document is multiplied with the transpose of the U matrix (rather than multiplying S also as done in LSA) to generate the Feature Set. Since, the size of the matrix U is very large, corresponding feature reduction is ensued. Top k features are selected in the diagonal matrix, S, since it corresponds to both the term and documents of the Data Space. The corresponding rows in the U matrix, to these top k highest elements from the S matrix are used. This new matrix form the reduced term by document matrix and is used for generating the feature set.

4.1.6 FEATURE SET

Each document is done transformed with the TF*IDF vectorizer and is multiplied with transpose of reduced term by document matrix. The resultant vectors are stored in a data construct called the feature set, which shall be used for training and testing the Neural Network.

4.1.7 TRAINING AND TESTING SET

The Feature is split in a 90-10 fashion for training and testing the neural network. The split is done in by randomly choosing 10% of the data set (Ling-Spam Corpus provides the data set in a 10 part fashion

which is pre-randomised).

4.2 MODULE DESIGN

4.2.1 TF*ID VECTORIZER TOOL

Using NLTK utilities, this tool is initialised to pre-process the natural language text, by removing stop words, tokenizing, stemming and vectorizing the text, in the same order as mentioned. It converts the tokens into vectors using TF*IDF algorithm.

TEXT STEMMER

For stemming, a Snowball stemmer is made use of. The initialization algorithm for this module is show below.

```
from nltk.stem.snowball import SnowballStemmer
def stem_tokens(tokens,stemmer):
    stemmed = []
    for item in tokens:
        #if not item == 'oed':
        stemmed.append(stemmer.stem(item))
    return stemmed

def tokenize(text):
    tokens = nltk.word_tokenize(text)
    stems = stem_tokens(tokens,stemmer)
    return stems
```

Figure 4.1 Vectorizer

TOKENIZER

The tokenizer function is used to tokenize the text and to call the stemmer function. The pseudocode for the same is shown below.

4.2.2 SINGLE VALUE DECOMPOSITION

In linear algebra, the singular value decomposition (SVD) is a factorization of a real or complex matrix. It is the generalization of the eigendecomposition of a positive semidefinite normal matrix (for example, a symmetric matrix with positive eigenvalues) to any matrix via an extension of polar decomposition. This algorithm is used to reduce the dimensionality of the Term by document matrix.

$$< U, S, V >= np.linalg.svd(D, full_matrices = 0, compute_uv = 1)$$

4.2.3 TERM BY DOCUMENT MATRIX

The resultant of the Single value decomposition poses to be a very large matrix which is tough to process later on. As explained before the U matrix is reduced by finding the K largest values in S matrix. The algorithm for the same is given below.

```
def findklargest(s,k):
    d=[]
    j=0
    for i in s:
        d.append([j,i])
        j=j+1
    d=sorted(d,key=lambda x:x[1])
    d.reverse()
    #print d
    s=[]
    for i,j in d:
        if k<0:
            break
        s.append(i)
        k=k-1
    return s
```

Figure 4.2 Find K largest Rows

4.2.4 CREATE DATASET

Once the feature space is defined each document is processed using the feature space matrix (U) to create the data set for testing and training the neural network. The algorithm for the same is given below.

Creating Data Set

1: $s = u^T * tfidf.transform(s)$

2: **return** s

4.2.5 NEURAL NETWORK

A simple feed forward neural network is used to classify the documents as spam and ham. The algorithm is designed by having two layers before the output layer, with the activation functions being tanh and softmax respectively. The optimizer function for the same is Stochastic Gradient Descent. The number of nodes in each layer is defined by the formula (Nprev-Nnext)/2.

4.3 CLASS DIAGRAM

The class diagram of the spam classification system is shown in the figure. This diagram depicts the functions of various modules in the system clearly. It also shows the interaction between the modules of the system thereby providing a clear idea for implementation.

4.4 COMPLEXITY ANALYSIS

4.4.1 TIME COMPLEXITY

The Time complexity of each module is shown in the table 4.1.

4.4.2 COMPLEXITY OF THE PROJECT

1) The complexity of the system lies in dealing with words that are not present in the training set.

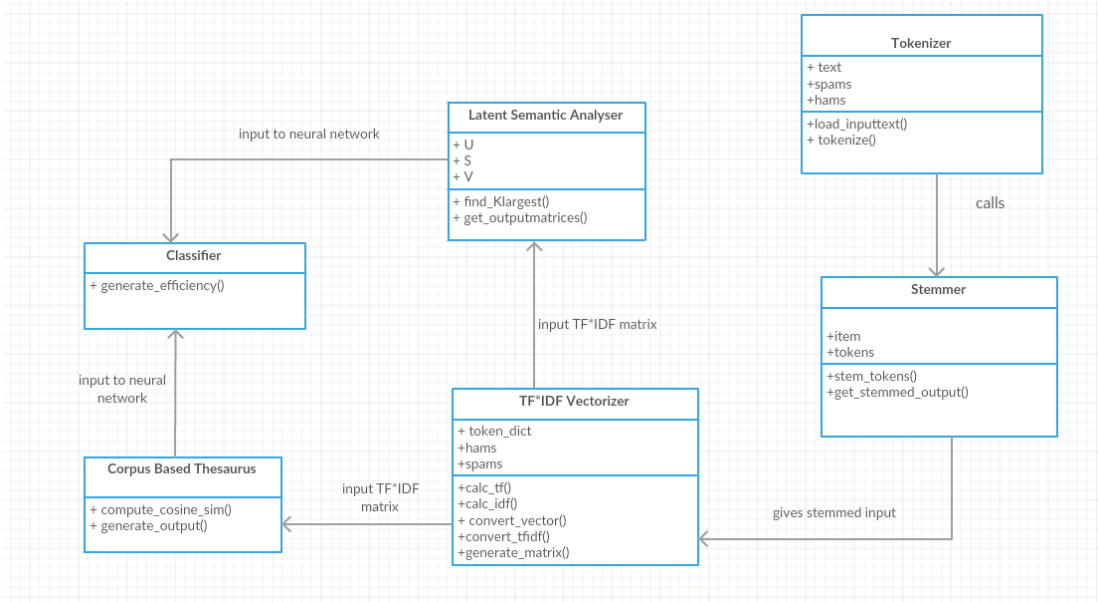

Figure 4.3 Class Diagram

Table 4.1 Time Complexity

Sno	**Module**	**Complexity**
1	TF*IDF Vectorizer	O(n^2)
2	Single Value Decomposition	O(n^3)
3	Reduction of Term by Document Matrix	O(n^2)
4	Creating the Data Set	O(n)
5	Neural Network - Training and Testing	O(n^3)

2) The term by document matrix is computed for each word in the dataset.

3) Words or sentences that are trained as spams but appear in ham emails reduce the overall efficiency of the system.

4) In Corpus based Thesaurus,instead of checking co-occurence between just 2 words,we could determine a feature corresponding to relationship between n words.

CHAPTER 5

SYSTEM DEVELOPMENT

The system described consists of various packages like NLTK,SnowBall Stemmer,and neural network.The overall code overview showing the organisation of these various packages of the spam classification system can be seen in the figure below.

5.1 SYSTEM PROTOTYPES

The input and output to each module of the system is described in this section.

5.1.1 SEMANTIC FEATURE SPACE

This module takes as input,the spam and ham email bodies.It tokenizes the contents of spam and ham arrays,converts them to lowercase and generates corresponding output arrays as simplified email corpus. The snowball stemmer takes the simplified email corpus as input and removes the suffix of different occurrences of the verb and produces a primitive email corpus as the output.

TF*IDF-VECTORIZER

The TF*IDF Vectoriser takes the primitive email corpus as input and assigns weights to each word by computing Term frequency matrix and the inverse document frequency matrix.The output is a TF*IDF matrix.

VALUE DECOMPOSITION

The SVD takes as input,the TF*IDF matrix and splits it into a triplet matrices ¡U,S,V¿, which is then passed through the reduction algorithm. The resultant matrix is kept as the template matrix.

5.1.2 Corpus Based Thesaurus Approach

The basic idea of corpus based thesaurus is to calculate the similarities between two terms on the basis of their co-occurrence in a document corpus. This approach is based on the association hypothesis that related terms tend to co-occur in documents in the corpus. As a result, this type of automatic thesaurus consists of a set of weighted term associations.

Algorithmically, the only difference between the LSFS approach and the corpus based thesaurus approach is that the Term by document matrix before passing through the SVD is passed through a cosine similarity algorithm to find the co-occurrence between any two given words in the corpus. The resultant is then further reduced through decomposition.

5.1.3 NEURAL NETWORK DESIGN

In order to improve the back propagation algorithm in terms of its rate of convergence and global search capability, we consider the adjustment of the learning rate, since the learning behavior of the back propagation depends very much on the choice of learning rate. The adaptive back propagation algorithm uses statistical technique to evaluate each learning phase. The learning effects indicate the direction of the network globally and get rid of the blindness of mechanical and repeated single learning. The next learning phase adjusts the learning model based on the evaluation effects in the previous learning phase. During the learning phase, the system records: the Minimum Error in

the Current learning phase (CME), the Minimum Error in the Previous learning phase (PME), and the Minimum Error that we get so far from the beginning of the training process, which we called Global Minimum Error (GME). Then, we calculate the learning effects of the networks: the global effect, relative effect and synthesis effect. The Global Effect (GEffect) is used to evaluate the effect of the optimum value between the current learning phase and global learning phase; the Relative Effect (REffect) is used to evaluate the effect between the current learning phase and previous learning phase; the Synthesis Effect (SEffect) is used to evaluate the effect in the whole learning process.

5.1.4 SYSTEM DEPLOYMENT

Once, the system is trained, any given email can be taken and transformed using the Tfidf vectorizer. The transformed vector is multiplied with the template reduced term by document matrix to create the feature set. When the feature set is passed through the neural net, the networks prediction is the classification.

CHAPTER 6

RESULTS AND DISCUSSION

6.1 DATASET FOR TESTING

The input to the system is the Ling Spam PU1 Corpus. The dataset is divided into 10 subdirectories (part1, part2part10), each of which contains 241 ham messages and 48 spam messages. The subdirectories correspond to the 10 fold validation experiments, where 9 parts are used for training and 1 part is used for testing. The results of this module testing as well as the testing of the entire system are summarized below.

6.2 OUTPUT OBTAINED IN VARIOUS STAGES

This section shows the results obtained during module testing.

6.2.1 INPUT

Part 10 of the dataset is considered as the input to the system. It contains 241 ham messages and 48 spam messages. Figure 6.1 depicts the input.

6.2.2 TOKENIZING AND STOP WORD REMOVAL

The output of Tokenization and Stop Word removal is shown in figure 6.2.

6.2.3 TF-IDF VECTORIZATION

Figure 6.3 shows the output of the TF-IDF vectorizer.

6.2.4 SINGULAR VALUE DECOMPOSITION

The output of the SVD stage is shown in figure 6.4. The matrices U, S and V are also depicted.

6.3 SAMPLE SCREENSHOTS DURING TESTING

The screen shot is shown in Figure 6.1 for better reference.

```
Subject: special issues

names , the journal of the american name society , is planning two special issues for
late 1995 or early 1996 . the first is on ' computers in onomastic research , ' and
the second ' statistics in onomastic research . ' if you would like to contribute to
one ( or both ) of these special issues , send a 1 - page idea paper to the editor at
the address below . nothing has to be definite at this time , but i would expect that
both issues would deal with problems faced in name research and how computers ( and
statistics ) could contribute to solving them . i put a similar notice on the
american name society list and found that there was wide-spread interest in both
areas . so if you would like to contribute , send the idea paper to me shortly , by
either hard copy , fax or email . edward callary , editor , editor , names english
department northern illinois university dekalb , il 60116 fax : 815-753 - 0606
email : tb0exc1 @ mvs . cso . niu . edu ( make sure you type zero rather than o after
tb ) i hope to hear from people from a variety of disciplines who have an interest in
names . please let me know if you have questions or comments .
```

Figure 6.1 Screenshot of an email

An example of the term by document matrix is show in Figure 6.2.

```
In [21]: tfs=tfidf.fit_transform(token_dict.values())
         #print(tfs)
         a=tfs.toarray()
         print "\n Term by document matrix generated.."
         print a
         print "\n Densed.."

 Term by document matrix generated..
[[ 0.          0.          0.         ...,  0.          0.          0.        ]
 [ 0.04195251  0.          0.02100127 ...,  0.          0.          0.        ]
 [ 0.          0.          0.         ...,  0.          0.          0.        ]
 ...,
 [ 0.          0.          0.         ...,  0.          0.          0.        ]
 [ 0.          0.          0.         ...,  0.          0.          0.        ]
 [ 0.          0.          0.         ...,  0.          0.          0.        ]]

 Densed..
```

Figure 6.2 Screenshot of Term of Document Matrix

An example of the single value decomposition is shown in Figure 6.3.

```
#a="This sentence has unseen text such as king"
#print tfs.values()
#print tfidf.transform([a])
#convert(tfidf)
u=[]
s=[]
v=[]
print "\n SVD..."
ula,s,v=np.linalg.svd(a, full_matrices=0, compute_uv=1)
print "Generated matrices are.. U:"
print u
print "\n S:"
print s
print "\n V:"
print v
```

```
 SVD...
Generated matrices are.. U:
[]

 S:
[  9.52595326e+00   5.80726876e+00   4.70749867e+00 ...,   7.57578359e-17
   4.14456762e-17   4.34213803e-31]

 V:
[[ -3.24299298e-02  -1.04443998e-01  -6.83009412e-02 ...,  -2.98494617e-04
   -8.00991848e-05  -3.37381221e-04]
 [  8.89651539e-03  -5.81750891e-02  -1.97438054e-01 ...,   2.78467348e-04
   -2.65147617e-04   3.83577469e-04]
 [ -1.49979638e-02   7.76755554e-02   7.09524159e-02 ...,   3.37940066e-04
    3.09236581e-05   4.35972907e-04]
 ...,
 [  1.36646891e-03   1.84931123e-03   3.16376937e-03 ...,  -9.39700268e-04
    2.97804400e-05   4.24646002e-03]
 [ -1.21008630e-02  -1.91924343e-04   4.27256423e-04 ...,   7.38331791e-05
    2.20205250e-04  -3.06068693e-03]
 [  7.17228024e-03  -3.72617361e-04   1.02896548e-03 ...,  -2.43763793e-04
    3.81187007e-04   2.79959555e-03]]
```

Figure 6.3 Screenshot of SVD

The system is tested for various test cases which are detailed in Appendix A module-wise.

wecol ’ 98 - - western conference on linguistics arizona state university cordially invites you to attend wecol ’ 98 (western conference on linguistics) , to be held in conjunction with lasso xxvii , 9-11 october 1998 . information about the conference (s) may be found at our websites : http : / / www . public . asu . edu / teresalw / wecol . htm http : / / www . public . asu . edu / teresalw / lasso . htm these sites include preliminary programs as well as information about the meeting site , lodging , nearby areas of interest , and your

host institution arizona state university . please feel free to contact the pagemaster of the site , teresa . wells @ asu . edu , with any questions . teresa wells research assistant to dr . elly van gelderen english department arizona state university

The training of the neural network using the corpus is shown in Figure 6.4.

```
In [10]: ''''from sklearn.neural_network import MLPClassifier
clf = MLPClassifier(solver='lbfgs', alpha=1e-5,hidden_layer_sizes=(5, 2), random_state=1,verbose=True)
clf.fit(anew, l1)

'''
from keras.models import Sequential
from keras.layers import Dense, Activation
from keras.utils.np_utils import to_categorical
model=Sequential()
from  keras.layers.core import Dropout
model.add(Dense(100,input_dim=2601))
#model.add(Dropout(0.5))
model.add(Activation("tanh"))
#model.add(Dense(100,input_dim=1000))
#model.add(Activation("tanh"))
model.add(Dense(2))
model.add(Activation('softmax'))

model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['binary_accuracy'])
#label=to_categorical(label3)
model.fit(anew, l1, nb_epoch=5, batch_size=100)

Using TensorFlow backend.

Epoch 1/5
2601/2601 [==============================] - 5s - loss: 0.5751 - binary_accuracy: 0.8158
Epoch 2/5
2601/2601 [==============================] - 0s - loss: 0.4729 - binary_accuracy: 0.8339
Epoch 3/5
2601/2601 [==============================] - 0s - loss: 0.4503 - binary_accuracy: 0.8339
Epoch 4/5
2601/2601 [==============================] - 0s - loss: 0.4449 - binary_accuracy: 0.8347
Epoch 5/5
2601/2601 [==============================] - 0s - loss: 0.4405 - binary_accuracy: 0.8347

Out[10]: <keras.callbacks.History at 0x7f6acb58e550>
```

Figure 6.4 Screenshot of training

6.4 PERFORMANCE EVALUATION

The performance of the entire system is evaluated using the standard parameters described below.

6.4.1 PRECISION

DESCRIPTION

Precision refers to the fraction of retrieved instances that are relevant. It is a measure of exactness i.e what % of tuples that the classifier labeled as positive is actually positive. It is calculated as the number

of true positives (Tp) over the number of true positives plus the number of false positives (Fp). Precision always ranges from 0 to 1, with higher values indicating better classification. Precision is calculated by equation 6.1 as:

$Precision = TP/(TP+FP)$

TP is the number of actual ham messages classified correctly

FP is the number of actual spam messages misclassified as ham

ALGORITHM

The algorithm to calculate precision is as follows:

Find the total number of true positives i.e actual ham messages being correctly classified as ham (TP).

Find the number of false positives i.e actual spam messages being classified incorrectly as ham (FP).

Substitute TP and FP values in equation 6.1 to obtain precision.

6.4.2 RECALL

DESCRIPTION

Recall refers to the fraction of relevant instances that are retrieved. It is a measure of completeness i.e what % of positive tuples did the classifier label as positive. It is calculated as the number of true positives (Tp) over the number of true positives plus the number of false negatives (Fn). Recall also ranges from 0 to 1, with 1.0 being considered as the perfect score. Recall is calculated using equation as:

$Recall = TP/(TP+FP)$

TP is the number of ham messages classified correctly

FN is the number of ham messages misclassified as spam

ALGORITHM

The algorithm to calculate recall is as follows

Find the total number of true positives i.e actual ham messages being correctly classified as ham (TP).

Find the number of false negatives i.e actual ham messages being classified incorrectly as spam (FN).

Substitute TP and FP values in equation 6.2 to obtain precision.

6.4.3 F-MEASURE

DESCRIPTION

F measure (or F1 Score) is the harmonic mean of precision and recall. It varies from 0 to 1, with higher values considered better. F measure is calculated by equation 6.3 as:

$F - Measure = 2TP/(2TP + FP + FN)$

TP is the number of ham messages classified correctly

FP is the number of spam messages incorrectly classified as ham

FN is the number of ham messages misclassified as spam

ALGORITHM

The algorithm to calculate the F measure is as follows:

Find the total number of true positives i.e actual ham messages being correctly classified as ham (TP).

Find the number of false positives i.e actual spam messages being classified incorrectly as ham (FP).

Find the number of false negatives i.e actual ham messages being classified incorrectly as spam (FN).

Substitute TP, FP and FN values in equation 6.3 to obtain precision.

6.4.4 ACCURACY

Accuracy is defined as the ratio of the True Positives and True Negatives to the entire classified dataset.

$Accuracy = (TP + TP)/(TP + FP + TN + FN)$

The major disadvantage of this metric is that for a biased dataset, even a trivial classifier one that classifies all test data as only spam/ham is likely to give an appreciable value. Therefore, F measure is generally considered as a better evaluation metric.

6.5 SCORES OBTAINED AND DISCUSSION

Table 6.1 shows the distribution of our test set. The first row contains the True Positive (TP) and False Negative (FN) values respectively. The second row contains False Positive (FP) and True Negative (TN) values respectively.

Table 6.1 Overall Classification of a test-batch

Overall Classification	**Positive**	**Negative**
Positive	203	38
Negative	37	10

$Precision = (203/(203 + 37)) = 0.8458$

$Recall = (203/(203 + 38)) = 0.8423$

$F - Measure = ((2 + 203)/(2 + 203 + 37 + 38)) = 0.844$

$Accuracy = ((203 + 10)/(203 + 37 + 38 + 10)) = 0.7473$

6.6 DOMAIN WISE ANALYSIS

The test dataset contains data related to three main domains Personal, Work and Academia. Table 6.2 shows the classification split up of the different domains. Using the formulae mentioned above, the domain-wise metrics can be obtained.

The domain wise distribution of evaluation metrics is shown below.

Table 6.2 Personal Domain -Spam classification of personal mails

Personal	Positive	Negative
Positive	68	12
Negative	11	0

Table 6.3 Work Domain -Spam classification of work mails

Work	Positive	Negative
Positive	67	13
Negative	12	5

Table 6.4 Academia Domain -Spam classification of academia mails

Academia	Positive	Negative
Positive	68	13
Negative	14	5

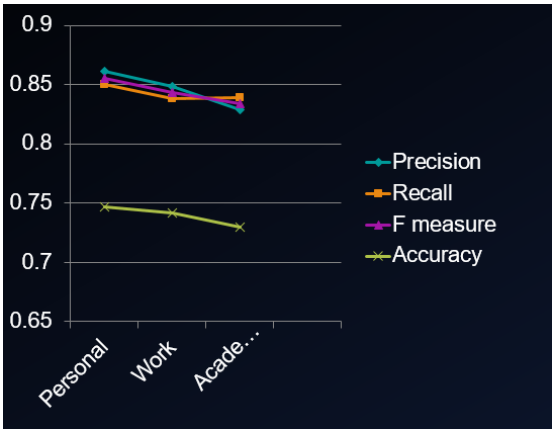

Figure 6.5 Domain Wise Analysis(a)

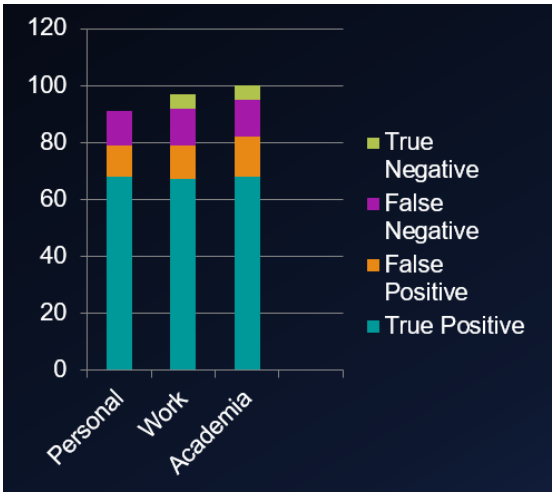

Figure 6.6 Domain Wise Analysis(b)

CHAPTER 7

CONCLUSIONS

7.1 SUMMARY

This is a classification system which classifies any email as ham or spam. The advantage of this system is that classification is done on the basis of contextual similarity. The system uses a Snowball stemmer to stem words and a TF-IDF Vectorizer to obtain a term-by-document frequency matrix. Singular Value Decomposition (SVD) is carried out to reduce the dimensionality following which the set of k (k can be varied) largest (strongest) features are selected. Finally, the selected features are fitted to a neural network which classifies each input email as spam or ham. In the Corpus Based Thesaurus approach, we find the cosine similarity between every two elements in the TF-IDF matrix. However, owing to severe memory and performance restrictions, this is difficult to obtain completely. The results of performance evaluation are very encouraging and show promising values for precision, recall and F measure of 0.846, 0.842 and 0.844 respectively. While looking at domain specific classification, precision, recall and F-measure scores obtained are all above 0.8.

7.2 CRITICISMS

The errors in the stemming propagate down to all the modules of the system and may influence final output. Neural networks typically have slow training speed. They are also very likely to get trapped in a

local minimum, thereby rendering the given task incomplete. The Corpus Based Thesaurus approach is very memory intensive and requires large amount of processing capability to complete entirely. More often than not, the CBT approach typically results in a hardware crash. Finally, some test cases resulted in false positive and rejection of legitimate mail. Most users would rather receive the spam email than lose the legitimate emails.

7.3 FUTURE WORK

Feature representation methods and classification algorithms are two critical components for spam filtering and even for data mining. This project investigates the effectiveness of semantic similarity measure feature representation methods and neural network algorithms, the combination of these two methods achieved very promising results for spam filtering. However, using improved neural network approaches such as Adaptive Backpropagation (ABPNN) are likely to improve classification efficiency even further, by eliminating the traditional drawbacks of neural networks. The ABPNN approach is likely to involve tuning the learning rate at every iteration. Our future work is to investigate more efficient feature representation methods and classification algorithms for data mining and apply the method to other data mining applications.

APPENDIX A

Test Cases for Each Module

This section provides the test cases for each of the modules of the system developed.

A.1 SNOWBALL STEMMER

A.1.1 Test Pre-requisites

Any text in English is to be given as input.

A.2 DESCRIPTION

The set of test cases to this module covers different English words with similar suffixes or roots.

A.3 TEST CASES

A.3.1 TC ID: 01

Input

Text with multiple words with the same root.

Expected Output

All words with similar roots are stemmed to give a simplified corpus.

A.3.2 TC ID: 02

Input

Text with multiple words with the same suffix.

Expected Output

The similar suffixes are stripped to give a simplified corpus.

A.4 TOKENIZER

A.4.1 Test Pre-requisite

Any text in English is to be given as input.

A.4.2 Description

The set of test cases to this module covers different English words and special characters.

A.4.3 Test Cases

TC ID: 01

Input: Text with multiple words. Expected Output: String is split into the required substrings.

TC ID: 02

Input: Text with multiple words and individual special characters. Expected Output: String and the special characters are split into the necessary components.

A.5 TF-IDF VECTORIZER

A.5.1 Test Pre-requisite

Any simplified English text corpus, after stemming and tokenizing, is to be given as input.

A.5.2 Description

All test case yield a standard type of output i.e the frequency counts of words in the corpus.

A.5.3 Test Cases

TC ID: 01

Input: Simplified email corpus Expected Output: Vectorized weighted term associations, based on frequency.

A.6 SINGLE VALUE DECOMPOSITION

A.6.1 Test Pre-requisite

The TF-IDF matrix from the previous step is to be given as input.

A.6.2 Description

Reduces the matrix dimensionality and extracts the important features

A.6.3 Test Cases

TC ID: 01

Input: Regular TF-IDF matrix. Expected Output: The output is a feature-reduced TF-IDF matrix.

A.7 MODIFIED DATASET CREATION

A.7.1 Test Pre-requisite

The semantic feature se obtained from the previous step is taken as the input for this step.

A.7.2 Description

The important features are chosen to create a new dataset which is used to fit the neural network

A.7.3 Test Cases

TC ID: 01

Input: Regular TF-IDF matrix with reduced features. Expected Output: A new feature set is created.

A.8 NEURAL NETWORK FITTING AND CLASSIFICATION

A.8.1 Test Pre-requisite

The semantic feature se obtained from the previous step is taken as the input for this step.

A.8.2 Description

This step produces the final output by classifying the test data as spam or ham.

A.8.3 Test Cases

TC ID: 01

Input: k-largest features chosen from the previous step Expected Output: Correctly classified as ham (true positive case).

TC ID: 02

Input: k-largest features chosen from the previous step Expected Output: Incorrectly classified as ham (false positive case).

TC ID: 03

Input: k-largest features chosen from the previous step Expected Output: Correctly classified as spam (true negative case).

TC ID: 04

Input: k-largest features chosen from the previous step Expected Output: Incorrectly classified as spam (false negative case).