ADAPTIVE ANTI-SPAM FILTERING

by

Levent Özgür

B.S. in Computer Engineering, Boğaziçi University, 2001

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science
in
Computer Engineering

Boğaziçi University

2003

ADAPTIVE ANTI-SPAM FILTERING

APPROVED BY:

Assist. Prof. Tunga Güngör . . . .
(Thesis Supervisor)

Prof. Fikret Gürgen . . . .
(Thesis Co-supervisor)

Prof. Ethem Alpaydın . . . .

Assist. Prof. Uluğ Bayazıt . . . .

Prof. A.C. Cem Say . . . .

DATE OF APPROVAL:

ACKNOWLEDGEMENTS

First, I would like to thank my supervisor Assist Prof. Tunga Güngör and co-supervisor Prof. Fikret Gürgen for their guidance. They supported me from the beginning and gave me a huge amount of time for this thesis.

I would also like to thank my thesis committee members: Prof. Ethem Alpaydın, Assist. Prof. Uluğ Bayazıt, Prof. A.C. Cem Say.

I wish to express my sincere gratitude to all my friends who give their valuable support throughout the year.

I would also like to thank all the people who, in a way, participated in this thesis by forwarding their spam and normal e-mail examples to specified addresses.

Finally I am deeply grateful to my family (including my two months old niece, İpek!). They mean too much things for me. Not only my English but also my Turkish was never enough to express my feelings to them.

ABSTRACT

ADAPTIVE ANTI-SPAM FILTERING

We propose an anti-spam filtering algorithm that is used for Turkish language based on Artificial Neural Networks (ANN) and Bayes Filter. The final product is an anti-spam filtering program which works compatible with Outlook so it is user-specific, thus adapts itself with the characteristics of incoming e-mails. The algorithm has two parts: the first part deals with morphology of Turkish words. The second part classifies the e-mails by using the roots of words extracted by morphology part. The input vectors to ANN are chosen with two models: based on binary model and probabilistic model. Two structures of ANN are employed in this study: single layer perceptron (SLP) and multilayer perceptron (MLP). Bayes Filter is also implemented with three different approaches: Binary Model, Probabilistic Model, Advance Probabilistic Model. Spam detection performance of the proposed system is improved by including non-Turkish words. A total of approximately 750 mails (410 spam and 340 normal) are used in the experiments. A success rate over 90% is achieved.

ÖZET

UYARLAMALI SPAM-ÖNLER FİLTRELEMESİ

Bu çalısmada, Yapay Sinir Ağları ve Bayes Filtresi uygulamalarını temel alarak Türkçe spam mesajlarını filtreleyen algoritma ortaya koyuyoruz. Sonuçta, Microsoft Outlook ile bağlantılı çalışabilen bir spam-önler filtresi ortaya çıktığı için, ürün son kullanıcıya yöneliktir ve bundan dolayı kullanıcıya özeldir; program kullanıcının spam ve normal mesajlarını öğrenerek kendini her kullanıcı için uyumlu hale getirir. Algoritmamızın iki temel kısmı var: Birinci kısım Türkçe kelimelerin morfolojisini incelerken ikinci kısım Morfolojik incelemeden gelen kelime köklerini kullanarak spam mesajları filtreler. Öğrenme algoritmalarının input vektörleri iki şekilde belirlenir: ikili model ve olasılık modeli. Bu çalışmada, Yapay Sinir Ağlarının tek katmanlı ve çok katmanlı algılayıcı birimleri kullanılırken; Bayes Filtresi de ikili modeli, olasılık modelini ve ileri olasılık modelini kullanmaktadır. Bu çalışma için yaklaşık 750 (410 spam, 340 normal) mesaj kullanı lmıstır. Filtrelemede 90%'dan yüksek başarı oranı elde edilmiştir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AOL	American Online
ANN	Artificial Neural Networks
ATN	Augmented Transition Network
FSA	Finite State Automaton
FSTN	Finite State Transition Network
LM	Learning Module
ML	Machine Learning
MLP	Multi Layer Perceptron
MM	Morphology Module
NLP	Natural language processing
SLP	Single Layer Perceptron

1. INTRODUCTION

In the past few years, internet technology has effected our daily communication style in a very strong way. With internet, e-mail is also used extensively in daily life nowadays. This technology makes it possible to communicate with many people at the same time in a very easy and cheap way. But, many e-mails are received by users without their desire. Thus, a problem occurs at the user-end due to many unwanted mails. Spam (can also be named as junk or bulk) mails are the general name for those types of e-mails. Spam mails, by definition, are the electronic messages posted blindly to thousands of people usually for advertisement. As time passes, much more percentage of the mails are treated as spam and this increases the level of seriousness of the problem.

In 1998 Canor and LaMacchia found that 10% of the incoming e-mails to the network was spam. New studies will surely show higher percentages. As another example, American Online (AOL) has stated that it has received 1.8 million spam mails from Cyber Promotions until precautions have been taken.

According to a research performed by Symantec in 2002 [1], 37% of the people receive over 1000 spam mails per week while 63% of them receive over 50. 65% of the participants waste at least 10 minutes for spam messages daily while 24% of them waste over 20 minutes.

According to another research in United States [1], IT departments of middle-sized companies waste 10 hours on average per week for updating their spam filters. On the other hand, Microsoft is known to employ 20 people working for the spam problem in all their time.

Some spammers do not only steal your time by bothering you with various advertisements, but also sell your nail addresses to other spammers which the problem more complicated and serious.

Many techniques have been proposed to alleviate this problem, but they are not completely satisfactory. For example the famous mail server "hotmail" performs its spam-filter according to addresses. If your address is seen on the mail, it accepts it, otherwise it treats it as spam. Needless to say, most spam mails pass this test and some important mails are treated as spam. Also, some servers try to collect addresses which are reported as spammers (people who send spam messages) and treat the mails coming from them as spam. However, spammers are all aware of most of these techniques so all those solutions lack the dynamic structure of the problem, which causes ineffective solutions.
Some more complex approaches were also proposed against these problems. Most of them were implemented by using machine learning methods. Naive Bayes Network Algorithms were used frequently and they have shown a big success in English spam mails. Knowledge-based and rule-based systems were also used by researchers for English spam filters [2, 3, 4].

1.1. Motivation

The aim of thesis is to present an algorithm for anti-spam filtering which is used for Turkish language based on ANN and Bayes Filter. In order to classify mails, a data set should be prepared containing spam and normal mail examples. The content of these mails are mapped to a space whose dimensions are the measured features of the text, e.g. words. For the classification process in Turkish, it is obvious that the main classifier of the texts should be roots of these words instead of all the words in the text or only the suffixes of these words. Here, Turkish version of the work seems more challenging when compared with English which needs a complex Morphological Analysis. There will also a final study for the thesis, in which a software will be produced that is compatible with Outlook and makes spam check online for the user.

1.2. Outline

We deal with the spam problem first in a general way by studying related studies performed in similar areas in Chapter 2. These related studies include Morphological

Analysis of Turkish, which is really an important subject for our subject. The studies also include learning algorithms implemented in Turkish, especially the ones that we will use in this study: Artificial Neural Network and Bayes Filter. We also deal with Microsoft Outlook programming concepts which enable us to make our filter user-specific. Then, our methodology is detailed in Chapter 2. Morphology module, learning module and outlook module and the integrity of all those models with our project is explained in detail in this chapter. In Chapter 4, results of several experiments are discussed. Chapter 5 summarizes the work done and discusses the future work which will improve the performance of the system.

2. RELATED RESEARCH

As mentioned in the previous section, machine learning algorithms should be applied for a dynamic, compatible solution of anti-spam filtering. There is also the concept of Morphological Analysis in this subject, which enables the learning algorithms work more accurate. Besides all these studies, producing a software which is compatible with a pop-3 mail program and performs spam check in the inbox is also in the scope of this thesis. For this purpose, we have also dealt with Outlook programming which enables us to implement this type of product.

2.1. Previous Studies

In the literature, Naive Bayes Filter was implemented frequently for anti-spam filtering [2]. Sahami proposed a Naive Bayes Filter Model in which each message was represented by a vector $X = (x_1, x_2, x_3, \ldots, x_n)$, where these were the values of $X_1, X_2, X_3, \ldots, X_n$ [5, 6]. They used binary attributes: zero or one depending on whether some characteristic represented by x_i was present in the message. Attributes corresponded to words in their study and to select those critical words, they used mutual information (MI) of each candidate attribute X with the category-denoting variable C:

$$MI(X,C) = \sum_{x \epsilon 0,1, c \epsilon 0,1} P(X=x, C=c) * log \frac{P(X=x, C=c)}{P(X=x) * P(C=c)} \tag{2.1}$$

The attributes with the highest MIs were selected and from the Bayes Theory, they calculated a score for each mail and each class to categorize these mails.

In addition to this study of Sahami, Androutsopoulos used the concepts of cross-validation (using different parts of the data for the tests in different runs), lemmatiza-

tion (finding the root of words) and stop-lists (preventing words occurring too much in the language) [2]. Over 90% success was obtained in these studies.
Besides these studies, Cohen proposed a model of Feature Selection and Feature Extraction for text categorization [7]. In that study, the optimal feature set size for word-based indexing was found to 10-15 (which was surprisingly low) features despite the large training sets.

Dogan proposed a model for text categorization based on Mistake-Driven Learning [8]. This method used words in a probabilistic model for the formation of feature vectors rather than binary model. This was performed to tolerate the length variation of documents and significant success was achieved in this study.

Besides these studies, learning rules were used for categorization of mails by Cohen which resulted in steep learning curves with rapid learning rates in which learning took place with around 200 examples [4].

2.2. Morphology Module

2.2.1. General Concepts

Computational linguistics stands for the study between the general discipline of linguistics and artificial intelligence which is a subset of computer science.

Natural language processing (NLP) is one of the most important subfields of computational linguistics. It generates and understands language forms which have high complexity.

We can categorize NLP in four groups [9]. These groups are:

- Acoustic-phonetic: It is the process at the core level. It takes the sounds and translates the input into words.
- Morphological: It gets the words represented in written language as input and

converts them into some kind of representation (finding the root of the word, the category, the feaures of the word, the affixes,etc.) by processing them with the use of several language elements (lexicon, morphological rules, etc.).

- Syntactic: It gets the individual words as input from the Morphological step and combines them into larger language elements by checking grammatical rules of the language.
- Semantic-pragmatic: It is the final process which takes the syntactic representation and understands it.

Of all these above, we will deal with Morphological Part of NLP in this thesis. For the learning module, which will be explained in the following sections, there is no need to store every form of word each with different affixes. Instead, we will get the words in mails and according to the morphology module (MM), we will get the root form of all words.

According to the study of Güngor we have implemented morphological analysis of Turkish words by using C programming language [9]. Several steps are performed during this analysis:

- An Augmented Transition Network (ATN) formalism is introduced which contains all of the categories of words and the suffixes (14 categories, 200 suffixes).
- A root lexicon containing 21,500 words and a proper noun lexicon containing 11,500 words are used during this study.

2.2.2. Formalism Types

Formalism types are different techniques that are well-known in language theory. They are widely used for the development and analysis of programming languages. Below are the four of them:

Finite State Transition Networks: A Finite State Automaton (FSA) is the simplest computing machine. A given automaton produces two types of output (True or

False) according to the correctness of the grammer of the sentence. A Finite State Transition Network (FSTN) is, in fact, as a specification of FSA to recognize elements of the given language.

Finite State Transducers: It is a recognizer which emits an output string during each move made

Recursive Transition Networks: Recursive Transition Network is just like an FSTN except that it introduces the extra concept of a named subnetwork. So it is possible for an arc to name a subnetwork to be traversed, instead of a specific input that is to appear.

Augmented Transition Networks: With Augmented Transition Networks (ATN), we use registers for storing information. Registers hold values for comparison. It is possible to shuffle information between registers when network is traversed.

An ATN is an 8-tuple M=(Q,A,P,K,O,R,q0,Z0), where:

- Q is a finite set of states or nodes
- A is a finite set of acceptable input symbols
- P is a finite set of symbols that may appear on the stack
- K is a mapping from QXAXPXRi to QXP*XR0, where Ri and R0 represent the contents of the registers before and after the transition
- O is a finite set of operations for storing information in the registers
- R is a finite set of operations for reading information stored in registers
- q0 in Q is the initial state
- Z0 in P is the start symbol, i.e. the symbol that appears initially on the stack

In our project, we have used ATN model; because with other models, large vocabularies of input and output languages lead to large networks which can not be expressed concisely [9]. With ATN, registers manage this complexity of languages.

2.2.3. Turkish Morphology

Table 2.1. Turkish alphabet

Type	Letters
Turkish alphabet	a,b,c,ç,d,e,f,g,ğ,h,ı,i,j,k,l, m,n,o,ö,p,r,s,ş,t,u,ü,v,y,z
Vowels	a,e,ı,i,o,ö,u,ü
Low vowels	a,e,o,ö
High vowels	ı,i,u,ü
Rounded vowels	o,ö,u,ü
Unrounded vowels	a,e,ı,i
Back vowels	a,ı,o,u
Front vowels	e,i,ö,ü
Consonants	b,c,ç,d,f,g,ğ,h,j,k,l,m,n, p,r,s,ş,t,u,v,y,z
Voiced consonants	b,c,d,g,ğ,j,l,m,n,r,v,y,z

In this section, we will deal with Turkish morphology from a view point of linguistic and combine this research with other modules of the thesis. To study the linguistic part, we should first define Turkish alphabet and categorize the letters in the alphabet as seen in Table 2.1.

We can categorize the morphological analysis of Turkish in two steps: Morphophonemic analysis, Morphotactic analysis.

2.2.3.1. Morphophonemic Analysis. There are some morphophonemic rules in Turkish, which determine the form of the suffixes. The form of a suffix that will be affixed to a word may change according to some properties of that word. Below are different types of morphophonemic rules [9]:

Vowel Harmony: Vowel Harmony is the process in which the vowels in all syllables of a word except the first one assimilate to the preceding vowel with respect to certain phonetic features. In Turkish, this is performed as left-to-right, in which operating is performed sequentially from syllable to syllable according to some rules:

- A non-initial vowel assimilates to the preceding vowel in frontness
- A non-initial high vowel assimilates to the preceding vowel in rounding
- A non-initial low vowel must be unrounded; that is, the vowels o and ö do not occur except in first syllables of the words.

Table 2.2. Inflectional suffixes for nouns

Types	Suffixes
Plural Suffix	*-lar*
Possessive Suffixes	*-(I)m, -(I)mIz, -(I)n, -(I)nIz, -(s)I, -larI*
Case Suffixes	*-DA, -CA, -Dan, -Il, -(y)A, -sIz, -(y)I, -(y)lA, -(y)InAn -(y)Ilan*
Genitive Suffix	*-(n)In*
Relative Suffix	*-ki*

Consonant Harmony: Say that the word is in its root form or a suffix beginning with a consonant is affixed. In multi-syllabic words and in certain mono-syllabic roots, the final voiced consonants b,c,d,g (or ğ)generally become voiceless (it changes to p,ç,t,k). Also in some suffixes beginning with one of the consonants c,d,g, this initial consonant may change according to the last phoneme of the word it follows (ç,t,k).

Root Deformations: Turkish roots are usually not flexed. On the other hand, there are some situations in which phonemes are changed by assimilation or various other deformations. For example , personal pronouns (ben - bana, sen-sana) may be flexed. There are also some other examples concerning root deformations.

2.2.3.2. Morphotactics. The morphotactic analysis determines the order of the suffixes. There are predefined categories (noun, verb, etc.) in the language and each type

Table 2.3. Inflectional suffixes for verbs

Types	Suffixes
Voice Suffixes	*-(I)n, -(I), -Ar, -Ir, -(I)t, -DIr, -t,-(I)l, -(I)n*
Negation Suffixes	*-mA, -(y)AmA*
Compound Verb Suffixes	*-(y)Adur, -(y)Akoy, -(y)Agel, -(y)Ayaz, -(y)Agor* *-(y)Iver, -(y)Akal*
Mood Suffix	*-(y)Abil*
Main Tense Suffixes	*-(A)r, -(y)A, -DI, -(y)AcAĞ, -(I)r, -(y)AlIm, -(I)yor* *-(y)In, -mAktA, -malI, -mIş, -(y)sA*
Question Suffix	*-mI*
Second Tense Suffixes	*-(y)DI, -(y)mIş*, -(y)sA
Person Suffixes	*-m, -(y)Im, -n, -sIn, -k, -(y)Iz, -nIz, -sInIz, -lAr*
Predicate Suffix	*-DIr*

has its own affixes. Also some suffixes are valid after some other suffixes which have been previously affixed to a word and some not.

Below are the main predefined categories:

- Nouns: Table 2.2 shows the inflectional suffixes that can be affixed to nouns.
- Verbs: Table 2.3 shows the suffixes that can be affixed to verbs.
- Verbal Nouns: Sentences can be classified in two groups in Turkish, as verb sentences and noun sentences. Verb sentences show an action represented by a verb. On the other hand, in a noun sentence there is no explicit verb. (e.g. öğrenciyim - I am a student) In those sentences some suffixes are used as verb ‘to be‘ in English. Table 2.4 shows those suffixes.
- Participles: Verb sentences can be transformed into a noun, adjective, or adverb clause by adding certain derivational suffixes to the verb of the sentence. Table 2.5 shows those types of suffixes.

Table 2.4. Inflectional suffixes for verbal nouns

Types	Suffixes
Question Suffix	*-mI*
Tense Suffixes	*-(y)DI, -(y)mIş, -(y)sA*
Person Suffixes	*-m, -(y)Im, -n, -sIn, -h, -(y)Iz, -nIz, -sInIz, -lAr*
Predicate Suffix	*-DIr*

Table 2.5. Some derivational suffixes affixed to verbs

Types	Suffixes
Noun and Adjective	*-DI-u G, -(y)AcA(-u G), -mA, -(y)An, -mA-u G* *-(y)AsI, -mIş, -(y)Iş*
Adverb	*-cAsInA, -(y)IncA, -mAdAn, -(y)IncA, -mAksIzIn, -(y)Ip* *-(y)All, -(y)ken*

2.3. Learning Module

Two different learning algorithms are implemented for this study: Artificial Neural Network and Bayes Decision Theory.

2.3.1. Artificial Neural Network

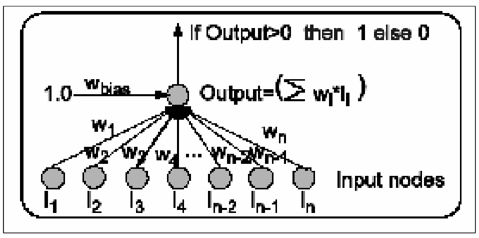

Figure 2.1. Single Layer ANN

2.3.1.1. Definition. Artificial Neural Network (ANN) is a machine learning algorithm that is inspired by the way biological nervous systems, such as the brain, process information. In this system, neurons, which are highly interconnected processing elements work together to solve specific problems. There are two phases in ANN as in many other learning algorithms [11, 14]:

- Training Phase:A training data set is used to determine the weight parameters that define the neural model. This trained neural model will be used later in the test phase to process real test patterns and yield classification results. Weight of each neuron or each interneuron connection is calculated during this phase.
- Test Phase:Knowledge is acquired by the network through these weights and values of inputs (neurons) in the test phase.

2.3.1.2. Types of ANN - SLP and MLP. Two kinds of ANN, namely, Single Layer Perceptron (SLP) as in Figure 2.1 and Multi Layer Perceptron (MLP) are implemented in this study. In the training phase, weight values of neurons change according to the following formula for SLP:

$$w_i(t+1) = w_i(t) + (err)I_i \tag{2.2}$$

After several loops, weights take their final values which give minimum error. In the test part, unseen data are introduced to the network.

When we implement ANN with hidden layer nodes (MLP), the complexity of the problem increases. There are hidden nodes, which are represented by the vector $Z = (z_1, z_2, z_3, \ldots, z_n)$. The weight of those hidden layer nodes are represented by the vector $V = (v_1, v_2, v_3, \ldots, v_n)$. The program is implemented according to the following algorithms:

$$v_i(t+1) = v_i(t) + (err)I_i \tag{2.3}$$

$$w_{hi}(t+1) = w_{hi}(t) + (err)Ihi \tag{2.4}$$

where $(err)Ihi$ is dependent on Vi(t)

2.3.1.3. Types of ANN - Supervised and Unsupervised. The neural networks are commonly categorized in terms of their corresponding training algorithms: unsupervised networks, and supervised networks.

Supervised learning networks have been the mainstream of neural model development. The training data consist of many pairs of input/output training patterns.

For an unsupervised learning rule, the training set consist of input training patterns only. Therefore, the network is trained without benefit of any teacher.

We have both input (mails) and output(decisions(whether they are spam)) patterns in this thesis so our model will be based on supervised learning in this study.

2.3.1.4. Usage of ANN. Neural networks have found many successful applications in computer vision, signal/image processing, speech/character recognition, expert systems, medical image analysis, remote sensing, robotic processing, industrial inspection, and scientific exploration [12].

Some popular applications of neural networks also include [13]:

- Forecasting applications Banking: predicting the issue of loans (credit scoring), predicting the recovery of bad loans (recovery scoring). Forecasting the behaviour of new customers. Identifying good risks and offering them loans.
- Finance: predicting share prices, volatilities. Help for position-taking. Portfolio and asset management. Industry: predicting demand for a product or service for better production planning. Servo-control for machines or chemical reactors (through model inversion).

- Marketing: predicting which prospects will become customers (marketing targeting) or answer a mail solicitation (mailing targeting). Turnover prediction. Public life and administration: analysis and prediction of crime, tax return analysis for fraud detection; risk analysis for renters; economic forecasting; etc.

2.3.2. Naive Bayes Classifier

2.3.2.1. Definition. Bayes Classifier(Filter) is a fundamental statistical approach in machine learning [15]. It uses probability and the costs that accompany such decisions so it is based on quantifying the tradeoffs between various classification decisions. We will study Bayes Filter by giving an example in mail classification [16]. In our study there are two types of e-mails: spam and normal. We let C denote these categories, with $C = C_1$ for spam and $C = C_2$ for normal. In this theory, there is a some prior probability $P(C_1)$ that the next new mail is spam and some prior probability $P(C_2)$ that it is normal. If there are no other types, these probabilities sum to one. So, if we want to categorize mails without any knowledge about them, it seems logical to use the following decision rule:

Decide C_1 if $P(C_1) > P(C_2)$; otherwise decide C_2

Luckily, we are not usually asked to make decisions with so little information. For example, in our study we have used the words in the mails for categorization. A mail with more "spam-like" words are categorized as spam and vice-versa. Different combinations of words yield different categories. For categorization, probability of input vector values occurring in normal mails and spam mails are calculated. This is called the class-conditional probability density function and is expressed as $P(X|C)$. It is the probability of X input vector occurring given that the class is C. Let us suppose that, we know both the prior probabilities $P(C_1)$, $P(C_2)$ and the conditional probabilities $P(X|C_1)$, $P(X|C_2)$. Also say that, $P(X)$ is the probability of the input, which is independent of the classes. Bayes Theory, here, offers a formula which leads us to categorization of mails [16]:

$$P(C_i|X) = \frac{P(C_i) * P(X|C_i)}{P(X)} \tag{2.5}$$

Just as in ANN, feature vector X can be decomposed as $(x_1, x_2, x_3, \ldots, x_n)$, where each is a special attribute of the feature vector. So the formula becomes:

$$P(C_i|X) = \frac{P(C_i)}{P(X)} * \prod_{j=1}^{n} P(X_j|C_i) \tag{2.6}$$

Here, a usual transformation takes place by getting the logarithm of the equation, eliminating P(X) variable which is common in all classes and also eliminating $P(C_i)$, which we consider same for both classes [11]:

$$P(C_i|X) \sum_{j=1}^{n} log(P(X_j|C_i)) \tag{2.7}$$

As it is seen, this operation does not compute a probability but a score that is sufficient to find the most suitable class for the input. In our implementations, we use different forms of this equation which will be explained in the next section.

2.3.2.2. Usage of Naive Bayes Classifier. Naive Bayes is a well known and studied algorithm both in statistics and machine learning [15]. The term *naive* comes from the assumption that the attributes are independent given the class. It is said to have a limited capacity for information storage that only depends on the number of features and number of values of the features [15]. On the other hand, it has a surprisingly good performance in a wide variety of areas, including many where there are clear dependencies between features.

2.4. Outlook Model

Although outlook is known only to be a famous mail program, one of the great advantages of it is the fact that, it is also suitable for developing user-specific applications which enable us to make our anti-spam filter program work compatible with Outlook.

Outlook provides five key building blocks for developing applications: folders, fields, views, forms and actions [22].

2.4.1. Folders

Folder is a focal point for any Outlook application. They hold data; views for that data, forms, agents and rules. They provide users with a storage location for information and a hierarchy structure that makes finding information easy. Inbox is said to be the most popular and main folder in Outlook. It is possible to create and customize extra folders in Outlook [22].

2.4.2. Fields

Fields are named variables where Outlook stores the data for your application. These fields are associated with properties of their folders. For example, possible fields in our Inbox include From, To, Subject and Message. It is possible to create and customize extra fields in folders in Outlook.

2.4.3. Views

Views enable us flexibility in the presentation and organization of information, which can be used in any type of Outlook folder. Outlook supports five type of views:

Table view, timeline view, card view, day/week/month view, icon view. It is possible to create and customize extra views in Outlook.

2.4.4. Forms

Forms in the Outlook enables us to manage other Outlook items. It allows us to distribute and collect information electronically both inside and outside your organization. Possible form types in Outlook are message forms, post forms, contact forms, office document forms [22]. Forms are designed in Design Mode, in which you use controls, action buttons, text fields for the user to control the application. Built-In Outlook controls are label control, textbox control, listbox control, combobox control, checkbox control, optionbutton control etc. In Outlook there is the form library in which we can publish our forms after designing them. So the designed forms become available to the user controls in the main Outlook window.

2.4.5. Actions

Actions are either built-in or custom responses to a particular item on a form. Possible actions in Outlook are Reply, Reply To All, Forward. It is possible to create actions in Outlook.

2.4.6. VBScript

All these tools are sufficient structures for developing applications in Microsoft Outlook but on the other hand, they are not enough for a complete user application. A programming environment is needed for the programmer to develop his own application by combining all these tools. Outlook provides this built-in development environment that uses the Microsoft Visual Basic Scripting Edition (VBScript) programming language [22]. Using VBScript routines, we can write procedures to manipulate items, folders, controls and the other Outlook objects.

3. PROBLEM STATEMENT AND METHODOLOGY

In this research our aim is to design an anti-spam filter for Turkish language. For this purpose, a data set is used with more than 800 mails. As shown in Figure1, a Turkish morphological program is prepared that uses this data set and returns the root form of the words in the data set for the use of the Learning Module (LM) of the project. Also an outlook interface is prepared in this study for the user to get the mail contents online and run the anti-spam filter program.

3.1. Data Set

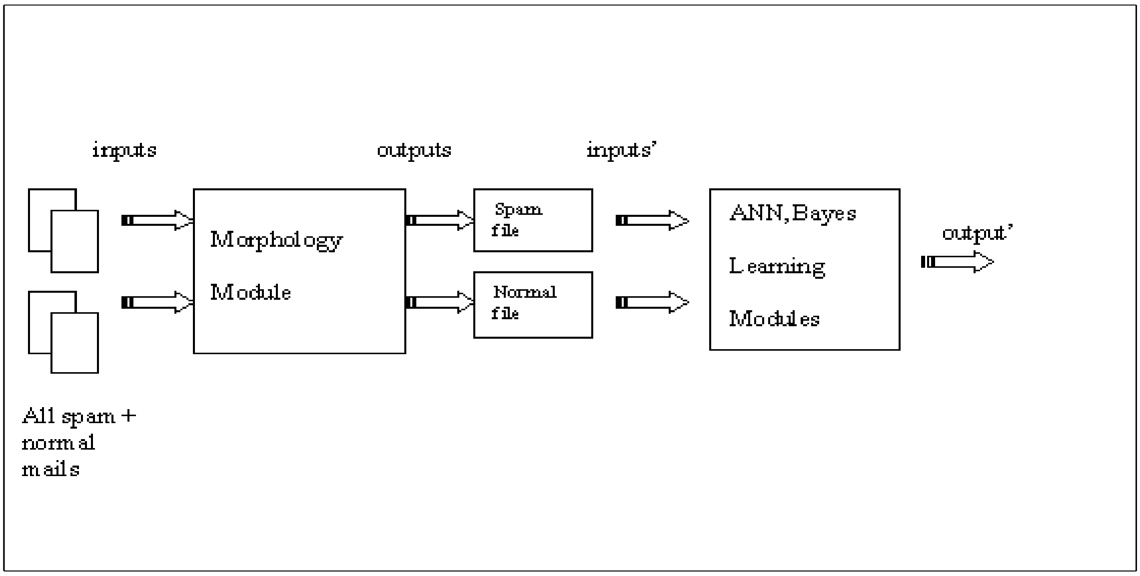

Figure 3.1. Main Steps of the Program

Spam and normal mails are collected in the following addresses:

- turkcespam@yahoo.com
- junk@cmpe.boun.edu.tr
- turkcejunk@hotmail.com
- turkcenormal@hotmail.com

As seen in Figure 3.1 The MM uses these mails to create two main files (spam and normal) for the use of the LM which will classify the e-mails. For the runs, 337 spam mails are obtained randomly from these addresses and by the help of the morphology program, one file is created that stores the root of each word for each spam mail. Then 247 normal mails are obtained from daily normal mails and another file is created that stores the root of each word for each normal mail.

3.2. Morphology Module

For the MM, we have studied two main implementations. One of them was created by using Pascal language and the other by using Prolog language [9, 10]. These were successful implementations with more than 90% success in finding the correct root form of the words. Prolog program was slower than the Pascal program. Also, an executable file can be generated by any Pascal program which enables the program to run without the support of its compiler. With Prolog language, we do not have this chance. So, for all these reasons we have selected to work with Pascal program.

We have decided to implement all the LMs in C programming language because, compared to Pascal programming language, C is still used widely in programming and also is more powerful with its more libraries and functions [17, 18]. So we have prepared the morphology program also in C language to be compatible with the LM.

A Turkish morphological analysis program is prepared based on a research on turkish morphology [9]. Turkish is an agglutinative language which makes morphological process more challenging. In Turkish, given a word in its root form, we can derive a new word by adding an affix (prefix, infix or suffix) to this root form, then derive another word by adding another affix to this new word, and so on. This iteration process may continue several levels. A single word in an agglutinative language may correspond to a phrase made up of several words in a non-agglutinative language. This makes the language more complex than non-agglutinative languages like English.

Another important issue that is dealt with in this work is the use of special Turkish characters (‘ç’,‘ğ’,‘ı’,‘ö’,‘ü’,‘ş’). These usually are not supported in many text fields; so people frequently use ('c','g','i','o','u','s') instead of these characters. In this study, all possible variations in a given word are examined (i.e. all possible words are formed by replacing each of the above-mentioned characters by its corresponding special Turkish character) until an acceptable Turkish root is found.

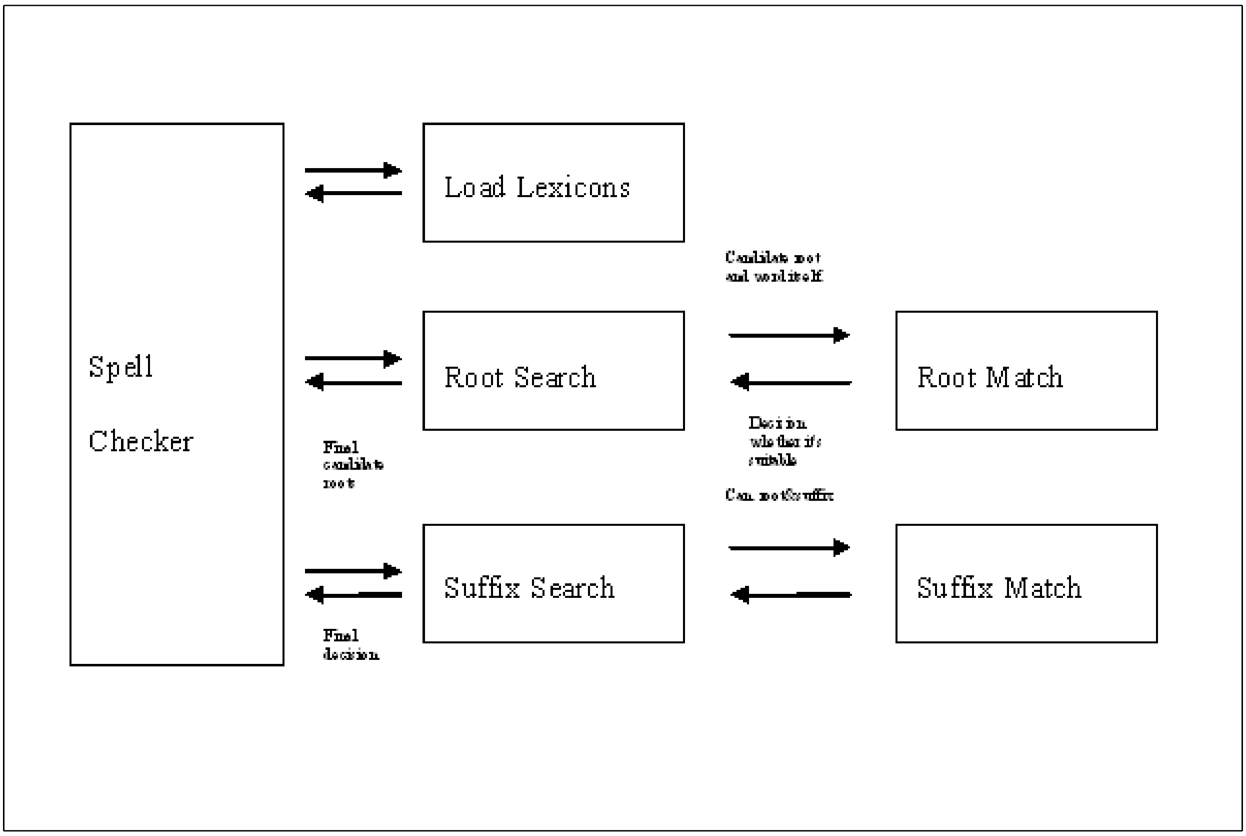

Figure 3.2. Spelling Checker

3.2.1. Implementation of the Morphology Module

In our study, there are two independent lexicons, one for root words and the other for the suffixes. Our module parses words according to the rules of the morphotactics of the language. These approaches may be classified in two groups:

- Suffix Stripping: Parsing is performed from-right-to-left of the word. It begins the analysis at the end of the word, trying to match a substring from the end of the word with a suitable suffix from the suffix lexicon. If any match is found, this process goes on by parsing the remainder of the word after stripping that suffix off.

- Root Matching: Parsing is performed from-left-to-right. It begins the analysis at the beginning of the word, trying to match the initial substrings of the word with a suitable root from the root lexicon. When a suffix is found to match a further substring of the word in a permitted way, what class of suffixes may follow is determined. If the end of the word can be reached by iteration of this process and if the last suffix analyzed is one that may end a word, the parse is successful. We have chosen root matching algorithm because this approach narrows the choice of possible suffixes that can be combined with a stem of the current stem category at every step [19].

After selecting the algorithm for parsing, implementation part is performed. As seen in Figure 3.2, first lexicons were loaded in the program. Then root search and root match algorithms were performed. then suffix search and suffix match for the given roots were performed.

<u>3.2.1.1. Lexicons.</u> The lexicons are the place where almost all the information for spelling checking is stored. At the beginning of the program, all the lexicons are loaded to suitable dynamic structures for the program usage. We have used three types of lexicons in our program: root lexicon, proper noun lexicon, suffix lexicon [19].

Root lexicon holds all the root words (more than 22,000). In our program, the root lexicon is kept in a sorted form in an array. First two letters of the word is used to search the suitable root for the word. Each root has the following information:

- the word
- its grammatical categories
- a number of flags

For proper noun lexicons, the proper nouns are separated and put into a special lexicon. The structure of the lexicon is similar to the root lexicon. On the other hand, hashing is performed by three letters and there is no field for the grammatical categories.

The implementation of the suffix lexicon mirrors the transition network representation of the morphotactics of the language [19]. Each suffix has the following information:

- the suffix
- the type of the suffix
- the flag of the suffix
- termination flag
- category of the word after this suffix is used
- a pointer to the next suffix node
- flag for suffix signalling: whether it deforms and whether it drops.

3.2.1.3. Root Search. For the root search, first the program produces possible candidate roots by matching an initial substring of its input to the roots in the root lexicon. These are tested whether it is the correct one in the suffix search part.

3.2.1.4. Suffix Search. The root category of each candidate holds the possible suffixes that can be affixed to the word. The parser tries to find a suffix that matches the surface string. This recursive process continues until the end of the input string is reached or there is no suitable suffix and root left for the input.

3.2.1.5. Performance. The spelling checker was found to check approximately 3000 words/min with a success rate over 90%. Because of the Turkish character problem, checking of an average sized mail took approximately six seconds. In order to make the program faster, we have limited the number of trials of possible variations to the number 16 instead of trying all possible variations of the word. Spelling check rate has become 2-3 seconds/mail on average after this change.

3.3. Learning Module

Two main learning algorithms were implemented in this study: ANN and Bayes Approach.

3.3.1. ANN

As mentioned in the previous section, two models were selected in ANN: SLP and MLP. Pre-processed for ANN, details in training and test part would be explained in this part.

3.3.1.1. Pre-processes for ANN. For all these Neural Network implementations, we first formed the feature vector and then specified the weights of the nodes in the network.

- Formation of Feature Vector: We can define feature vector as the group of critical words that are used in classification of mails as spam or normal. For the formulation of feature vectors, we first selected the candidates for feature vectors. In this study, candidates for these critical words were all the words in the training data (For some runs, only the words in spam training data were used). Given categories (c:spam or normal) and candidates for feature vector, selection was performed by finding Mutual Information of them [2]:

$$MI(X,C) = \sum_{x\epsilon 0,1, c\epsilon 0,1} P(X=x, C=c) * log \frac{P(X=x, C=c)}{P(X=x) * P(C=c)} \tag{3.1}$$

  The attributes with the highest MIs were selected to form the feature vector. These highest valued words were most probably the words occurring frequently in one class of mails and not so much in the other. So these selected words were said to be the best classifiers for spam and normal mails.

- Weight of Nodes: As mentioned above, feature vector was used for classification. Each message was represented by a vector $X = (x_1, x_2, x_3, \ldots, x_n)$. In our study, words are represented by x_i. Calculation of this value is performed in two ways: Binary and Probabilistic Model. In Binary Model, the problem is whether the word occurs in the text:

$$x_i = \begin{cases} 1 & \text{,where } x_i \text{ occurs in the given mail} \\ 0 & \text{,otherwise} \end{cases} \tag{3.2}$$

As seen, this algorithm does not take the length of the text into consideration. So, when the text is very long or very short, wrong results are likely to occur. The second algorithm takes the length of the text into consideration. Number of occurrences of the word is also important. The weights of the nodes are, in fact, the frequency of that word in the text and calculated by the probability formula:

$$X_i = \frac{\# \, of \, occurrances \, of \, that \, word \, in \, the \, text}{\# \, of \, all \, words \, in \, the \, text} \tag{3.3}$$

<u>3.3.1.2. Test Part.</u> After the training part of ANN algorithm, given the vector $X = (x_1, x_2, \ldots, x_n)$, weights of each input were calculated and tests were performed in the second part. During implementation, a large portion of the data set was used for training data and a small portion was chosen (usually 50 spam, 50 normal examples) for tests. 6 different runs were performed by changing these training and test examples. So, different parts of the whole set were specified as test data during these runs and finally, the average of the results were taken as final results.

3.3.2. Bayes Approach

For Bayes Approach, feature vector that contains the inputs was formed as same in ANN. Then, considering those inputs, three main algorithms were implemented to score classes for each of these inputs: Binary Model, Probabilistic Model, Advance

Probabilistic Model.

3.3.2.1. Binary Model. In this model, we calculated the score for each e-mail belonging to class i as follows:

$$P(C_i|X) = \begin{cases} c * \sum_{j=1}^{n} prob[j][i] & \text{,where input } j \text{ occurs in this example} \\ -\sum_{j=1}^{n} prob[j][i] & \text{,otherwise} \end{cases} \tag{3.4}$$

Here, $P(C_i|X)$ was calculated by dividing the number of mails containing that input by the total number of mails belonging to that class. In mails, occurrence of an input word usually indicates a stronger idea for classifying mails than a non-occurrence of that word. c indicates the level of this strength. In the test, c was taken between 1-14 in different runs.

For a mail, the values for each input was added for both two classes (spam, normal). The resulting class was chosen as the class with more points as shown below:

$$\text{Decide } C_1 \text{ if } P(C_1) > P(C_2) \text{ ; otherwise decide } C_2$$

3.3.2.2. Probabilistic Model. In this model, we calculated the score for each e-mail belonging to class i as follows:

$$P(C_i|X) = \begin{cases} c * \sum_{j=1}^{n} prob[j][i] * hitNo & \text{,where input } j \text{ occurs in this example} \\ -\sum_{j=1}^{n} prob[j][i] & \text{,otherwise} \end{cases} \tag{3.5}$$

Here, $P(C_i|X)$ is the total number of that input word j in all mails belonging to class i; not just the number of mails containing that word. As mentioned above, c

is the level of strength for the words occurring and *hitNo* is the total number of that input word in this specified mail.

For a mail, the values for each input was added for both two classes (spam, normal). The resulting class was chosen as the class with more points as shown below:

Decide C_1 if $P(C_1) > P(C_2)$; otherwise decide C_2

<u>3.3.2.3. Advance Probabilistic Model.</u> In this model, we calculated the score for each e-mail belonging to class i as follows:

$$P(C_i|X) = \begin{cases} c * (\sum_{j=1}^{n} prob[j][i] * hitNo)/counter & \text{,where input } j \text{ occurs in this example} \\ -\sum_{j=1}^{n} prob[j][i] & \text{,otherwise} \end{cases} \tag{3.6}$$

Here, $P(C_i|X)$ is the total number of that input word j in all mails belonging to class i; not just the number of mails containing that word. As mentioned above, c is the level of strength for the words occurring and *hitNo* is the total number of that input word in this specified mail. *counter* is the number of total words in this mails. So in this model, not only the frequency of the words was calculated but also the length of the text was considered.

For a mail, the values for each input was added for both two classes (spam, normal). The resulting class was chosen as the class with more points as shown below:

Decide C_1 if $P(C_1) > P(C_2)$; otherwise decide C_2

3.4. Outlook Model

Up to now, what we have implemented in this study are the training and test Programs that perform spam tests on the data coming from the morphological study

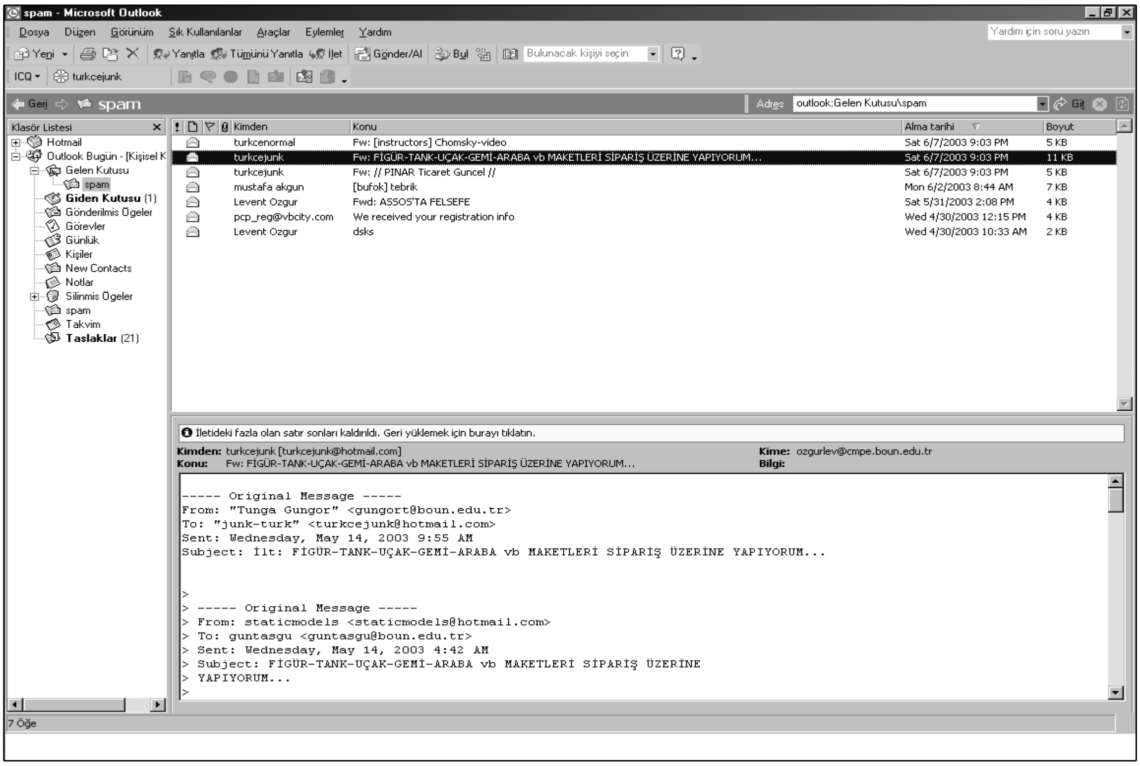

Figure 3.3. Spam folder of Outlook

of the mails stored in certain text fields as explained in the previous sections. But, our subject is anti-spam filter and it is a fact that this spam filtering is better performed online for the users while checking their mails. Outlook is said to be most popular mail program nowadays, and as explained above, and it is possible to make your programs run compatible with Outlook. So we have added an Outlook interface to our project for the users to avoid spam mails online with our filter using ANN module while checking their mails.

In the Outlook toolbar, we can reach our program from the 'Select Form' in the Tools menu. Our program uses the following necessary files to filter mails:

- Morphology program
- Necessary lexicons and other files for Morphology Program
- Learning program (two versions: training and test)
- Text file that holds the weight values for ANN module

Our VBScript program first creates a folder called Spam inside the folder Inbox to host the spam mails filtered [20, 21, 22]. Then, if user chooses to filter mails, according to the weights in the text file, mails are classified as spam or normal and spam mails are transferred to Spam folder as seen in Figure 3.3.

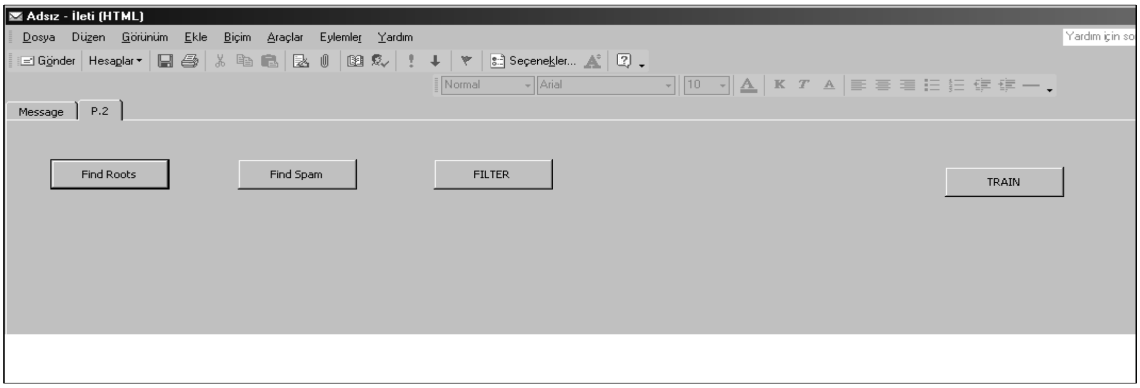

Figure 3.4. Design of the form

If the user chooses to train his data as seen in Figure 3.4, mails in his Inbox are selected as normal mails and the other mails in his Spam folder are selected as spam mails. With the other mails in the text file, training phase of ANN is performed and new weights of the inputs are printed in the text file holding weight values.

4. TEST RESULTS

Six different runs were performed for each algorithm and for each number of parameters ((10,40,70) for ANN algorithms and (1,11,21,31,41) for Bayes Approach). Learning coefficient was taken as 0.1 for ANN. Each run has taken a different part of the data set as training and test; so the average of all the results became the result of that algorithm for that number of input.

All these algorithms were implemented in Visual C++ 6.0, under Windows XP, on an Intel Pentium PC running at 1.6 GHz with 256 MB RAM.

Below, there is the comparison of runs for each algorithm. There, the success rates refer to rates for the best cases in the test runs if nothing else is indicated.

4.1. Test 1

In SLP, two different weight specification algorithms, namely, probabilistic model and binary model were used which were explained above. Probabilistic model seemed to achieve more successful results; so with MLP, probabilistic model was used.

So, four different algorithms were implemented, namely, SLP with binary model, SLP with probabilistic model, MLP with 150 hidden layer nodes and MLP with 250 hidden layer nodes. These algorithms were implemented with three different input numbers: 10-40-70. The inputs were selected as the words occurring frequently in spam mails and seldom in normal mails. Only Turkish words were admitted to the LM. This means, when a root form could not be found, that word was not included in the LM.

The results of the implementation are shown in Figure 4.1, Figure 4.2, Figure 4.3and Figure 4.4. In the figures, algorithms are represented as:

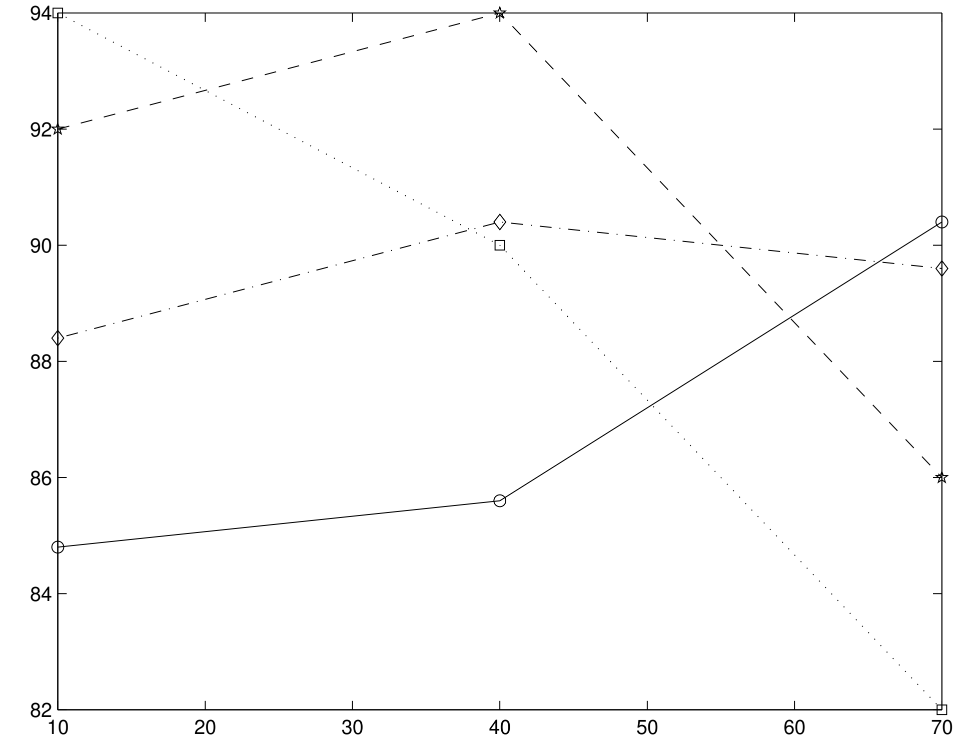

Figure 4.1. Several implementations of ANN - test normal data

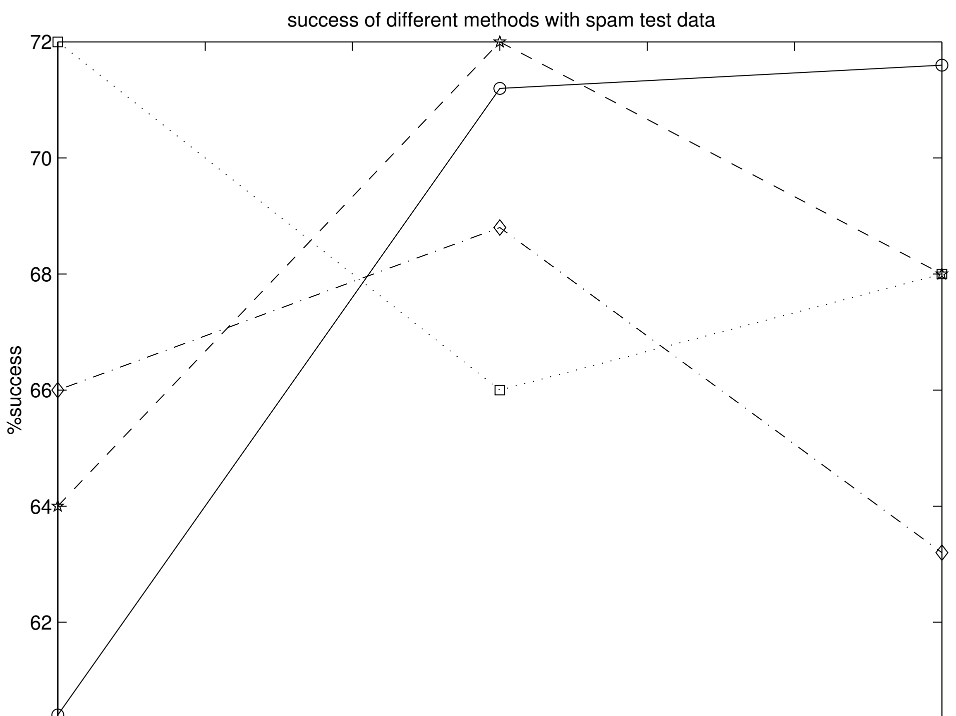

Figure 4.2. Several implementations of ANN - test spam data

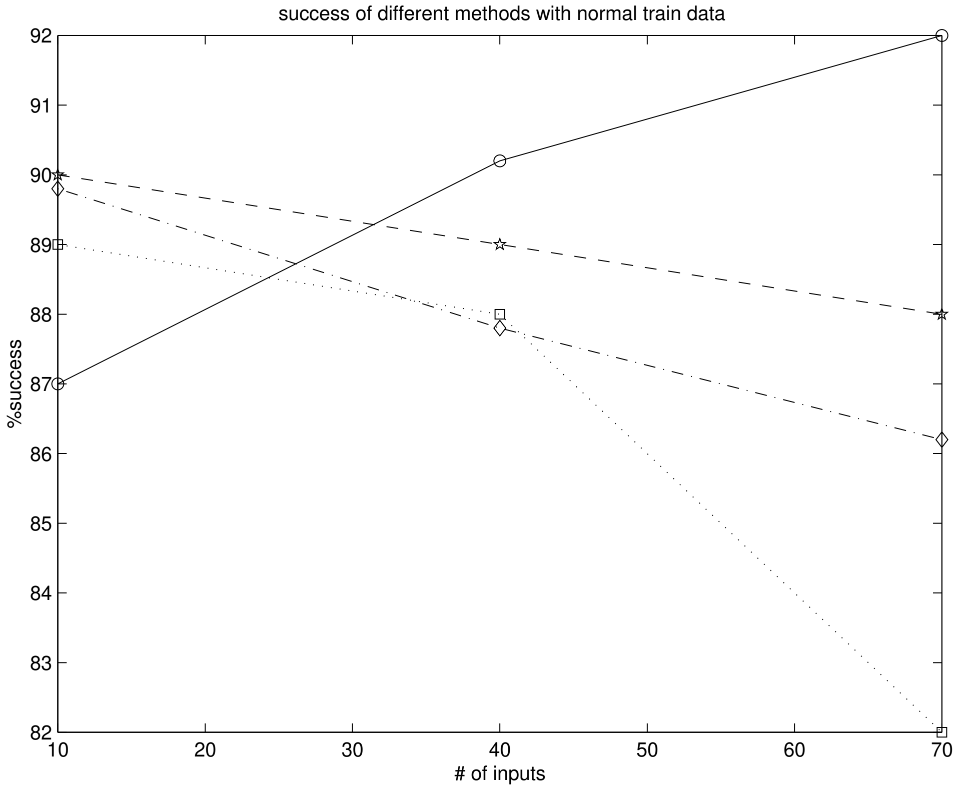

Figure 4.3. Several implementations of ANN - train normal data

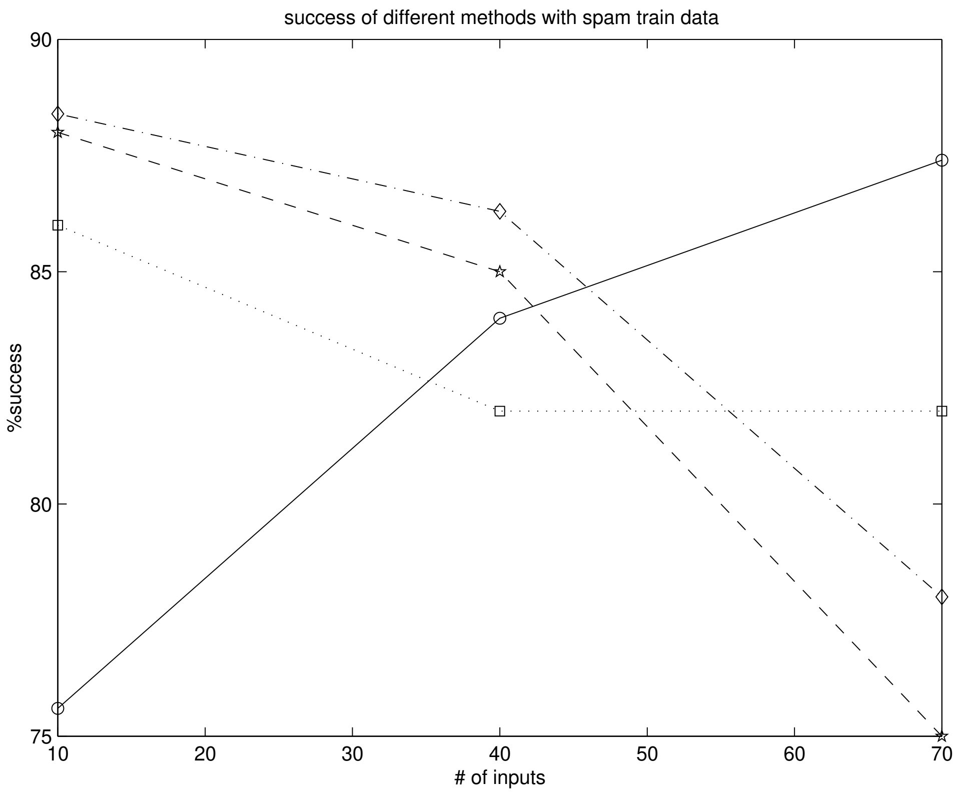

Figure 4.4. Several implementations of ANN - train spam data

- SLP with probabilistic model: circle -
- SLP with binary model: diamond -.
- MLP with 150 hidden layer nodes: square ..
- MLP with 250 hidden layer nodes: star –

From the experiments, we can conclude that optimal number of inputs is between 40 and 70 rather 10. With 10 inputs, success rates are really low when compared with 40 and 70 inputs. This means, 10 words are not enough to classify mails. On the other hand, we can conclude that success rates of 40 and 70 inputs are almost the same.

As can be seen from the figures, different algorithms may be more successful than others with different number of inputs or different data sets. For their best cases, we can conclude that, probabilistic model (81% total) achieves slightly better results than binary model (79% total). When compared with SLP algorithms, MLP algorithms achieve slightly better results (83% total) only with very large hidden layer node numbers causing big time complexities which yield inefficient results for such type of a filter application. We are planning to design a final product for end-users which means time complexity is an important measure for us. With these results, SLP with probabilistic model is selected to be most efficient among all.

4.2. Test 2

Although SLP with probabilistic model seemed to be the best of the implemented models, the success rates of the filter were not as high as expected, especially in spam test runs. To increase these rates, we have developed some more models to work together with SLP. One of these models is to use also the non Turkish words in the mails for our spam filter in addition to Turkish words. We have implemented it by using the word itself when an accepted Turkish root of the word can not been found in the program. With this improvement, the success rates of the filter have reached 85% in total, which was previously about 80%. Main reason for this improvement may be the fact that, some non-Turkish words (*http, www, com, etc.*) are frequently used in turkish spam mails while not so much used in normal mails. So they are said to be

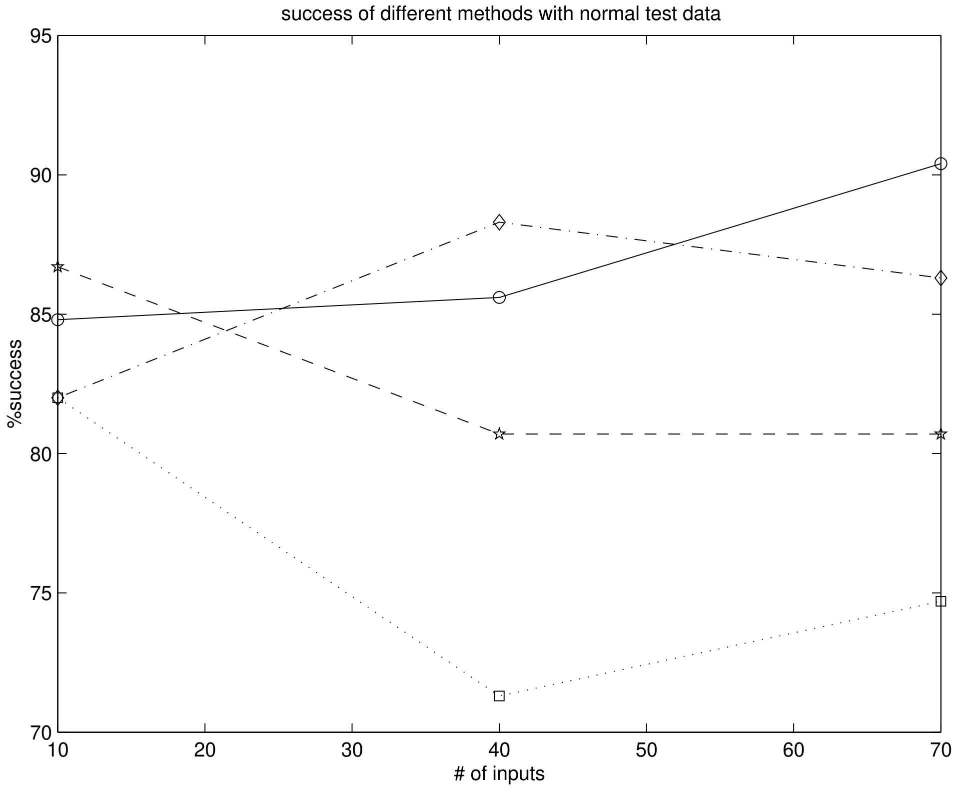

Figure 4.5. Several implementations of SLP - test normal data

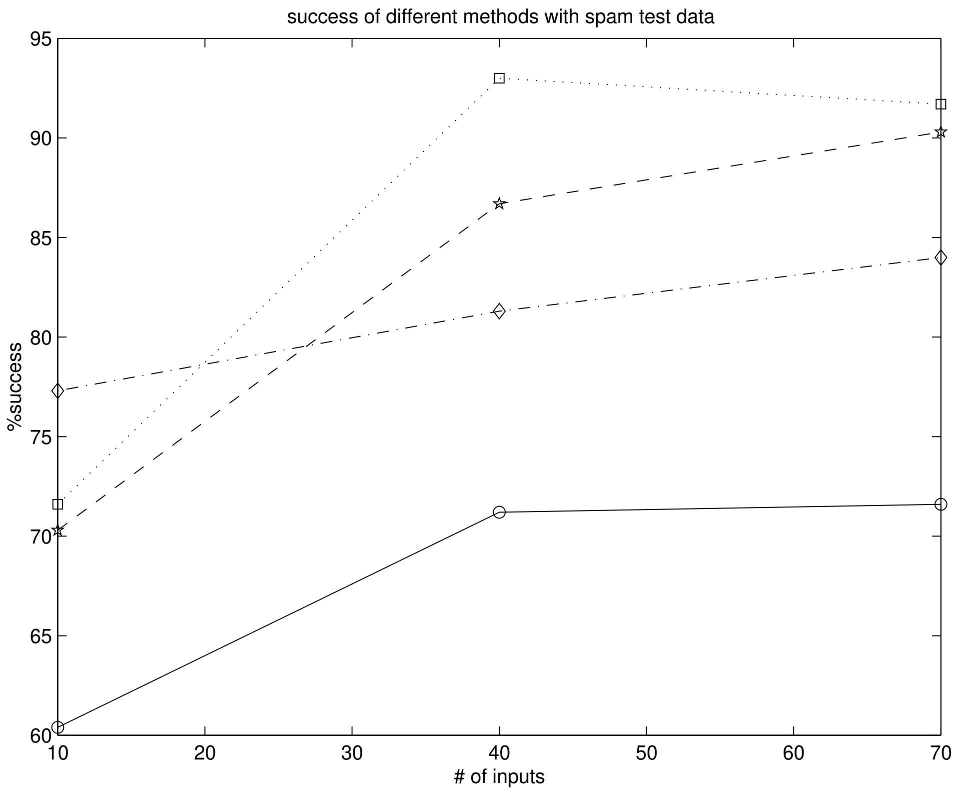

Figure 4.6. Several implementations of SLP - test spam data

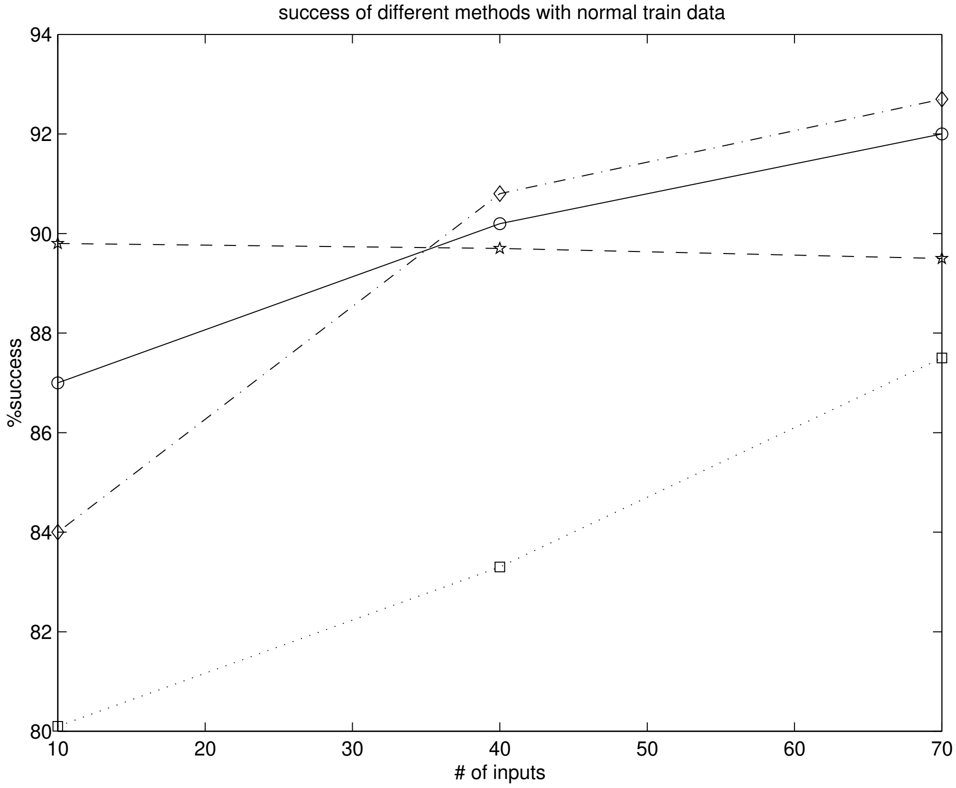

Figure 4.7. Several implementations of SLP - train normal data

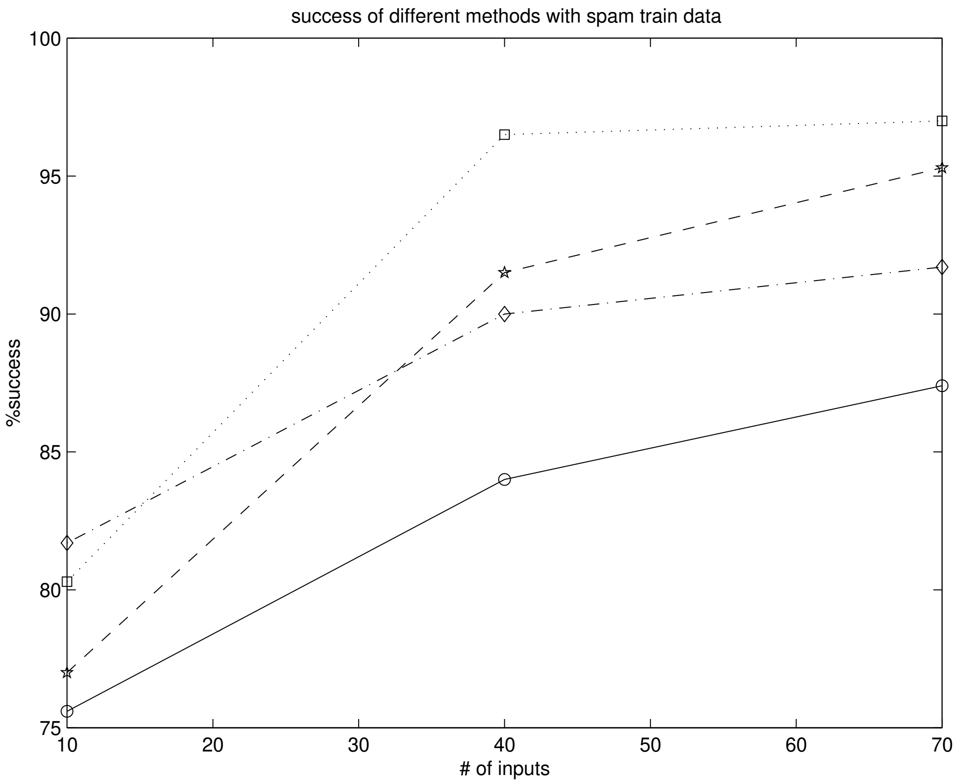

Figure 4.8. Several implementations of SLP - train spam data

better classifiers than most of the Turkish words in the mails.

As seen in Figure 4.5, Figure 4.6, Figure 4.7 and Figure 4.8 we also implemented two other models by improving the selection of the candidates for feature vector. In the figures, algorithms are represented as:

- with only accepted Turkish words: circle -
- with all words (Turkish+non-Turkish) accepted: diamond -.
- with all words(Turkish+non-Turkish) accepted + all words (spam+normal) as candidate for input : square ..
- with all words(Turkish+non-Turkish) accepted + all words(spam+normal) as candidate for input (words occurring multiplied by 3/2): star –

Previously, inputs to the ANN algorithm were most probably selected as the words occurring frequently in spam mails and much less in normal mails. Now we implemented the model so that, we have also selected words that occur frequently in normal mails and much less in spam mails [10]. With spam mails, this approach showed better results than the previous ones but disappointingly, a decline of success was observed with normal mails. With some improvements in this model (values of the probability of words occurring in the mails were weighted emprically by 3/2), an increase in success was observed and this model has showed the best performance (86% total) among all of the models.

4.3. Test 3

As it is seen in the Figure 4.9, Figure 4.10, Figure 4.11 and Figure 4.12, Bayes Approach seems more successful than ANN, especially with binary approach (88% total). In the figures, Bayes Classifier was implemented with

- Binary Model: circle -
- Probabilistic Model: diamond -.
- Advance Probabilistic Model : square ..

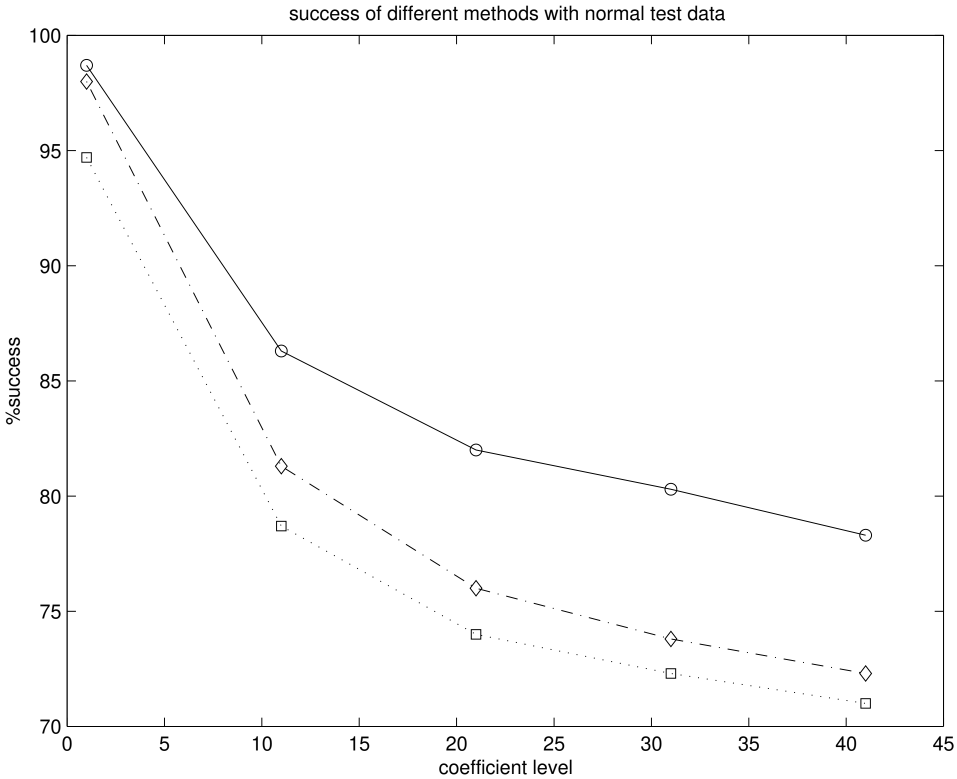

Figure 4.9. Several implementations of Bayes Approach - test normal data

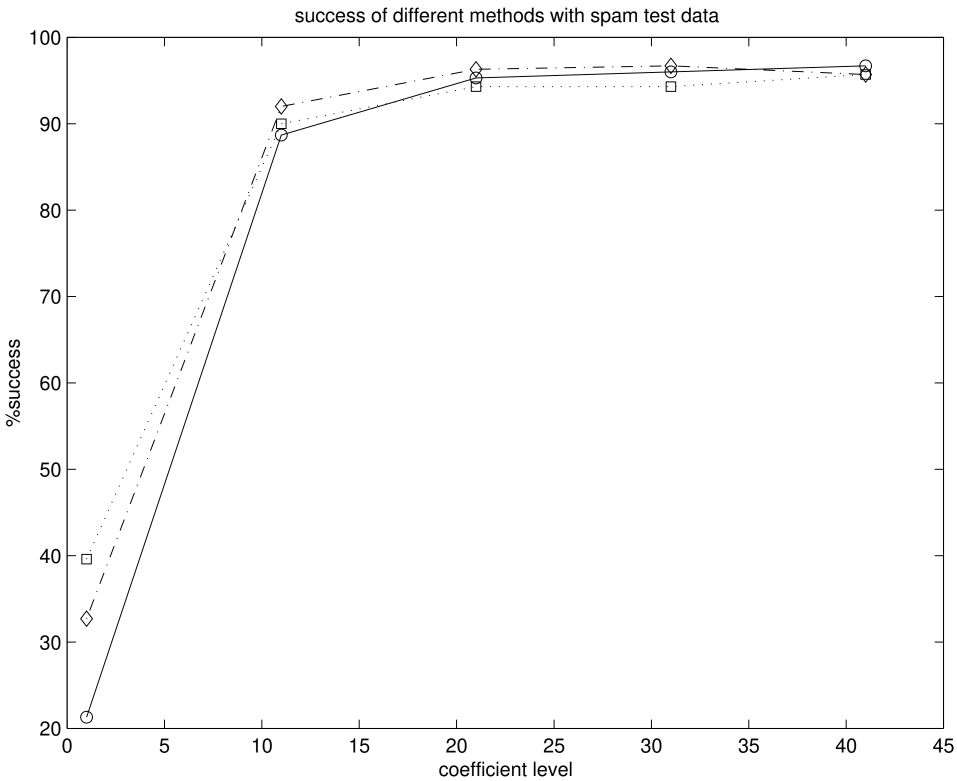

Figure 4.10. Several implementations of Bayes Approach - test spam data

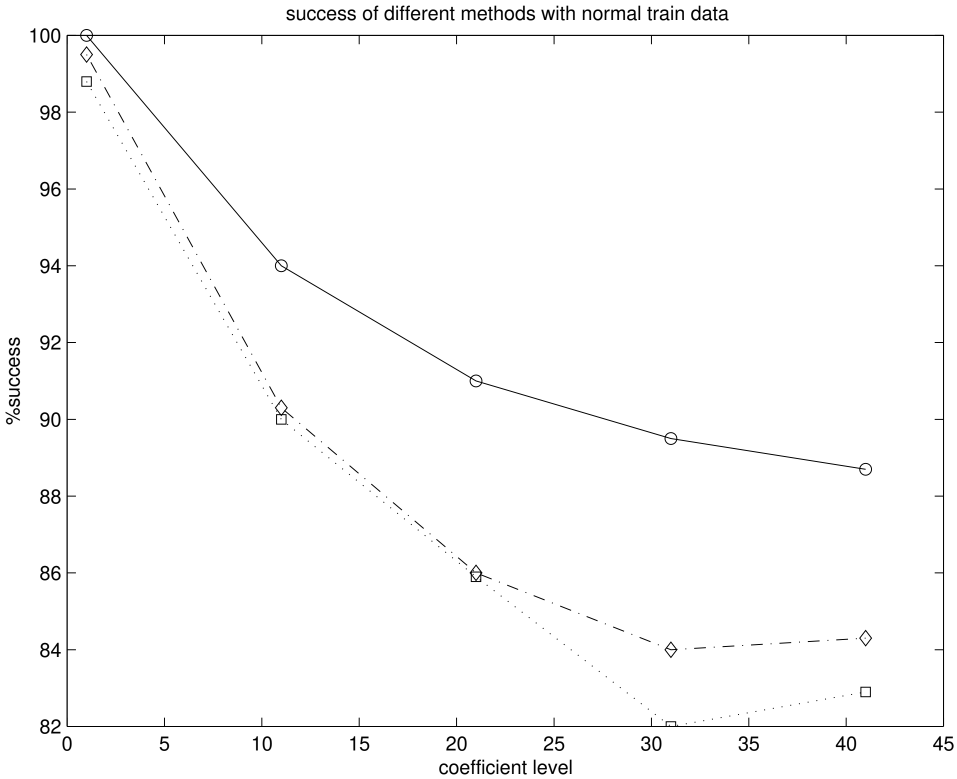

Figure 4.11. Several implementations of Bayes Approach - train normal data

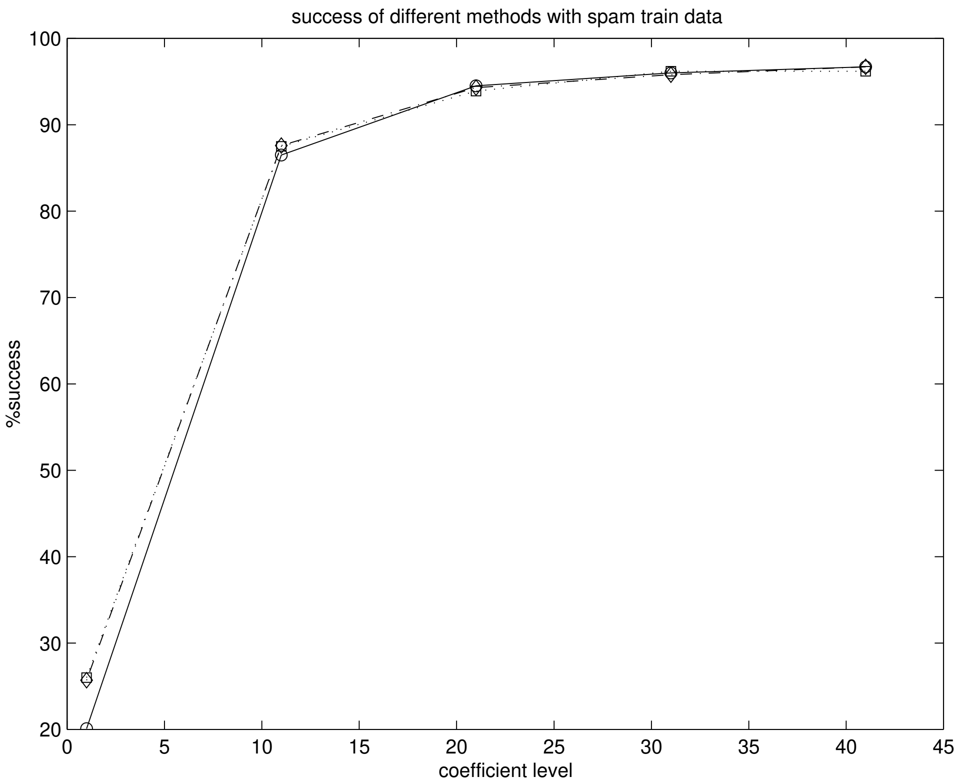

Figure 4.12. Several implementations of Bayes Approach - train spam data

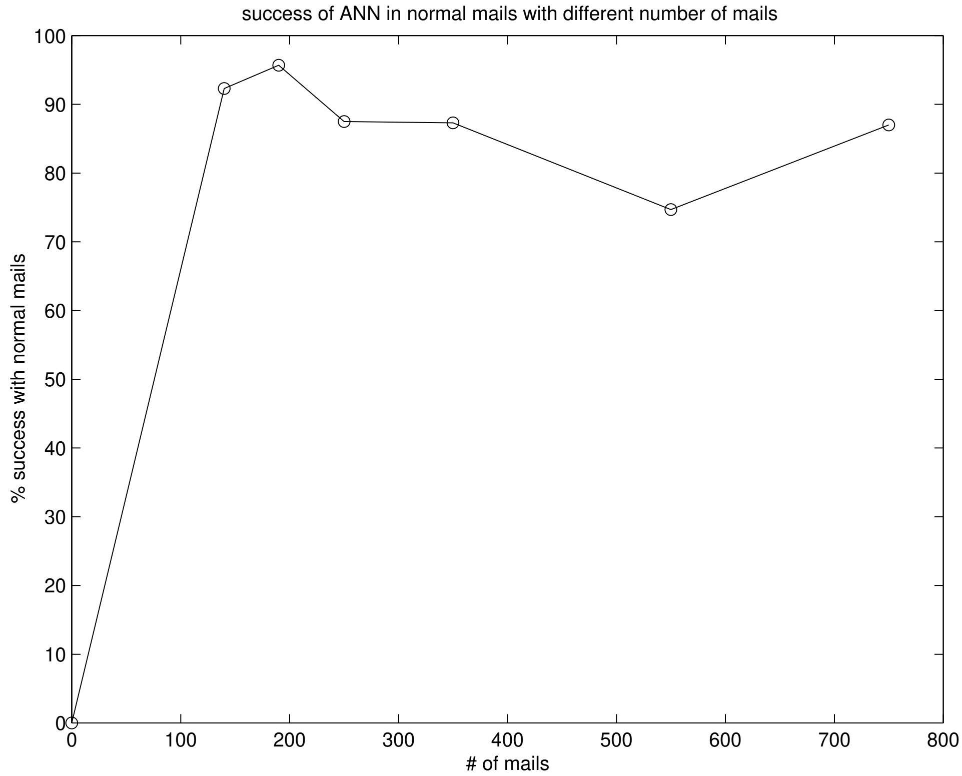

Figure 4.13. ANN implementation in different number of normal mails

Testes were performed with different coefficients. In mails, occurrence of an input word usually indicates a stronger idea for classifying mails than a non-occurrence of that word. Coefficient number indicates the level of this strength. During the tests, coefficient was taken between 1-14 in different runs. Disappointingly, a decline of success was observed with two other models(probabilistic and advance probabilistic model), which had in fact more complex structure than the Binary Model by taking also the length of the text and also the number of occurrences of the word into consideration.

4.4. Test 4

Motivated by an interest in understanding the learning rate of the algorithms, we have made a final test by using SLP algorithm of ANN with different number of mails. As seen in Figure 4.13-4.14, significant amounts of success was obtained with even a few number of mails (totally 90% success in test runs with only 190 mails). We understand that, much of the learning takes place with around 100-150 mails which means a rapid learning rate.

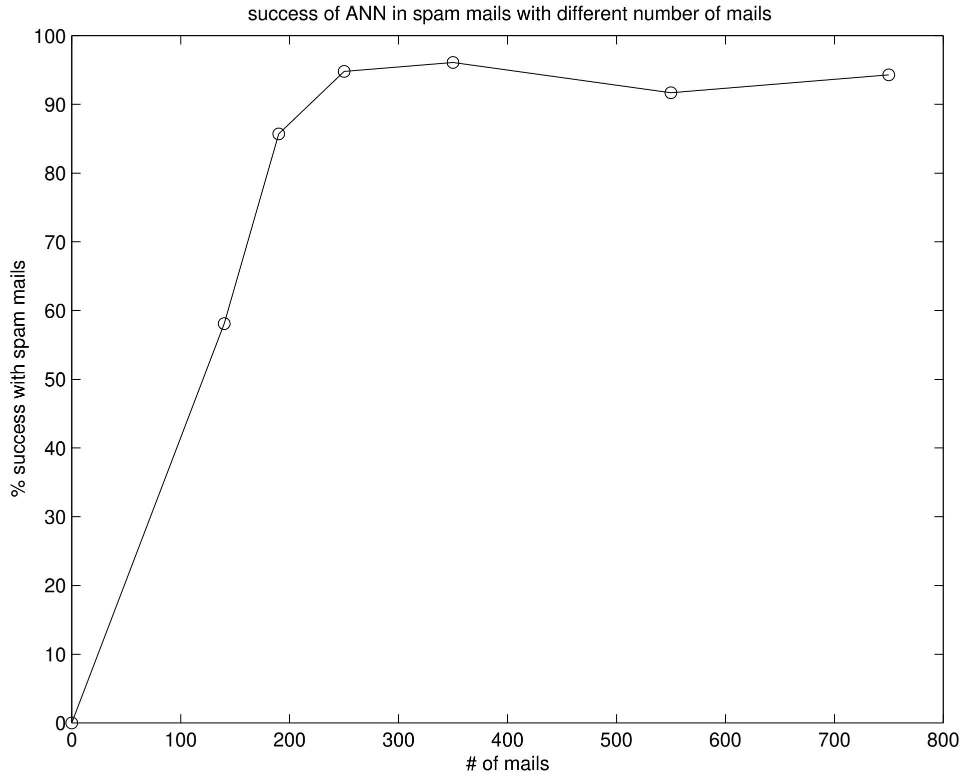

Figure 4.14. ANN implementation in different number of spam mails

4.5. Test 5

Although it is not in the main scope of our project, we have run the program also with English mails. The best results among all algorithms were achieved by probabilistic SLP of ANN with 70 inputs. 89.4% success was achieved in spam mails while 75.0% success was achieved in normal mails. The results were not as high as we expected. We may count the following reasons:

- Number of mails were small: 95 normal mails (45 of them were for test) and 150 spam mails (100 of them were for test) were used in the experiment. An experiment with more examples will probably show more success.
- English spam mails were prepared in a more professional way than the Turkish ones. Some mails did not have any text fields at all, there were only images in those files. Also, some mails were not using the correct form of the word (i.e. *g-a-i-n m-o-n-e-y* instead of *gain money*) which made it difficult to classify mails.

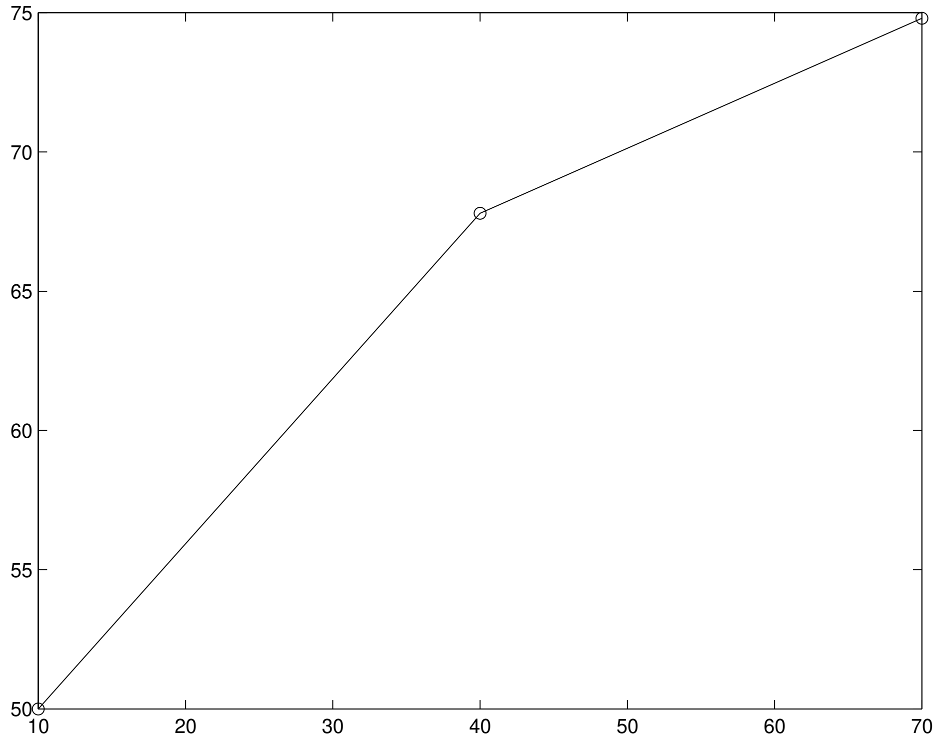

Figure 4.15. Results of English normal mails with several input numbers

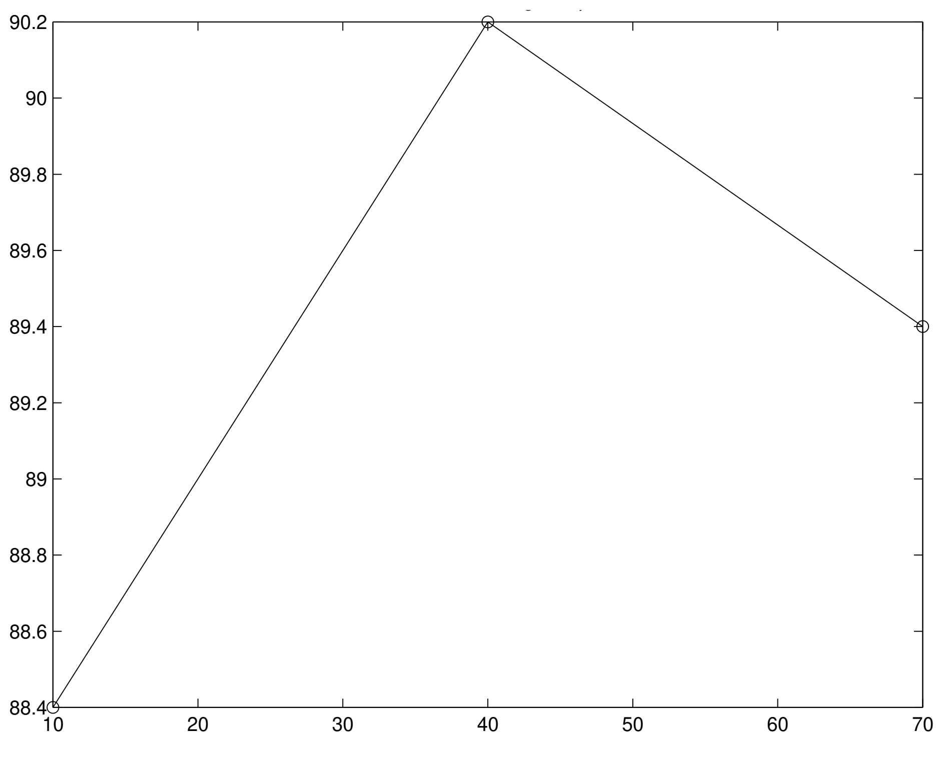

Figure 4.16. Results of English spam mails with several input numbers

4.6. Classifiers

Table 4.1. Best Classifier Words for Mail Categorization

Rank	Word	Score	Rank	Word	Score
1	tl	1.72	2	kdv	1.43
3	sen	1.33	4	http	1.31
5	reklam	1.25	6	site	1.16
7	edu	1.09	8	yap	1.09
9	en	1.09	10	ve	1.08
11	firma	1.07	12	euro	1.06
13	boun	1.04	14	tanıtım	1.04
15	usd	1.02	16	fiyat	1.02
17	tek	1.00	18	ürün	0.97
19	adet	0.95	20	kategori	0.95
21	tel	0.94	22	internet	0.93
23	rehber	0.92	24	program	0.92
25	gönder	0.91	26	cem	0.91
27	üretim	0.90	28	mb	0.90
29	ilet	0.90	30	neyse	0.89
31	tüm	0.88	32	iyi	0.88
33	halk	0.88	34	ama	0.87
35	web	0.86	36	türkiye	0.86
37	filan	0.85	38	bil	0.85
39	tr	0.84	40	ücret	0.84

The top 40 classifier words that form the feature vector for the learning algorithms are listed in Table 4.1. The scores are calculated according to the relativity of the word with the normal and spam mails. The words with higher scores are said to be better classifiers, occurring more frequently in one class and less in the other class.

From the table, we may conclude that most of the these words economic oriented (*tl, kdv, reklam, adet, euro, etc.*). Most of the spam mails (at least, the ones in our corpus) offer us their products so it's not surprising that classifiers for spam mails are mainly words about economics.

Also, although our study is about filtering Turkish spam mails some high-valued classifiers are non-Turkish words (*edu, http, www, etc.*) mostly occurring in frequently spam mails. These are internet oriented words and spam mails usually have a mail or internet address so these words are also not surprising to be the classifier words for mail categorization.

Also there is the fact that most of the classifiers are the words occurring in spam mails and less in normal mails. There are few examples for the opposite situation. This situation will be discussed in the next section with more details.

4.7. An Overview of Success Rates

Table 4.2 shows the success rates of each algorithm. From the experiments, we conclude that non-Turkish words (*http, www, com, etc.*) are surprisingly better classifiers than standard Turkish words by occurring frequently in spam mails and not so much in normal mails. This means that, in the future studies, although a suitable root form is not found in the MM for the word, it will be admitted to the LM in its base form.

Also, when we take the inputs only from spam mails, we see that success rates are almost the same (85%-86%) as the rates in which we use both mail (spam and normal) words. But we prefer mixing those mail words and choose inputs from this mixture although we have understood that most of the classifier words are spam words (*tanıtım, gazete, basın, reklam, tarih, destek, tel, etc.*) that do not occur too much in normal mails. There are very few normal mail words in the classifier words and interestingly these are private names, most probably my name or names of my friends. This is said to be a good property for a user-specific filter, because a mail having the name of your friends is most probably not spam.

Table 4.2. Time complexities of different experiments

Algorithm	Success Rate(%)
SLP(Binary Model+ non-Turkish words+ spam words as candidate)	79
SLP(Probabilistic Model+ non-Turkish words+ spam words as candidate)	81
MLP(150 hidden layer nodes+ Probabilistic Model+ non-Turkish words+ spam words as candidate)	80
MLP(250 hidden layer nodes+ Probabilistic Model+ non-Turkish words+ spam words as candidate)	83
SLP(Probabilistic Model + all words(Turkish+non-Turkish)+ spam words as candidate)	85
SLP(Probabilistic Model + all words(Turkish+non-Turkish)+ all words as candidate(spam+normal))	86
All Bayes implementations	88-86-84

4.8. An Overview of Time Complexities

Table 4.3 shows the average time (in seconds) of each algorithm. It is clear that SLP model without hidden layer nodes and without all words included is the fastest model with 1.5 and four seconds each. However, as described above, success rates of SLP are far below with respect to SLP with all words. Note that, these experiments were run with over 500 e-mails. For a user-specific application, number of mails may decline. Also note that, even with the algorithm of SLP with all words as candidates, time is below 1 minute, which means it is a reasonable time even for a user application.

Table 4.3. Time complexities of different experiments

Algorithm	Time(second)
SLP(Binary Model+ non-Turkish words+ spam words as candidate)	1.5
SLP(Probabilistic Model+ non-Turkish words+ spam words as candidate)	4
MLP(150 hidden layer nodes+ Probabilistic Model+ non-Turkish words+ spam words as candidate)	19
MLP(250 hidden layer nodes+ Probabilistic Model+ non-Turkish words+ spam words as candidate)	>100
SLP(Probabilistic Model + all words(Turkish+non-Turkish)+ spam words as candidate)	45
SLP(Probabilistic Model + all words(Turkish+non-Turkish)+ all words as candidate(spam+normal))	46
All Bayes implementations	46

5. CONCLUSION

In this research, we have designed and implemented a Turkish anti-spam filter. As far as studies on Turkish language concerned, our filter is the first to filter Turkish spam mails.

We have dealt with three important and different phases in this project. In the first phase, a Turkish morphological program was prepared based on the previous study by Güngör which has extracted the root form of Turkish words in approximately 750 mails with over 90% success and speed of approximately 3000 words/min. In the second phase, these root forms were used in the LM for classification of mails as spam or normal. ANN and Bayes Filter were employed for this phase which yielded over 90% success in filtering spam mail. During the experiments in the learning phase, we have understood that non-Turkish words (*http, www, com, etc.*) were surprisingly better classifiers than standard Turkish words by occurring frequently in spam mails and seldom in normal mails. In the third phase, we have designed a final product which works in parallel with Microsoft Outlook and enables user to not only filter his spam mails, but also make his filter adaptable to new mails by continuing the training operation on these mails. Thus, our filter has become a user-specific Turkish anti-spam filter.

We have also worked with English data which have not yielded very successful results mainly because of the small number of English training mails. Also English spam mails were said to be designed in a more professional way with more images and less words and also some ambiguous words were used instead of correct ones in the spam mails.

As a future study, we plan to adapt our filter program to English mails by increasing the training size and also improving the algorithm to overcome those professional spam mails. This improvement in the algorithm may be an additional study on the attachments (pictures, text files, etc.) of the mails or simply addition of an extra property of ANN or Bayes in the learning phase.

Also, there is the fact that, anti-spam filter is said to be a subproblem of text categorization, in which predefined categories are assigned to texts based on their content. We can broaden our research and make a study on automatic classification of Turkish texts, which is a task of increasing importance.

REFERENCES

1. "http://www.turk.internet.com", 2003.

2. Apte C., F. Damerau, S.M. Weiss, "An Evaluation of Naive Bayesian Networks" , *Proceedings of the Machine Learning in the New Information Age*, 2000.

3. Androutsopoulos I., J. Koutsias, "Automated Learning of Decision Rules for Text Categorization" , *ACM Transactions on Information Systems*, 12(3):233-251,1994.

4. Cohen W., "Learning Rules That Classify E-mail" , *Proceedings of the AAAI Spring Symposium on Machine Learning in Information Access*Stanford, California, 1996.

5. Dumais S. T., J. Platt, D. Hecherman, M. Sahami, "Inductive learning algorithms and representations for text categorization", *Proc. 7th International Conference on Information and Knowledge Management CIKM*, Germany, 1998, pp. 148-155.

6. Koller D., M. Sahami, "Toward Optimal Feature Selection", *Proceedings of the 13th Interna-tional Conference on Machine Learning (ML)*, Bari,Italy, July 1996, pp. 284-292.

7. Lewis D., "Feature Selection and Feature Extraction for Text Categorization" , *Proceedings of the DARPA Workshop on Speech and Natural Language*, pp. 212-217, Harriman, New York, 1992.

8. Dagan I., Y. Karov, D. Roth, "Mistake-Driven Learning in Text Categorization" , *Proceedings of the Conference on Emprical Methods in Natural Language Processing*,pp. 55-63, Providence, Rhode Island, 1997.

9. Güngör T., "Computer Processing of Turkish: Morphological and Lexical Investigation" , PhD Thesis, Boğaziçi University, 1995.

10. Çetinoğlu Ö, "A Prolog Based Natural Language Processing Infrastructure for Turkish" , MS Thesis, Boğaziçi University, 2001.

11. Bishop, C., *Neural Networks for Pattern Recognition*, Oxford Unv. Press, 1995.

12. "http://www.gc.ssr.upm.es/inves/neural/ann1/anntutor.htm", 2003.

13. "http://www.pmsi.fr/neurin2a.ht", 2003.

14. Mitchell, T.M., *Machine Learning*, McGraw-Hill, 1997.

15. Gama J., "A Linear-Bayes Classifier" *, IBERAMIA-SBIA 2000, LNAI 1952*, pp. 269-279, 2000.

16. Duda R. O, P. E. Hart, D. G. Stork, *Pattern Classification*, John Wiley & Sons,Inc, 2001.

17. Koffman, E. B., *Pascal*, Addison-Wesley, 1992.

18. Kernighan B. W., D. M. Ritchie, *The C Programming Language*, Prentice Hall, 1988.

19. Akin H. L., S. Kuru, T. Güngör, İ. Hamzaoğlu, D. Arbatlı"A Spelling Checker and Corrector for Turkish" *, İkinci Türk Yapay Zeka ve Yapay Sinir AğlarıSempozyumu*,pp. 113-120,1993.

20. Uysal M., *Microsoft VB ile Yazılim Geliştirme*, Beta, 1997.

21. Çubukçu F., *Visual Basic Temel Kullanım Kılavuzu*, Alfa, 1998.

22. Rizzo T., *Programming Microsoft Outlook and Microsoft Exchange*, Microsoft Press, 2000.

23. "http://www.paulgraham.com/spam.html", 2003.

24. "http://www.spam.org.tr/", 2003.