On Email Spam Filtering Using Support Vector Machine

Ola Amayri

A Thesis

in

The Concordia Institute

for

Information Systems Engineering

Presented in Partial Fulfillment of the Requirements
for the Degree of Master of Applied Science (Information Systems Security) at
Concordia University
Montréal, Québec, Canada

January 2009

Your file Votre référence
ISBN: 978-0-494-63317-5
Our file Notre référence
ISBN: 978-0-494-63317-5

ABSTRACT

On Email Spam Filtering Using Support Vector Machine

Ola Amayri

Electronic mail is a major revolution taking place over traditional communication systems due to its convenient, economical, fast, and easy to use nature. A major bottleneck in electronic communications is the enormous dissemination of unwanted, harmful emails known as "spam emails". A major concern is the developing of suitable filters that can adequately capture those emails and achieve high performance rate. Machine learning (ML) researchers have developed many approaches in order to tackle this problem. Within the context of machine learning, support vector machines (SVM) have made a large contribution to the development of spam email filtering. Based on SVM, different schemes have been proposed through text classification approaches (TC). A crucial problem when using SVM is the choice of kernels as they directly affect the separation of emails in the feature space. We investigate the use of several distance-based kernels to specify spam filtering behaviors using SVM. However, most of used kernels concern continuous data, and neglect the structure of the text. In contrast to classical blind kernels, we propose the use of various string kernels for spam filtering. We show how effectively string kernels suit spam filtering problem. On the other hand, data preprocessing is a vital part of text classification where the objective is to generate feature vectors usable by SVM kernels. We detail a feature mapping variant in TC that yields improved performance for the standard SVM in filtering task. Furthermore, we propose an online active framework for spam filtering. We present empirical results from an extensive study of online, transductive, and online active methods for classifying spam emails in real time. We show that active online method using string kernels achieves higher precision and recall rates.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my advisor, Dr. Bouguila, for his continuous support and encouragement throughout my graduate studies. Because of his input, advice, and challenge, I have matured as a researcher and as a graduate student. I am very grateful for my sisters, brothers, and grandparents for the support and happiness they always provide me with. Finally, my sincere thanks and deepest appreciation go out to my parents, Suhair and Moufied for their affection, love, support, encouragement, and prayers to succeed in my missions.

Table of Contents

List of Figures vii

List of Tables viii

List of Symbols x

1 Introduction 1

1.1 General Overview of Spam Phenomenon 1

1.2 Counter-measures For Blocking Spam Emails 3

1.2.1 White List / Black List 3

1.2.2 Challenge-Response and Micropayment or Postage Systems 4

1.2.3 Matching Systems 5

1.2.4 Machine Learning System 5

1.3 Support Vector Machine 6

1.4 Thesis Overview 10

2 Spam Filtering using Support Vector Machine 12

2.1 Feature Mapping 13

2.1.1 Feature Extraction 13

2.1.2 Term Weighting 15

2.2 Kernels 17

2.2.1 Distance Based Kernels 17

2.2.2 String Kernels 18

2.3 Experimental Results 21

3 Spam Filtering using Online Active Support Vector Machine 32

3.1 Online SVM 32

3.2 Transductive Support Vector Machine (TSVM) 33
3.3 Active Learning 35
3.4 Experimental Results 37

4 Conclusions and Future work **46**

List of References **48**

List of Figures

2.1 The overall Performance of spam classification, where different feature mapping are applied 27

3.1 Classification Error on the trec05p-1 for RBF_{χ^2}.logTF and SSk kernels by varying the number of unlabeled emails in training for TSVM 40

3.2 Classification Error on the trec05p-1 for RBF_{χ^2}.logTF and SSk kernels by varying the number of labeled emails in training for TSVM 43

List of Tables

2.1 Examples of classic distance based kernels 17
2.2 Case of experiments 23
2.3 The performance of SVM spam filtering on trec05-1 normalized using L_1-norm, and stop words have been removed 26
2.4 The AUC value of trec05-1 normalized using L_1-norm, and stop words have been removed 26
2.5 The performance of SVM spam filtering on trec05-1 normalized using L_1-norm, and without removing stop words 28
2.6 The AUC value of trec05-1 normalized using L_1-norm, and without removing stop words 28
2.7 The performance of SVM spam filtering on trec05-1 normalized using L_2-norm, and stop words have been removed 29
2.8 The AUC value of SVM spam filtering on trec05-1 normalized using L_2-norm, and stop words have been removed 29
2.9 The performance of SVM spam filtering on trec05-1 normalized using L_2-norm, and without removing stop words 30
2.10 The AUC value of SVM spam filtering on trec05-1 normalized using L_2-norm, and without removing stop words 30
2.11 The performance of SVM spam filtering on trec05-1 using string kernels 31
2.12 Training time for different combinations of frequency and distance kernels . . . 31
2.13 Training time for string kernels 31
2.14 The AUC value of SVM spam filtering on trec05-1, using string kernels. 31

3.1 The performance of Online SVM spam filtering on trec05-1 39
3.2 The performance of Online SVM spam filtering on trec05-1 using string kernels 39
3.3 The performance of TSVM spam filtering on trec05-1 41

3.4 The performance of TSVM spam filtering on trec05-1, where unlabeled training emails are from trec06 data set 42
3.5 The performance of TSVM spam filtering on trec05-1 using string kernels . . . 42
3.6 The performance of TSVM spam filtering on trec05-1 using string kernels, where unlabeled training emails are from trec06 data set 43
3.7 The performance of Online Active SVM spam filtering on trec05-1 44
3.8 The performance of Online Active SVM spam filtering on trec05-1 using string kernels 44
3.9 Training time for different combinations of frequency and distance kernels . . . 45
3.10 Training time for string kernels 45

List of Symbols

SVMs	Support Vector Machines
NNTP	Network News Transfer Protocol
SMTP	Simple Mail Transfer Protocol
ISPs	Internet Service providers
CPU	Central Processing Unit
IP	Internet Protocol
qp	quadratic programming
KKT	Karush-Kuhn-Tucker
TC	Text Categorization
BoW	Bag-of-Word
TF	Raw Frequency
ITF	Inverse Raw Frequency
IDF	Inverse Document Frequency
WD	Weighted Degree Kernel
WDs	Weighted Degree Kernel with Shift
SSK	String subsequence kernel
SV	Support Vector
TSVM	Transductive Support Vector Machine
ROC	Receiver Operating Characteristic
AUC	Area Under ROC Curve

CHAPTER 1

Introduction

This chapter presents a brief overview of spam emails phenomenon, along with contributing factors, damage impact, and briefly discusses possible countermeasures to mitigate spam problem. Moreover, we explore how to use Support Vector Machines (SVMs) to model anti-spam filters, and introduce deployed kernels properties.

1.1 General Overview of Spam Phenomenon

Years ago, much attention has been paid for automation services, business, and communication. Internet affords evolutionary automated communication that greatly proved its efficiency such as electronic mail. Electronic mail has gained immense usage in everyday communication for different purposes, due to its convenient, economical, fast, and easy to use nature over traditional methods. Beyond the rapid proliferation of legitimate emails lies adaptive proliferation of unwanted emails that take the advantage of the internet, known as spam emails. Generally, spam is defined as "unsolicited bulk email" [1].

By definition, spam emails can be recognized either by content or delivery manner. Spam emails can be sent, for instance, for commercial purposes, where some companies take the advantage of emails to widely advertise their own products and services. Fraudulent spam emails "phishing" [2] were employed to serve online frauds. In this case, spammers impersonate trusted authorities, such as server's administrator at schools, banks and ask users for sensitive information such as passwords, credit cards numbers, etc. Other spam emails contain a piece of malicious code that might be harmful and might cause a damage to the end user machines [3]. Furthermore, some of the investigations that dealt with the content of spam emails has applied "genres" concept in analysis of spam, where spam emails have the same structure

as legitimate emails such as letters, memo (for instance, see [5]). Alternatively, spam emails were recognized according to the volume of dissemination and permissible delivery "Spam is about consent, not content" [1].

Various means were used to propagate spam emails worldwide. Spam propagation undergoes three main stages: harvesting, validation, and spamming. Furthermore, spammers have benefited from the Internet medium nature, where some mail providers on the Internet such as yahoo.com became an origin for spam. Since creating accounts is priceless and easy, spammers developed automated tool to create and then send spam emails. Generally, harvesting is collecting victims email addresses using number of cost-effective and easy techniques. Involved techniques range from buying, where they buy a list of valid email addresses which were harvested from web sites. Unfortunately, although list-sale is prohibited by law in countries, it is allowed at others. Network News Transfer Protocol (NNTP) was lunched years before world wide web with newsgroup and forums that have rich information including email addresses of writer of the articles, visitors, for instance, and they are accessible to everyone. The fact that make it a public and easy source for collecting email addresses. Spammers use more techniques for harvesting such as direct access to the servers, Simple Mail Transfer Protocol (SMTP) brute force attack, and viruses (for instance, see [31]). Next, in validation stage spammer tries to verify if the recipient of their email read the email or not. Spammers can verify using return recipient approval, SMTP verification, active user intervention, and virus verification [6]. Finally, spamming is dissemination of spam messages using harvested email addresses. Pursuing this further, spammer benefits from some server configuration standards, such as SMTP that doesn't verify the source of the message. Consequently, spammers can forge the source of the message and impersonate other users in the Internet by hijacking their unsecured network servers to send spam emails which is known as "Open relays". Moreover, "Open proxy" is designed for web requests and can be accessible from any Internet user. It is helping the user to forward Internet service request by passing the firewalls, which might block their requests. Identifying the source of the request using open proxy is impossible. Consequently, spam filtering techniques fail to detect spam origin, and block it. Other techniques were used for spamming such as end-to-end delivery, zombie systems, etc [31].

The problem raised when the phenomenon considers plaguy effects. Although, some Internet service providers (ISPs) remove a lot of known spams before deposit them in user email

accounts, a lot of spam emails bypass ISPs as well, causing ISP's CPU time consuming. Preciously, the lower cost of disturbing spam emails might cause financial disasters. Even more, users complain the privacy of their email, claim the offensive content of spam email, wasting time in filtering, and sorting their emails, decreasing business productivity, and wasting the connection bandwidth. Receiving few spam emails a day won't be that matter. In meanwhile, the overwhelmed users of a huge quantity of spam emails, might spend more money to make their inbox larger in order to save losing legitimate emails (for instance, see [7]). Suffering caused from spam emails is far worse, where some of spam messages might crash the email server temporary [8].

1.2 Counter-measures For Blocking Spam Emails

As the threat is widely spread, variety of techniques have been developed to mitigate sufferings of spam emails [6]. Involved techniques were implemented on server side, where the server block the spam message before it starts its SMTP transaction. Others were implemented on client side, where the decision is left to the end user to define the spam (i.e. what is spam for some users, might not for others). Some of them were adopted in industry and have reported good result in decreasing the misclassification of legitimate emails. However, there is no universal solution to this problem regarding the dynamic nature of the spam problem which is known as "Arm force" [31]. In what follows we introduce some of the techniques which have been proposed to mitigate the damage impact [22].

1.2.1 White List / Black List

White list (Safe list) is a list constructed by individual users, and contains the addresses of their friends or more general the addresses they recognized as legitimate. This technique has been employed besides other techniques such as machine learning techniques and matching system techniques which we shall discuss next. Safe list is a good technique so far for people who do not expect to receive emails from people they do not know, where only the people in the safe list can deposit their emails in user inbox, all others not mentioned are marked as spam. On the other hand, for commercial, research purposes, for instance, people in those fields expect to receive a legitimate email from senders they do not know as a usual routine

of their work. Furthermore, they do not want to miss any of these emails so far, nor wasting their time scanning the spam folder to check if they have any legitimate email resides there [6]. However, spoofing is a trick that has been used by the spammers to forge the safe lists addresses, where they can impersonate anyone else in the internet as the email protocol allows this. Consequently, you can receive a spam email from your self or any of addresses in your safe lists.

Black list (Block list) has constructed on a router level with IP "Internet Protocol" address which were considered as an end-point of known spam email. Clearly, then no IP address mentioned in this list can start SMTP transaction, which would save the bandwidth from spam traffic [31]. However, sometimes black lists contain non-spammers (victims). Moreover, some of the IP address are a source for either spam emails or good emails at the same time, which prevent sending the legitimate emails. Furthermore, it is not up to date for new spam techniques.

1.2.2 Challenge-Response and Micropayment or Postage Systems

Generally, if the end users system doesn't recognize the received email as legitimate, it will ask the sender to enter a challenge response in a form of a picture with few letters or numbers, this is called Human challenges. However, spammers easily overcome this problem by using tools to solve the issue. Another type is micropayment systems [7]. This technique attempts to force the spammers pay some money when they send spam emails, where the user asks the spammer to deposit a small amount of money in his account. If the end user takes the grant, he would then open the email, otherwise he should return it back. Some receivers sometimes take the money even if the email is not spam [29]. Computational challenge is an automated techniques that ask the sender for a hash value of the message (or the header only) if the sender gives the correct answer, his message is removed from spam folder to the user inbox. If the challenge is more complex, the spammer spends more time to solve it which results in less number of spam emails sent [25] [26] [27].

1.2.3 Matching Systems

In this technique the email system attempts to find a match between the new incoming spam message and the old messages list marked by the end user as a spam, if there is a match it is reported as spam, otherwise as legitimate email. Spammers trick these systems by randomizing their message (adding a random numbers or letters at the end of each subject) [31]. Rule-based systems were constructed to overcome these randomization, where there is a team of technicians who attempt to find the similarities between those messages and writing rules to capture future messages [6]. Fuzzy-hashing is a matching technique that searches for the similarity between messages and compute a large hash number for each spam message, if the next incoming message has the same hash number then it is spam.

1.2.4 Machine Learning System

One of the solutions is an automated email filtering. Variety of feasible contributions in the case of machine learning have addressed the problem of separating spam emails from legitimate emails. Traditionally, many researchers have illustrated spam filtering problem as a case of text categorization after it reported a good result in machine learning [10]. However, the researchers have realized later the nature and structure of emails are more than text such as images, links, etc. Supervised machine learning starts by obtaining the training data, which is manually prepared by individuals. The training data have two classes, one is the legitimate emails, another is the spam emails. Next, each message is converted into features (words); that is, words, time, images, for instance in the message. Then, build the classifier that would predict the nature of future incoming message.

The best classifier is the one that reduces the misclassification rate. Many machine learning techniques have been employed in the sake of spam filtering such as Boosting Trees [15], k-nearest neighbour classifier [23], Rocchio algorithm [16], Naive Bayesian classifier [17], and Ripper [9]. Furthermore, these algorithms filter the email by analyzing the header only, or the body only, or the whole message. Support Vector Machine is one of the most used techniques as the base classifier to overcome the spam problem [18]. Some studies developed spam filtering in a batch mode [11]. Lately, studies have focused on online mode [12] [13] which prove its effectiveness in real time. A comparison between those two modes can be found in [14]. In the

following section we briefly introduce an overview of SVMs.

1.3 Support Vector Machine

In this section, we describe briefly the mathematical derivation of Support Vector Machines (SVMs). Support Vector Machines are known to give accurate discrimination in high feature space [9], and it received a great attention in many applications such as text classification [10]. SVMs have out-performed other learning algorithms with good generalization, global solution, number of tuning parameters, and its solid theoretical background. The core concept of SVMs is to discriminate two or more classes with a hyperplane maximizing the margin by solving quadratic programming (qp) problem with linear equality and inequality constraints.

Suppose that the set of random independent identically distributed training vectors drawn according to $P(\vec{x}, y) = P(\vec{x})P(\vec{x}|y)$ belonging to two separate classes, given by $\{(\vec{x}_1, y_1)$, ..., $(\vec{x}_l, y_l)\}$, $\vec{x}_i \in \mathrm{R}^n, y_i \in \{-1, 1\}$, where $\vec{x}_i$ is n-dimensional training vector, y indicating the class in which $\vec{x}_i$ belongs. Suppose we have a hyperplane which separates the two different classes, given by

$$\vec{w}.\vec{x}_i + b \geq 1, \forall \vec{x}_i \in Class1 \tag{1}$$

$$\vec{w}.\vec{x}_i + b \leq -1, \forall \vec{x}_i \in Class2 \tag{2}$$

The points $\vec{x}$ which lie on the hyperplane satisfy $\vec{w}.\vec{x}$+ b=0, where $\vec{w}$ is the normal of the hyperplane. The hypothesis space in this case is the set of functions, given by

$$f_{\vec{w},b} = sign(\vec{w}.\vec{x} + b) \tag{3}$$

Since SVMs provide unique and global solution, looking at Eq. 3 we can observe a redundancy, where if the parameter $\vec{w}$ and b are scaled by the same quantity, the decision surface is unchanged. Consider the Canonical hyperplane (The set of hyperplanes that satisfy Eq. 4), where the parameter $\vec{w}$, b are constrained by

$$y_i[\vec{w}.\vec{x}_i + b] \geq 1, i = 1, ..., l. \tag{4}$$

The optimal hyperplane is given by maximizing the margin (i.e. perpendicular distance from the separating hyperplane to a hyperplane through Support Vector (SV)) subject to the

constraints in Eq. 4. In linearly separable we need to find Canonical Hyperplanes that correctly classify data with minimum norm, or equivalently minimum $||\overrightarrow{w}||^2$. This is formulated as

$$\text{Minimize } \Phi(\overrightarrow{w}) = \tfrac{1}{2}||\overrightarrow{w}||^2$$

Subject to:

$$y_i(\overrightarrow{w}.\overrightarrow{x}_i + b) \geq 1 \tag{5}$$

This problem can be solved using Lagrange Multipliers, where by transforming it to its dual problem will allow us to generalize the linear case to the nonlinear case.

The solution that minimizes the primal problem subject to the constraints is given by

$$\max_{\alpha} W(\alpha) = \max_{\alpha}(\min_{\overrightarrow{w},b} \Phi(\overrightarrow{w}, b, \alpha)) \tag{6}$$

Then we construct the lagrangian

$$\Phi(\overrightarrow{w}, b, \alpha) = \frac{1}{2}||\overrightarrow{w}||^2 - \sum_{i=1}^{l} \alpha_i(y_i[\overrightarrow{w}.\overrightarrow{x}_i + b] - 1) \tag{7}$$

The solution of this optimization problem is determined by the saddle point of this Lagrangian. The minimum with respect to $\overrightarrow{w}$ and b of the Lagrangian, α, is given by[1]

$$\frac{\partial \Phi}{\partial b} = 0 \implies \sum_{i=1}^{l} \alpha_i y_i = 0 \tag{8}$$

$$\frac{\partial \Phi}{\partial w} = 0 \implies \overrightarrow{w}^* = \sum_{i=1}^{l} \alpha_i^* y_i \overrightarrow{x}_i \tag{9}$$

$$Constraint(1) \qquad y_i(\overrightarrow{w}^*.\overrightarrow{x}_i + b*) - 1 \geq 0 \tag{10}$$

$$MultiplierCondition \qquad \alpha_i \geq 0 \tag{11}$$

$$ComplementarySlackness \qquad \alpha_i^*[y_i(\overrightarrow{w}^*.\overrightarrow{x}_i + b^*) - 1] = 0 \tag{12}$$

Eq. 9 shows that the solution of optimal hyperplane can be written as a linear combination of the training vectors. Where only training vectors $\overrightarrow{x}_i$ with $\alpha_i > 0$ involved in the derivation of Eq. 9. Substituting Eq. 8 and Eq. 9 in Eq. 7 the dual problem is

$$\max_{\alpha} W(\alpha) = \max_{\alpha} -\frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j \overrightarrow{x}_i.\overrightarrow{x}_j + \sum_{i=1}^{l} \alpha_i \tag{13}$$

[1]Using the superscript * to denote the *optimal* values of the cost function.

By the linearity of the dot product and Eq. 9, the decision function (Eq. 3) is given by

$$f(\overrightarrow{x}) = sign(\sum_{i=1}^{l} y_i \alpha_i (\overrightarrow{x} . \overrightarrow{x}_i) + b^*) \tag{14}$$

For any support vector $\overrightarrow{x}_i$.

To this end, we discussed the case in which the data is separable. However, in general this will not be the case. So how can we extend the optimal separating hyperplane to find feasible solution for non-linearly separable case?

In order to deal with this case, [9] introduced positive slack variables ξ_i, $i = 1, ..., l$ which measure the amount of violation of the constraints. And a penalty function given by

$$F_\sigma(\xi) = \sum_i \xi^\sigma \sigma > 0, \tag{15}$$

Eq. 1 and Eq. 2 are modified for the non-separable case to

$$\overrightarrow{w} . \overrightarrow{x}_i + b \geq 1 - \xi_i, \ for \ y_i = 1 \tag{16}$$

$$\overrightarrow{w} . \overrightarrow{x}_i + b \leq -1 + \xi_i, \ for \ y_i = -1 \tag{17}$$

$$\xi_i \geq 0 \ \forall i. \tag{18}$$

The new primal problem is given by

$$Minimize \ \Phi(\overrightarrow{w}, \xi) \ = \ \frac{1}{2} ||\overrightarrow{w}||^2 + C(\sum_{i=1}^{l} \xi_i)^k \tag{19}$$

Subject to

$$y_i(\overrightarrow{w} . \overrightarrow{x}_i + b) \geq 1 - \xi_i \qquad i = 1, ..., l \tag{20}$$

$$\xi_i \geq 0 \qquad i = 1, ..., l \tag{21}$$

where C controls the tradeoff between the penalty and margin and to be chosen by the user, a larger C corresponds to assigning a higher penalty to errors. Penalty functions of the form $C(\sum_{i=1}^{l} \xi_i)^k$ will lead to convex optimization problems for positive integers k [35].

To be on the wrong side of the separating hyperplane, a data-case would need $\xi_i > 1$. Hence, the $\sum_i \xi_i$ could be interpreted as measure of how the violations are, and is an upper bound on the number of violations. We construct the lagrangian,

$$\Phi(\overrightarrow{w}, b, \xi, \alpha, \mu) = \frac{1}{2} ||\overrightarrow{w}||^2 + C(\sum_{i=1}^{l} \xi_i)^k - \sum_{i=1}^{l} \alpha_i (y_i [\overrightarrow{w} . \overrightarrow{x}_i + b] - 1 + \xi_i) - \sum_{i=1}^{N} \mu_i \xi_i \tag{22}$$

The lagrangian has to be minimized with respect to $\overrightarrow{w}$, $\overrightarrow{b}$, ξ and maximized with respect to α, μ. The dual problem is given by [34]

$$\max_{\alpha} W(\alpha, \mu) = \max_{\alpha,\mu}(\min_{\overrightarrow{w},b,\xi} \Phi(\overrightarrow{w}, b, \alpha, \xi, \mu)) \tag{23}$$

Differentiating Eq. 23 we derive the Karush-Kuhn-Tucker (KKT) conditions

$$\frac{\partial \Phi}{\partial b} = 0 \implies \sum_{i=1}^{l} \alpha_i y_i = 0 \tag{24}$$

$$\frac{\partial \Phi}{\partial \overrightarrow{w}} = 0 \implies \overrightarrow{w}^* = \sum_{i=1}^{l} \alpha_i^* y_i \overrightarrow{x}_i \tag{25}$$

$$\frac{\partial \Phi}{\partial \xi} = 0 \implies C - \alpha_i - \mu_i = 0 \tag{26}$$

$$Constraint(1) \qquad y_i(\overrightarrow{w}^*.\overrightarrow{x}_i + b*) - 1 + \xi_i \geq 0 \tag{27}$$

$$Constraint(2) \qquad \xi_i \geq 0 \tag{28}$$

$$MultiplierCondition(1) \qquad \alpha_i \geq 0 \tag{29}$$

$$MultiplierCondition(2) \qquad \mu_i \geq 0 \tag{30}$$

$$ComplementarySlackness(1) \qquad \alpha_i^*[y_i(\overrightarrow{w}^*.\overrightarrow{x}_i + b^*) - 1 + \xi_i] = 0 \tag{31}$$

$$ComplementarySlackness(2) \qquad \mu_i \xi_i = 0 \tag{32}$$

Substituting the KKT equations we obtain:

$$\max_{\alpha} W(\alpha) = \max_{\alpha} -\frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j \overrightarrow{x}_i.\overrightarrow{x}_j + \sum_{i=1}^{l} \alpha_i \tag{33}$$

By the linearity of the dot product and Eq. 25, the decision function can be written as

$$f(x) = sign(\sum_{i=1}^{l} y_i \alpha_i (\overrightarrow{x}.\overrightarrow{x}_i) + b^*) \tag{34}$$

The extension to more complex decision surfaces is done by mapping the input variable $\overrightarrow{x}$ in a higher dimensional feature space, then apply linear classification in that space. In order to construct a hyperplane in a feature space we transform the n-dimensional input vector $\overrightarrow{x}$ into an N-dimensional feature vector through a choice of an N-dimensional vector function ϕ [9]:

$$\phi : \mathrm{R^n} \rightarrow \mathrm{R^N} \tag{35}$$

Then, we construct an N-dimensional linear separator $\overrightarrow{w}$ and a bias b for the transformed vectors [9]

$$x \rightarrow \phi(\overrightarrow{x}) = (a_1\phi_1(\overrightarrow{x}), a_2\phi_2(\overrightarrow{x}), ..., a_n\phi_n(\overrightarrow{x})) \tag{36}$$

where $\{a_n\}_{n=1}^{\infty}$ are some real numbers and $\{\phi_n\}_{n=1}^{\infty}$ are some real functions. Then, apply Soft Margin of SVM, substituting the variable $\overrightarrow{x}$ with the new "feature vector" $\phi(\overrightarrow{x})$. Under Eq. 36 SVM solution is given by

$$f(x) = sign(\sum_{i=1}^{l} y_i\alpha_i^*\phi(\overrightarrow{x}).\phi(\overrightarrow{x}_i) + b^*) \tag{37}$$

Constructing support vector networks comes from considering general forms of the dot-product in a Hilbert space [33]:

$$\phi(\overrightarrow{x}).\phi(y) \equiv k(\overrightarrow{x}, y) \tag{38}$$

According to the Hilbert-Schmidt Theory [32] any symmetric function $K(\overrightarrow{x}, y)$, with $K(\overrightarrow{x}, y) \in L_2$, can be expanded in the form:

$$K(\overrightarrow{x}, y) = \sum_{i=1}^{\infty} \lambda_i\phi_i(\overrightarrow{x})\phi_i(y) \tag{39}$$

Using Eq. 39 SVM solution is given by

$$f(\overrightarrow{x}) = sign(\sum_{i=1}^{l} y_i\alpha_i^*k(\overrightarrow{x}, \overrightarrow{x}_i) + b^*) \tag{40}$$

where $K(\overrightarrow{x}, \overrightarrow{x}_i)$ is called kernel function. The state of the art of SVMs evolved mapping the learning data from input space into higher dimensional feature space where the classification performance is increased. This has been developed by applying several kernels each with individual characteristics. Lately, the choice of the kernel became a widely discussed issue, since it reveals different performance result for various applications.

1.4 Thesis Overview

This thesis is organized as follows:

- The first Chapter contains an introduction to Spam phenomenon and Support Vector Machines, a brief review of some well known approaches found in the literature.

- In Chapter 2, we explore several feature mapping strategies in context of text categorization. We intensively investigate the effect of various combinations of term frequency, importance weight and normalization on spam filtering performance. Moreover, we propose the use of various string kernels and different distance-based kernels for spam filtering. Finally, we provide a detailed result for a fair comparison between different feature mapping and kernel classes using typical spam filtering criteria.

- In Chapter 3, we propose a framework of various online modes for spam filtering. We propose the use of online Support Vector Machines, Transductive Support Vector Machines and Active Online Support Vector Machines for spam filtering. We study proposed modes using different feature mapping and kernel classes, also.

- In the last Chapter, we summarize the various methodologies and contributions that were presented, and we propose some future research directions.

CHAPTER 2

Spam Filtering using Support Vector Machine

Automated spam filtering has been proposed as an efficient solution to overcome unwanted, overwhelmed emails. Machine learning (ML) approaches have proven to be effective in classification tasks, in particular, for solving spam problem. These approaches can be grouped into generative, and discriminative. Generative approaches attempt to build a probabilistic model. Alternatively, discriminative approaches involve building a learner model to discriminate future unseen unlabeled positive and negative examples based on seen examples. Among discriminative approaches SVMs has been shown as a "universal learner" [10], and promising classifier. Moreover, spam filtering using SVMs has been addressed as an instance of text categorization (TC).

TC is the task of constructing "automatic text classifier", in which the classifier is capable of assigning labels to stream of incoming natural language text documents according to their contents. Recently, ML approaches have dominated as a solution for this problem. Generally, supervised ML automatically constructs a classifier by learning, from an initial set of pre-classified documents $\Omega = \{\overrightarrow{d}_1, ..., \overrightarrow{d}_\Omega\} \subset D$, trained with pre-categories $C = \{c_1, ..., c_m\}$ [55]. Indeed, the construction of learner (called inductive process) for C relies on learning the characteristics of C from a training set of documents $Tr = \{\overrightarrow{d}_1, ..., \overrightarrow{d}_{|Tr|}\}$. The effectiveness of the classifier is tested by applying it to test set $Te = \Omega - Tr$ and checking out how often the classifier's decision match the true value of document encoded in the corpus. Spam filtering has been seen as a single-label TC. That is, the classifier assigns a Boolean value to each pair of unseen documents $\langle \overrightarrow{d}_j, c_i \rangle$ into two mutual categories, the relevant c_i and the irrelevant $\overline{c_i}$.

In this chapter, we discuss the effectiveness of SVMs for solving spam problem. Furthermore, we introduce varied feature mappings that have been employed to transform email

data into feature vectors usable by machine learning methods. Along with, we investigate the impact of using different kernels in the resulted performance.

2.1 Feature Mapping

Text Categorization, using kernel-based machines, evolves vector representation for the involved data. For some types of data, attributes are naturally in feature vector format, while others need some preprocessing to explicitly construct feature vectors that describe images, text data, etc. In this case, a document $\overrightarrow{d}_j$ is described as a weighted vector $\overrightarrow{d}_j = \langle w_{ij}, ..., w_{|T|j} \rangle$, of features $0 \leq w_{kj} \leq 1$, for handling word distribution over documents [40]. Among existing approaches, the text representation dissimilarity can be showed either on what one regards the meaningful units of text or what approach one seeks to compute term weight [55]. Terms usually identified with words syntactically [56] or statistically [57]. Moreover, term weights can be considered by occurrence of term in the corpus (term frequency) or by its presence or absence (binary). Generally, supervised TC classifier is engaged into three main phases: term selection, term weighting, and classifier learning [54]. In this section, briefly, we discuss different approaches that have been applied in text representation.

2.1.1 Feature Extraction

Many researchers have pointed out the importance of text representation in the performance of TC using SVMs. In the following, we review some possible term selection approaches and discuss their strength.

Hand-crafted features Many industrial spam filtering techniques have employed this approach, such as the open-source filter SpamAssassin [73]. Discrimination of messages essentially relies on human experts to identify features within message. Consequently, few irrelevant features are introduced in the feature domain, which keep the computation and storage cost minimum. On the other hand, the identified features are language-specific, demanding reformulation of features regarding each language. Moreover, the features require to change over time as the intelligent spammer may easily attack the filter [51]. Thus, the experts effort is important to predict the most informative features [74].

Bag-of-Word (BoW) approach, in contrast, requires less human effort and supply a wider, generic feature space. It is also known as "word-based feature". Particulary, the extraction of features is based on defining a substring of contiguous characters "word" w, where word boundary is specified using a set of symbolic delimiters such as whitespace, periods, and commas. Alternatively, the token document loses its context, content, and case of its words. All possible words w with a finite length in document $\overrightarrow{d}$ are mapped to sparse vector $\Phi_i(\overrightarrow{d})$ in feature space F of n dimensions, that is $\Phi_i(\overrightarrow{d}) = 1$ if the w_i is present in $\overrightarrow{d}$, and $\Phi_i(\overrightarrow{d}) = 0$, otherwise. Consequently, a very large feature space F is constructed. To solve this issue researchers suggest "Stop words" and "Stemming". Stop word list, particulary, removes poor descriptors such as auxiliaries, articles, connectives, propositions, for instance, and it might be created beforehand, based on word frequency. Next, using stemming [50] each word is replaced by its stem. Although, BoW has been shown to be efficient in solving spam filtering, it has been defeated with word obfuscation attack [51] which includes technique such as character substitution, intentional misspellings, and insertion of whitespace.

k-mer Another feature extraction technique is k-mer. Using k-mer approach the document can be represented by predefined sequences of contiguous characters (i.e. sub-strings) of length k, where the choice of k differs with different text corpora. k-mer is a language independent approach. Consequently, using this technique consists of defining the consecutive k characters only with no prior knowledge of language. This approach works efficiently in text categorization, where any mismatch effects the limited numbers of those parts (neighbor), and leaves the remainder inviolated (further). Additionally, k-mer feature space allows the insertion or deletion of any number of characters between k-mer "Overlapping". The choice of k is important. Too small value of k creates ambiguous features, while a too high value of k makes the chance of finding exact matches between strings improbable. Indeed, in a spam classification setting, the optimal value of k may be language dependent, or may vary with the amount of expected obfuscation within the text.

2.1.2 Term Weighting

Term weighting phase is a vital step in TC, involves converting each document d to vector space which can be efficiently processed by SVM. It consists of three main parts: Frequency transformation, importance weight, and normalization.

Frequency transformation

Raw Frequency (TF) [41] is the simplest measure to weight each word w in document d. The intuition behind raw frequency is that the importance for each w is proportional to its occurrence in the document d. The raw frequency of the word w in document d is given by:

$$W(d, w) = TF(d, w) \tag{1}$$

In TC, while raw frequency improves recall, it doesn't always improve precision, because of frequent appearance of words that have little discrimination power such as auxiliary. Another common approach is Logarithmic Frequency [41], which concerns logarithms of linguistic quantities rather than the quantities themselves. This is given by:

$$\log TF(d, w) = \log(1 + TF(d, w)) \tag{2}$$

Moreover, Inverse Frequency (ITF) is another approach proposed in [41]:

$$F(d, w) = 1 - \frac{\gamma}{f(d, w) + \gamma} \tag{3}$$

Where $\gamma > 0$.

Importance Weight

Inverse Document Frequency (IDF) [41] concerns the occurrence of words w across the whole corpus. In this method, it is assumed that words that rarely occur over the corpus are valuable, and disregards the occurrence of word within the document, which is expected to improve the precision. In other words, the importance of each word is inversely proportional to the number of documents in which the word appears. The IDF of word w is given by:

$$IDF(w) = \log(N/df(w)) \tag{4}$$

Where N is the total number of documents in the corpus, and $df(w)$ is the number of documents that contain the word w. One popular term weight frequency combination is TF-IDF. TF-IDF used to evaluate how important a word is to a document in corpus. The importance increases proportionally to the number of times a word appears in the document but it is offset by the frequency of the word in the corpus. This is given by:

$$TF - IDF(d, w) = TF(d, w) * \log(N/df(w)) \tag{5}$$

Redundancy [41], in contrast to IDF, concerns the empirical distribution of a word w over the entire documents d in the corpus. It is given by:

$$r_k = \log N + \sum_{i=1}^{N} \frac{f(d_i, w_k)}{f(w_k)} \log \frac{f(d_i, w_k)}{f(w_k)} \tag{6}$$

Where $f(d_i, w_k)$ is the co-occurrence of word w_k in document d_i, $f(w_k) = \sum_{i=1}^{N} f(d_i, w_k)$ is the number occurrences of term w_k in the whole document collection, and N is the total number of document in the corpus.

Normalization

Different emails vary in their length, where long email contains hundreds of words while short emails has some dozen words. Since long email is not more important than short email we divide word frequency by the total number of words in the document. This can be done by mapping the word frequency vector to the unit-sphere in the L_1 this is known as L_1-normalization, [41] given by:

$$f \longrightarrow \frac{f}{\|f\|_{l_1}} \tag{7}$$

We can also normalize emails by using L_2-normalization which has been used widely in SVM application as it yields to best error bounds. It is given by:

$$f \longrightarrow \frac{f}{\|f\|_{l_2}} \tag{8}$$

Furthermore, normalization of emails resist the "sparse data attack" [51], where spammers attempt to defeat the spam filtering by writing short emails instead.

$RBF_{Gaussian}$	$\exp\left(-\rho\|\vec{x}-\vec{x}_i\|^2\right)$
$RBF_{Laplacian}$	$\exp\left(-\rho\,\|\vec{x}-\vec{x}_i\|\right)$
RBF_{χ^2}	$\exp\left(-\rho\sum_i \frac{(\vec{x}_i-y_i)^2}{\vec{x}_i+y_i}\right)$
Inverse multiquadric Kernel	$\frac{1}{\sqrt{\|\vec{x}-\vec{x}_i\|^2+1}}$
Polynomial kernel	$(\vec{x}.\vec{x}_i+1)^d$
Sigmoid Kernel	$\tanh(\vec{x}.\vec{x}_i+1)$

Table 2.1: Examples of classic distance based kernels

2.2 Kernels

Another key design task, when constructing email spam filtering using SVMs, is the proper choice of kernel regarding the nature of the data. In this section, we explore different classes of kernels.

2.2.1 Distance Based Kernels

Support Vector Machines in classification problems, such as spam filtering, explore the similarity between input emails implicitly using inner product $K(X,Y) = \langle\Phi(X),\Phi(Y)\rangle$ i.e. kernel functions. Kernels are real-valued symmetric function $k(\vec{x},\vec{x})$ of $\vec{x} \in X$. Let $\chi = \{\vec{x}_1,...,\vec{x}_m\}$ be a set of vectors, the induced kernel matrix $K = k(\vec{x}_i,\vec{x}_j)_{i;j=1}^n$ is called positive definite (pd) if it satisfies $c^T Kc \geq 0$ for any vector $c \in \mathrm{R}^m$ [36]. These kernel functions got much attention as they can be interpreted to inner products in Hilbert spaces.

In distance based learning [36–38] the data samples $\vec{x}$ are not given explicitly but only by a distance function $d(\vec{x},\vec{x}')$. Distance measure, requires to be symmetric, has zero diagonal, i.e. $d(\vec{x},\vec{x}) = 0$, and be nonnegative. If a given distance measure does not satisfy these requirements, it can easily be symmetrized by $\bar{d}(\vec{x},\vec{x}') = \frac{1}{2}(d(\vec{x},\vec{x}') + d(\vec{x}',\vec{x}))$, given zero diagonal by $\bar{d}(\vec{x},\vec{x}') = d(\vec{x},\vec{x}') - \frac{1}{2}(d(\vec{x},\vec{x}) + d(\vec{x}',\vec{x}'))$ or made positive by $\bar{d}(\vec{x},\vec{x}') = |d(\vec{x},\vec{x}')|$. We call such a distance isometric to an L^2-norm if the data can be embedded in a Hilbert space H by $\Phi : \chi \longrightarrow H$ such that $\bar{d}(\vec{x},\vec{x}') = \| \Phi(\vec{x}) - \Phi(\vec{x}') \|$. After choice of an origin $O \in X$ every distance d induces a function

$$\langle\Phi(\vec{x}),\Phi(\vec{x}')\rangle_d^O = -\frac{1}{2}(d(\vec{x},\vec{x}')^2 - d(\vec{x},O)^2 - d(\vec{x}',O)^2) \tag{9}$$

Where $\langle\Phi(\vec{x}),\Phi(\vec{x}')\rangle_d^O$ represents the inner product in a Hilbert space X with respect to the origin O. Table 2.1 shows examples of these distance based kernels.

2.2.2 String Kernels

Recent researches suggest a new approach of using SVMs for text classification, based on a family of kernel functions called string kernels, that perform competitively with the state-of-the-art approaches. String kernels, in contrast of distance-based kernels, define the similarity between pair of documents by measuring the total occurrence of shared substrings of length k in feature space F. In this case, the kernel is defined via an explicit feature map. Additionally, string kernels are classified into two classes: the position-aware string kernel which takes advantage of positional information of characters/substrings in their parent strings such as inexact string match kernels, and the position-unaware string kernel such as spectrum kernel.

The efficiency of string kernels can be justified by different factors. First, it is an appealing method for highly inflected natural languages ("oriental languages" for instance, Japanese language, in text classification problems). Indeed, defining automatic word segmentation is more suspectable for errors where word delimiters utilized in western languages such as English text are easier to define like whitespace. Second, for spam filtering problem, for instance, spammer try to mislead filters by including non-alphabetical characters in their text, string kernels can handle these words. Third, it provides an automatic categorization for different types of document formats (See [46] for more details and discussion).

In the following section, let $\sum$ be the alphabet. The sequence is a string of symbols drawn from an alphabet, $s \in \sum^{|s|}$. The k-mer refers to k consecutive symbols, $\alpha = \alpha_1, \alpha_2, ..., \alpha_k \in \sum^k$. In this section, We briefly presents different string kernels.

Spectrum Kernel

The idea of spectrum kernel is to measure the total of all possible symbolic contiguous subsequences of a fixed length k contained in the document. Then, the spectrum feature is defined as [44]:

$$\Phi_k^{spectrm}(s) = (\phi_\alpha(s))_{\alpha \in \sum^k} \tag{10}$$

Where $\phi_\alpha(s)$ is the count of occurrences of α in the sequence s. The kernel calculates the dot product between the vectors holding all the k-mers counts for any pair of sequences:

$$K_k^{spectrm}(s_1, s_2) = \langle \Phi_k^{spectrm}(s_1), \Phi_k^{spectrm}(s_2) \rangle \tag{11}$$

The kernel value is large if two sequences share a large number of k-mers. Notice that, spectrum kernel is position independent. The kernel function can be computed very efficiently due to the increasing sparseness of longer k-mers.

Inexact String Match Kernels

Inexact String Match Kernels is an extension of the Spectrum kernel, which allow k-mer to match even if there have been a number of insertion, deletion, or character substitution.

Mismatch Kernel

The mismatch kernel [49] feature map obtains inexact matching of instance k-mers from the input sequence to k-mer features by allowing a restricted number of mismatches. The features used by mismatch kernel are the set of all possible subsequences of strings of a fixed length k. If two string sequences contain many k-length subsequences that differ by at most m mismatches, then their inner product under the mismatch kernel will be large. More precisely, the mismatch kernel is calculated based on shared occurrences of (k, m)-patterns in the data, where the (k, m)-pattern generated by a fixed k-length subsequence consists of all k-length subsequences differing from it by at most m mismatches [52].

For a fixed k-mer $\alpha = a_1 a_2 ... a_k$, $a_i \in \sum_{i=1}^{|a|}$, the (k, m)-neighborhood generated by α is the set of all k-length sequences β from $\sum$ that differ from α by at most m mismatches. We denote this set by $N(k, m)(\alpha)$. For a k-mer α, the feature map is defined as

$$\Phi_{(k,m)}^{Mismatch}(\alpha) = (\phi_\beta(\alpha))_{\beta \in \sum^k} \tag{12}$$

where $\phi_\beta(\alpha) = 1$ if β belongs to $N(k, m)(\alpha)$, and $\phi_\beta(\alpha) = 0$, otherwise. For m = 0, mismatch kernel, k-spectrum, and k-gram kernel are the same. The kernel is the dot product between the two k-mers count vectors, given by

$$\Phi_{(k,m)}^{Mismatch}(s_1, s_2) = \langle \Phi_{(k,m)}^{Mismatch}(s_1), \Phi_{(k,m)}^{Mismatch}(s_2) \rangle \tag{13}$$

Restricted-Gappy Kernel

For a fixed g-mer $\alpha = a_1 a_2 ... a_k$ ($a_i \in \sum_{i=1}^{k}$), let $G(g, k)(\alpha)$ be the set of all the k-length subsequences occurring in α (with up to $g - k$ gaps). Then the gappy feature map on α is

defined as [52]

$$\Phi^{Gap}_{(g,k)}(\alpha) = (\phi_\beta(\alpha))_{\beta \in \sum^k} \tag{14}$$

Where $\phi_\beta(\alpha) = 1$ if β belongs to $G(g,k)$, and $\phi_\beta(\alpha) = 0$, otherwise.

Extending the feature map to arbitrary finite sequences s by summing the feature vectors for all the g-mers in s :

$$\Phi^{Gap}_{(g,k)}(s) = \sum_{g-mers \alpha \in s} \phi_\beta(\alpha) \tag{15}$$

Wildcard Kernel

For the wildcard string kernel [52], the default alphabet is extended with a wildcard character $\sum \cup \{*\}$, where the wildcard character matches any symbol. The presence of the wildcard character in an α k-mer is position-specific.

A k-mer α matches a subsequence β in W if all non-wildcard entries of β are equal to the corresponding entries of α (wildcards match all characters). The wildcard feature map is given by

$$\Phi^{Wildcard}_{(k,m,\lambda)}(s) = \sum_{k-mers \alpha \in s} \phi_\beta(\alpha)_{\beta \in W} \tag{16}$$

Where $\phi_\beta(\alpha) = \lambda^i$ if α matches pattern β containing j wildcard characters, Where $\phi_\beta(\alpha) = 0$ if α does not match β, and $0 < \lambda \leq 1$.

Weighted Degree Kernel

WD kernel [48] basic idea is to count the occurrences of k-mer substrings at corresponding positions in a comparable pair. The WD kernel of order d compares two sequences x and x' of equal length l by summing all contributions of k-mer matches of lengths $k \in \{1, ..., d\}$, weighted by coefficients β_k:

$$k(x, x') = \sum_{k=1}^{d} \beta_k \sum_{i=1}^{l-k+1} I(s_{k,i}(x) = s_{k,i}(x')) \tag{17}$$

Where $s_{k,i}(x)$ is the string of length k starting at position i of the sequence x, $I(.)$ is the indicator function which evaluates to 1 when its argument is true and to 0, otherwise, and $\beta_k = 2\frac{d-k+1}{2(d+1)}$.

Weighted Degree Kernel with Shift

WDs kernel [49] can be shown as a mixture between spectrum kernel and the WD kernel. It is defined as:

$$k(x,x') = \sum_{k=1}^{d} \beta_k \sum_{i=1}^{l-k+1} \sum_{j=0,j+i\leq l}^{J} \delta_j \mu_{k,i,j,x,x'} \tag{18}$$

Where $\mu_{k,i,j,x,x'} = I(s_{k,i+j}(x) = s_{k,i}(x')) + I(s_{k,i}(x) = s_{k,i+j}(x'))$, $\beta_k = 2\frac{d-k+1}{2(d+1)}$, $\delta_j = \frac{1}{(2(j+1))}$, $s_{k,i}(x)$ is the subsequence of x of length k that starts at position i. The idea is to count the matches between two sequences x and x' between the words $s_{k,i}(x)$ and $s_{k,i}(x')$ where $s_{k,i}(x) = x_i x_{i+1} ... x_{i+k-1}$ for all i and $1 \leq k \leq d$. The parameter d denotes the maximal length of the words to be compared, and J is the maximum distance by which a sequence is shifted.

String subsequence kernel (SSK)

SSK [39] exploits the similarity between pair of documents by searching for more shared substrings the more substrings in common, the more similar they are; where substrings are a sequence of unnecessary contiguous characters in text document. Moreover, the contiguity of such substring can be considered by a decay factor λ of full length of substrings in the document [39]. The degree of contiguity of the subsequence in the input string s determines how much it will contribute to the comparison. For an index sequence $i = (i_1, ..., i_k)$ identifying the occurrence of a subsequence $u = s(i)$ in a string s, we use $l(i) = i_k - i_1 + 1$ to denote the length of the string in s. SSK is given by

$$K_n(s,t) = \sum_{u\in\sum^n} \langle \phi_u(s).\phi_u(t) \rangle = \sum_{u\in\sum^n} \sum_{i:u=s[i]} \lambda^{l(i)} \sum_{j:u=t[j]} \lambda^{l(j)} = \sum_{u\in\sum^n} \sum_{i:u=s[i]} \sum_{j:u=t[j]} \lambda^{l(j)+l(j)} \tag{19}$$

Where s and t are substrings, denoted by the length of $|s|$ and $|t|$, respectively.

2.3 Experimental Results

Experiments have been carried out to assess and compare the performance of SVMs classification in spam categorization problem. Recently, spam filtering using SVM classifier has been tested and deployed using linear kernel weighted using binary weighting schemes [12] [14]

[19]. We extend previous research on spam filtering, as we consider two main tasks. Firstly, we compare the use of various feature mapping techniques described in section 2.1 for spam email filtering. Secondly, we investigate the use of string kernels with a number of classical kernels and exploring that in terms of accuracy, precision, recall, F1, and running classification time. In seek of comparison, the performance of each task is examined using the same version of spam data set and the same pre-processing is applied for different kernels.

Data Sets in the purpose of comparison evaluation, we used trec05-p1 [76] data set which is a large publicly available spam data set. This data set has 92,189 labeled spam and legitimate emails, with 57% spam rate, and 43% ham rate. Moreover. the emails in the data set have a canonical order, where we used the first 50,000 emails for training and the remaining for testing.

Evaluation Criteria experiments have been evaluated using typical performance measures that has been proposed for spam filtering problem [21]:

1. $Precision$ is the proportion of retrieved items that are relevant, measured by the ratio of the number of relevant retrieved items to the total number of retrieved items.

$$Precision = \frac{true\ \ positive}{true\ \ positive + false\ \ positive}$$

2. *Recall* is the proportion of relevant items retrieved, measured by the ratio of the number of relevant retrieved items to the total number of relevant items in the collection.

$$Recall = \frac{true\ \ positive}{true\ \ positive + false\ \ negative}$$

3. $F - measure$ summarizes the performance of a given classifier.

$$F - measure = \frac{\left(1 + \beta^2\right)\ * precision * recall}{\beta^2 * precision + recall}$$

   Where β determines the amount of weight assigned to precision and recall. We used $\beta = 1$ for equally weighted precision and recall (also known as F 1).

4. *Accuracy*: is a typical performance measure that gives an indication of overall well performance of a given classifier.

(E1)	$Polynomial.TF.L_1$	(E2)	$Polynomial.TF.L_2$
(E3)	$RBF_{Gaussian}.TF.L_1$	(E4)	$RBF_{Gaussian}.TF.L_2$
(E5)	$RBF_{Laplacian}.TF.L_1$	(E6)	$RBF_{Laplacian}.TF.L_2$
(E7)	$RBF_{\chi^2}.TF.L_1$	(E8)	$RBF_{\chi^2}.TF.L_2$
(E9)	$Sigmoid.TF.L_1$	(E10)	$Sigmoid.TF.L_2$
(E11)	$Inversemultiquadric.TF.L_1$	(E12)	$Inversemultiquadric.TF.L_2$
(E13)	$Polynomial.logTF.L_1$	(E14)	$Polynomial.logTF.L_2$
(E15)	$RBF_{Gaussian}.logTF.L_1$	(E16)	$RBF_{Gaussian}.logTF.L_2$
(E17)	$RBF_{Laplacian}.logTF.L_1$	(E18)	$RBF_{Laplacian}.logTF.L_2$
(E19)	$RBF_{\chi^2}.logTF.L_1$	(E20)	$RBF_{\chi^2}.logTF.L_2$
(E21)	$Sigmoid.logTF.L_1$	(E22)	$Sigmoid.logTF.L_2$
(E23)	$Inversemultiquadric.logTF.L_1$	(E24)	$Inversemultiquadric.logTF.L_2$
(E25)	$Polynomial.ITF.L_1$	(E26)	$Polynomial.ITF.L_2$
(E27)	$RBF_{Gaussian}.ITF.L_1$	(E28)	$RBF_{Gaussian}.ITF.L_2$
(E29)	$RBF_{Laplacian}.ITF.L_1$	(E30)	$RBF_{Laplacian}.ITF.L_2$
(E31)	$RBF_{\chi^2}.ITF.L_1$	(E32)	$RBF_{\chi^2}.ITF.L_2$
(E33)	$Sigmoid.ITF.L_1$	(E34)	$Sigmoid.ITF.L_2$
(E35)	$Inversemultiquadric.ITF.L_1$	(E36)	$Inversemultiquadric.ITF.L_2$
(E37)	$Polynomial.logTF-IDF.L_1$	(E38)	$Polynomial.logTF-IDF.L_2$
(E39)	$RBF_{Gaussian}.logTF-IDF.L_1$	(E40)	$RBF_{Gaussian}.logTF-IDF.L_2$
(E41)	$RBF_{Laplacian}.logTF-IDF.L_1$	(E42)	$RBF_{Laplacian}.logTF-IDF.L_2$
(E43)	$RBF_{\chi^2}.logTF-IDF.L_1$	(E44)	$RBF_{\chi^2}.logTF-IDF.L_2$
(E45)	$Sigmoid.logTF-IDF.L_1$	(E46)	$Sigmoid.logTF-IDF.L_2$
(E47)	$Inversemultiquadric.logTF-IDF.L_1$	(E48)	$Inversemultiquadric.logTF-IDF.L_2$
(E49)	$Polynomial.IDF.L_1$	(E50)	$Polynomial.IDF.L_2$
(E51)	$RBF_{Gaussian}.IDF.L_1$	(E52)	$RBF_{Gaussian}.IDF.L_2$
(E53)	$RBF_{Laplacian}.IDF.L_1$	(E54)	$RBF_{Laplacian}.IDF.L_2$
(E55)	$RBF_{\chi^2}.IDF.L_1$	(E56)	$RBF_{\chi^2}.IDF.L_2$
(E57)	$Sigmoid.IDF.L_1$	(E58)	$Sigmoid.IDF.L_2$
(E59)	$Inversemultiquadric.IDF.L_1$	(E60)	$Inversemultiquadric.IDF.L_2$

Table 2.2: Case of experiments

5. *(AUC) area under the ROC* specifies the probability that, when we draw one positive and one negative example at random, the decision function assigns a higher value to the positive than to the negative example.

Experimental Setup SVM^{light} [20] package was used as an implementation of SVMs. SVM^{light} [1]has proved its effectiveness and efficiency in the context of text categorization problem using large data sets. We set the value of ρ in RBF kernels, and C for the soft margin via 10-fold cross-validation. We ran experiments for similar length of substrings used in string kernels (value of k parameter).

Results to examine the use of different feature mapping, we evaluate trec05p-1 data set using generic combinations of feature mapping approaches. As a preprocessing step, the textual part of each email was represented by a concatenation of headers (i.e. sender and recipient), subject line and body of the email, where HTML tags presented in the body were substituted, if

[1]http://svmlight.joachims.org/

present. Moreover, using BoW approach, each email is tokenized using symbol delimiters (i.e. whitespace), and two dictionaries of words are constructed. The first dictionary consists of entire extracted words without any alteration or reduction (50000 words). The second dictionary has a reduced number of words (25000 words), where words in stop words list provided in [56] were removed and stemming has been performed on remaining words by means of Porter's stemmer [50]. Moreover, punctuation have been removed and all letters have been converted to lowercase. We tested 60 combinations of frequency transformations, importance weight, text-length normalization and kernel functions. Table 2.2 lists combinations involved in this experiments.

Considering results in Table 2.3, classification performances vary among kernels. Indeed, the precision is between 50% and 81.91%, recall results are between 46.67% and 81.14%, and F1 is between 54.65% and 75.20%. These variations are due to different weighting schemes. While RBF_{χ^2}.TF reports highest F1 with 75.2%, RBF_{χ^2}.ITF achieves highest precision with 81.91%. Table 2.5 shows, a slight improvement for F1 with 81.23% and precision (81.91%), while recall remains the same (80.56%). Thus, RBF_{χ^2}.TF has gained the best performance in terms of F1, precision, and recall. Clearly, classification performance is better when no stop word list and stemming were applied. Note that, term frequency alone, such as TF, logTF produces comparable performances for all combinations, and outperformed term frequency combined with importance weights. IDF is the worst performing weighting scheme which confirms previous results in [41] that its not the best weighting scheme for text classification. Results in Tables 2.7 and 2.9 confirm our previous observations, and show improved precision, recall, and F1 with, 91.81%, 90.88%, and 91.34%, respectively. An important observation is that emails normalized using L_2-norm has given better result than emails normalized with L_1-norm. In addition, TF-IDF common weighting approach has performed well with L_2-norm, and generally it has revealed medium performance.

In next experiment, data have been presented as a sequence of symbols without any processing and different string kernels have been employed. We ran the experiments for different values of substrings, recall that, the length of the sequence significantly influences the performance of the string kernel to some degree [44], we set the length of all substrings used in those experiments equal to 4. Where experiments consider substring less than 3 states in less performance (the results in not included here). We varied the value of the decay vector λ for

SSK to see its influence in the performance, where the higher value of λ gives more weight to non-contiguous substrings ($\lambda = 0.4$ has provided a better performance). For mismatch kernel (k, m), wildcard kernel (k,w), and gappy kernel (g,k), the experiments have taken place with fixed values of allowed mismatch, wild card and gaps which are $m = 1$, $w = 2$, $k = 2$, respectively, as allowing higher mismatch will increase the computation cost. Table 2.11 lists the results. Thus, the performance among all string kernels are quite similar. For position un-aware kernels, SSK kernel and Spectrum kernel performed the best.

On the basis of kernels comparison string kernels performed better than distance-based kernels. Besides F1, precision, and recall, we evaluate involved kernels in terms of their computational efficiency, in order to provide insight into the kernels impact on filtering time. We measured the duration of computation for all kernels (see results in Tables 2.12 and 2.13). As expected, string kernels were defeated by their computational cost [52]. The cost of computing each string kernels entry $K(\overrightarrow{x}, \overrightarrow{x}')$ scales as $O(|\overrightarrow{x}||\overrightarrow{x}'|)$ in the length of the input subsequences [52]. Polynomial kernel weighting with TF has reported the minimum time among involved kernels (with 0.03 m), although, it has less performance. Although, spectrum kernel has reported the lowest time (19.45 m) among string kernels, with higher recall, and precision. Its time is about three times worse than worst distance based kernels, which is reported by $RBF_{Laplacian}$.ITF with 6.45 m. For AUC evaluation, note that although the different kernels and feature mapping methods achieve different performance level, with distance-based kernels normalized using $L1$-norm being the weakest and string kernels being the strongest, the difference between different performances is stable through the experiments (see Tables 2.4, 2.6, 2.8 and 2.10 for distance-based kernels, and Table 2.14 for string kernels).

Conclusion our intent in this chapter, was to investigate how data presentation may effect the performance of SVM classification. Indeed, we examined the different combinations of term frequency, importance weight, normalization, and kernel functions. We achieved best performance with emails weighted using term frequency only and normalized using $L2$-norm. Besides, the kernel choice is crucial in classification problem. The good kernel is the kernel that gives a valuable information about the nature of data, and report good performance. RBF kernels have the higher performance among distance based kernels in most of experiments. In terms of F1, precision, and recall, string kernels have outperformed distance-based kernels. In the next chapter, we depart from batch setting to more realistic settings by investigating active online and transductive settings.

Kernel	Precision	Recall	F1
Polynomial.TF	0.6782	0.6010	0.6373
$RBF_{Gaussian}$.TF	0.6848	0.5490	0.6094
$RBF_{Laplacian}$.TF	0.6266	0.7132	0.6671
RBF_{χ^2}.TF	0.7433	0.7610	**0.7520**
Sigmoid.TF	0.5705	0.5504	0.5603
Inverse multiquadric.TF	0.7549	0.8041	0.7787
Polynomial.logTF	0.7586	0.7300	0.7440
$RBF_{Gaussian}$.logTF	0.6550	0.4679	0.5459
$RBF_{Laplacian}$.logTF	0.5640	0.6451	0.6018
RBF_{χ^2}.logTF	0.8210	0.7589	0.7887
Sigmoid.logTF	0.5573	0.6234	0.5885
Inverse multiquadric.logTF	0.6522	0.7122	0.6809
Polynomial.ITF	0.7601	0.7567	0.7584
$RBF_{Gaussian}$.ITF	0.5151	0.6073	0.5574
$RBF_{Laplacian}$.ITF	0.6875	0.6678	0.6775
RBF_{χ^2}.ITF	**0.8191**	0.7956	0.8072
Sigmoid.ITF	0.6137	0.7132	0.6597
Inverse multiquadric.ITF	0.7549	**0.8114**	0.7821
Polynomial.IDF	0.7050	0.5618	0.6253
$RBF_{Gaussian}$.IDF	0.5151	0.7073	0.5961
$RBF_{Laplacian}$.IDF	0.7000	0.6660	0.6826
RBF_{χ^2}.IDF	0.6700	0.4667	0.5502
Sigmoid.IDF	0.6073	0.7132	0.6560
Inverse multiquadric.IDF	0.5000	0.7576	0.6024
Polynomial.TF-IDF	0.5586	0.6300	0.5922
$RBF_{Gaussian}$.TF-IDF	0.5980	0.6956	0.6431
$RBF_{Laplacian}$.TF-IDF	0.5500	0.5430	0.5465
RBF_{χ^2}.TF-IDF	0.6191	0.6951	0.6549
Sigmoid.TF-IDF	0.6789	0.7231	0.7003
Inverse multiquadric.TF-IDF	0.7436	0.8031	0.7722
Polynomial.logTF-IDF	0.5900	0.7729	0.6692
$RBF_{Gaussian}$.logTF-IDF	0.6920	0.6271	0.6580
$RBF_{Laplacian}$.logTF-IDF	0.6834	0.6312	0.6563
RBF_{χ^2}.logTF-IDF	0.6799	0.6534	0.6664
Sigmoid.logTF-IDF	0.6697	0.6423	0.6557
Inverse multiquadric.logTF-IDF	0.5678	0.6543	0.6080

Table 2.3: The performance of SVM spam filtering on trec05-1 normalized using L_1-norm, and stop words have been removed

Precision	TF	logTF	ITF	IDF	TF-IDF	logTF-IDF
Polynomial	0.6510	0.7602	0.7743	0.6379	0.6133	0.6702
$RBF_{Gaussian}$	0.5922	0.5609	0.5700	0.6018	0.6611	0.6709
$RBF_{Laplacian}$	0.6799	0.6264	0.6891	0.7010	0.5587	0.6699
RBF_{χ^2}	0.7607	0.8104	0.8253	0.5701	0.6688	0.6830
Sigmoid	0.5742	0.6001	0.6873	0.6701	0.7114	0.6731
Inverse multiquadric	0.7945	0.6965	0.7998	0.6254	0.7828	0.6340

Table 2.4: The AUC value of trec05-1 normalized using L_1-norm, and stop words have been removed

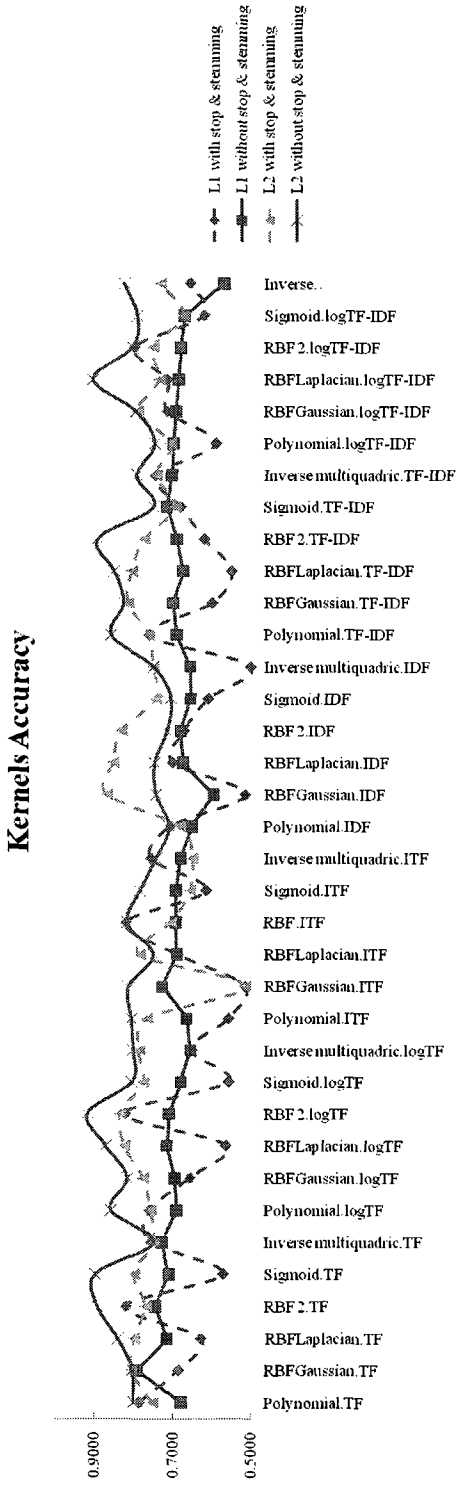

Figure 2.1: The overall Performance of spam classification, where different feature mapping are applied

Kernel	Precision	Recall	F1
Polynomial.TF	0.7841	0.7277	0.7548
$RBF_{Gaussian}$.TF	0.7899	0.7012	0.7429
$RBF_{Laplacian}$.TF	0.7145	0.7455	0.7297
RBF_{χ^2}.TF	**0.8191**	**0.8056**	**0.8123**
Sigmoid.TF	0.7089	0.7345	0.7215
Inverse multiquadric.TF	0.7278	0.7345	0.7311
Polynomial.logTF	0.6899	0.7108	0.7002
$RBF_{Gaussian}$.logTF	0.6933	0.7178	0.7053
$RBF_{Laplacian}$.logTF	0.7145	0.7422	0.7281
RBF_{χ^2}.logTF	0.7099	0.7345	0.7220
Sigmoid.logTF	0.6790	0.7155	0.6968
Inverse multiquadric.logTF	0.6532	0.7125	0.6816
Polynomial.ITF	0.6645	0.7289	0.6952
$RBF_{Gaussian}$.ITF	0.7266	0.7166	0.7216
$RBF_{Laplacian}$.ITF	0.6891	0.7326	0.7102
RBF_{χ^2}.ITF	0.6922	0.7456	0.7179
Sigmoid.ITF	0.6921	0.7178	0.7047
Inverse multiquadric.ITF	0.6789	0.7167	0.6973
Polynomial.IDF	0.6489	0.6900	0.6688
$RBF_{Gaussian}$.IDF	0.5951	0.6973	0.6422
$RBF_{Laplacian}$.IDF	0.6744	0.6999	0.6869
RBF_{χ^2}.IDF	0.6789	0.7061	0.6922
Sigmoid.IDF	0.6531	0.6723	0.6626
Inverse multiquadric.IDF	0.6549	0.5956	0.6238
Polynomial.TF-IDF	0.6899	0.6723	0.6810
$RBF_{Gaussian}$.TF-IDF	0.6983	0.7156	0.7068
$RBF_{Laplacian}$.TF-IDF	0.6734	0.7022	0.6875
RBF_{χ^2}.TF-IDF	0.6903	0.7321	0.7106
Sigmoid.TF-IDF	0.7144	0.7201	0.7172
Inverse multiquadric.TF-IDF	0.7014	0.6845	0.6928
Polynomial.logTF-IDF	0.6978	0.6845	0.6911
$RBF_{Gaussian}$.logTF-IDF	0.7145	0.6921	0.7031
$RBF_{Laplacian}$.logTF-IDF	0.7145	0.7298	0.7221
RBF_{χ^2}.logTF-IDF	0.7989	0.7756	0.7871
Sigmoid.logTF-IDF	0.6189	0.7326	0.6710
Inverse multiquadric.logTF -IDF	0.6549	0.6114	0.6324

Table 2.5: The performance of SVM spam filtering on trec05-1 normalized using L_1-norm, and without removing stop words

Precision	TF	logTF	ITF	IDF	TF-IDF	logTF-IDF
Polynomial	0.7746	0.7201	0.7112	0.6796	0.6965	0.7067
$RBF_{Gaussian}$	0.7576	0.7212	0.7338	0.6568	0.7189	0.7186
$RBF_{Laplacian}$	0.7414	0.7433	0.7354	0.7041	0.7142	0.7317
RBF_{χ^2}	0.8303	0.7378	0.7400	0.7076	0.7275	0.8022
Sigmoid	0.7379	0.7105	0.7199	0.6786	0.7342	0.6834
Inverse multiquadric	0.7493	0.6984	0.7097	0.6387	0.7096	0.6464

Table 2.6: The AUC value of trec05-1 normalized using L_1-norm, and without removing stop words

Kernel	Precision	Recall	F1
Polynomial.TF	0.7509	0.7866	0.7683
$RBF_{Gaussian}$.TF	0.8034	0.8431	0.8228
$RBF_{Laplacian}$.TF	0.7984	0.8412	0.8192
RBF_{χ^2}.TF	0.7699	0.7869	0.7783
Sigmoid.TF	0.7973	0.7562	0.7762
Inverse multiquadric.TF	0.7499	0.7801	0.7647
Polynomial.logTF	0.7565	0.7609	0.7587
$RBF_{Gaussian}$.logTF	0.8134	0.7931	0.8031
$RBF_{Laplacian}$.logTF	0.8234	**0.8556**	**0.8392**
RBF_{χ^2}.logTF	0.8345	0.7845	0.8087
Sigmoid.logTF	0.7756	0.7685	0.7720
Inverse multiquadric.logTF	0.7856	0.7645	0.7749
Polynomial.ITF	0.7668	0.7212	0.7433
$RBF_{Gaussian}$.ITF	**0.8690**	0.8034	0.8349
$RBF_{Laplacian}$.ITF	0.7844	0.7612	0.7726
RBF_{χ^2}.ITF	0.8312	0.7843	0.8071
Sigmoid.ITF	0.7430	0.7934	0.7674
Inverse multiquadric.ITF	0.7503	0.7856	0.7675
Polynomial.IDF	0.6812	0.7388	0.7088
$RBF_{Gaussian}$.IDF	0.5151	0.7073	0.5961
$RBF_{Laplacian}$.IDF	0.7345	0.7985	0.7652
RBF_{χ^2}.IDF	0.7045	0.7312	0.7176
Sigmoid.IDF	0.6561	0.7066	0.6804
Inverse multiquadric.IDF	0.6493	0.7156	0.6808
Polynomial.TF-IDF	0.7610	0.7723	0.7666
$RBF_{Gaussian}$.TF-IDF	0.7754	0.7603	0.7678
$RBF_{Laplacian}$.TF-IDF	0.8056	0.7690	0.7869
RBF_{χ^2}.TF-IDF	0.7734	0.7651	0.7692
Sigmoid.TF-IDF	0.6934	0.7045	0.6989
Inverse multiquadric.TF-IDF	0.7429	0.7122	0.7272
Polynomial.logTF-IDF	0.7061	0.6680	0.6865
$RBF_{Gaussian}$.logTF-IDF	0.7896	0.7045	0.7446
$RBF_{Laplacian}$.logTF-IDF	0.8523	0.8063	0.8287
RBF_{χ^2}.logTF-IDF	0.7522	0.7199	0.7357
Sigmoid.logTF-IDF	0.6732	0.7178	0.6948
Inverse multiquadric.logTF-IDF	0.7311	0.7026	0.7166

Table 2.7: The performance of SVM spam filtering on trec05-1 normalized using L_2-norm, and stop words have been removed

Precision	TF	logTF	ITF	IDF	TF-IDF	logTF-IDF
Polynomial	0.7856	0.7732	0.7642	0.7249	0.7811	0.7102
$RBF_{Gaussian}$	0.8397	0.8204	0.8534	0.6079	0.7809	0.7621
$RBF_{Laplacian}$	0.8376	0.8610	0.7858	0.7799	0.8063	0.8409
RBF_{χ^2}	0.7923	0.8267	0.8198	0.7402	0.7874	0.7540
Sigmoid	0.7902	0.7867	0.7853	0.6934	0.7142	0.709
Inverse multiquadric	0.7810	0.7898	0.7786	0.6914	0.7432	0.7301

Table 2.8: The AUC value of SVM spam filtering on trec05-1 normalized using L_2-norm, and stop words have been removed

Kernel	Precision	Recall	F1
Polynomial.TF	0.7999	0.8073	0.8036
$RBF_{Gaussian}$.TF	0.8244	0.8037	0.8139
$RBF_{Laplacian}$.TF	0.9034	0.8427	0.8720
RBF_{χ^2}.TF	0.8918	0.8761	0.8839
Sigmoid.TF	0.8965	0.7876	0.8385
Inverse multiquadric.TF	0.7912	0.8066	0.7988
Polynomial.logTF	0.8580	0.8356	0.8467
$RBF_{Gaussian}$.logTF	0.8151	0.8243	0.8197
$RBF_{Laplacian}$.logTF	0.8712	0.8867	0.8789
RBF_{χ^2}.logTF	**0.9181**	**0.9088**	**0.9134**
Sigmoid.logTF	0.8033	0.8123	0.8078
Inverse multiquadric.logTF	0.7994	0.8414	0.8199
Polynomial.ITF	0.8566	0.8033	0.8291
$RBF_{Gaussian}$.ITF	0.8119	0.7924	0.8020
$RBF_{Laplacian}$.ITF	0.7489	0.7721	0.7603
RBF_{χ^2}.ITF	0.8916	0.8922	0.8919
Sigmoid.ITF	0.7833	0.7652	0.7741
Inverse multiquadric.ITF	0.7423	0.7694	0.7556
Polynomial.IDF	0.7077	0.7160	0.7118
$RBF_{Gaussian}$.IDF	0.7430	0.7689	0.7557
$RBF_{Laplacian}$.IDF	0.7456	0.7089	0.7268
RBF_{χ^2}.IDF	0.7134	0.7666	0.7390
Sigmoid.IDF	0.7034	0.6892	0.6962
Inverse multiquadric.IDF	0.7491	0.7014	0.7245
Polynomial.TF-IDF	0.8066	0.8245	0.8155
$RBF_{Gaussian}$.TF -IDF	0.8015	0.8173	0.8093
$RBF_{Laplacian}$.TF-IDF	0.8499	0.8846	0.8669
RBF_{χ^2}.TF-IDF	0.8163	0.7612	0.7878
Sigmoid.TF-IDF	0.7489	0.7820	0.7651
Inverse multiquadric.TF-IDF	0.7494	0.8041	0.7758
Polynomial.logTF-IDF	0.7466	0.7399	0.7432
$RBF_{Gaussian}$.logTF-IDF	0.7934	0.7054	0.7468
$RBF_{Laplacian}$.logTF-IDF	0.8400	0.8632	0.8514
RBF_{χ^2}.logTF-IDF	0.8034	0.7861	0.7947
Sigmoid.logTF-IDF	0.7900	0.7123	0.7491
Inverse multiquadric.logTF-IDF	0.8241	0.7324	0.7755

Table 2.9: The performance of SVM spam filtering on trec05-1 normalized using L_2-norm, and without removing stop words

Precision	TF	logTF	ITF	IDF	TF-IDF	logTF-IDF
Polynomial	0.8214	0.8634	0.8475	0.7266	0.8404	0.7411
$RBF_{Gaussian}$	0.8312	0.8375	0.8200	0.7698	0.8267	0.7612
$RBF_{Laplacian}$	0.8891	0.8958	0.7596	0.7414	0.8839	0.8679
RBF_{χ^2}	0.9042	0.9302	0.9101	0.7494	0.8035	0.8123
Sigmoid	0.8563	0.8243	0.7898	0.7101	0.7812	0.7614
Inverse multiquadric	0.8164	0.8357	0.7731	0.7389	0.7912	0.7906

Table 2.10: The AUC value of SVM spam filtering on trec05-1 normalized using L_2-norm, and without removing stop words

Kernel	Precision	Recall	F1
SSk	0.9278	0.9281	0.9279
Spectrum	0.9249	**0.9362**	**0.9305**
Mismatch	0.8745	0.9100	0.8919
Wildcard	**0.9356**	0.8890	0.9117
Gappy	0.9190	0.9167	0.9178
WD	0.8978	0.9021	0.8999
WDs	0.8987	0.9189	0.9087

Table 2.11: The performance of SVM spam filtering on trec05-1 using string kernels

	Polynomail	$RBF_{Gaussian}$	$RBF_{Laplacian}$	RBF_{χ^2}	Sigmoid	Inverse multiquadric
TF	0.03	0.40	3.48	0.49	0.10	3.43
logTF	0.45	1.25	5.54	5.47	6.36	3.21
ITF	1.55	2.39	**6.48**	5.49	3.03	4.02
IDF	0.47	2.02	2.05	4.57	3.07	4.11
TF-IDF	0.02	0.39	3.00	0.35	0.55	3.43
logTF-IDF	0.45	1.20	4.30	5.00	6.40	3.20

Table 2.12: Training time for different combinations of frequency and distance kernels

SSk	Spectrum	Mismatch	Wildcard	Gappy	WD	WDs
20.08	19.45	**21.43**	20.47	20.02	19.56	20.32

Table 2.13: Training time for string kernels

SSk	Spectrum	Mismatch	Wildcard	Gappy	WD	WDs
0.9458	0.9510	0.9089	0.9304	0.9374	0.9187	0.9276

Table 2.14: The AUC value of SVM spam filtering on trec05-1, using string kernels.

CHAPTER 3

Spam Filtering using Online Active Support Vector Machine

In reality, spam filtering is typically tested and deployed in an online setting, by proceeding incrementally. Online learning allows a deployed system to adapt itself in a dynamic environment. While, labeled data sets are not often affordable prior classification, label data set is time consuming and tedious process. To overcome this problem, researchers have introduced new strategies to reduce the amount of required labeled data without altering the performance of the classifier, this is known as "active learning".

In this chapter, we describe various methods and techniques to adapt spam filtering for real time settings. We develop new online active framework for spam filtering using string kernels. Furthermore, we review several strategies in both active learning and transductive learning. We present our detailed results based on different kernels and feature mapping discussed in Chapter 2.

3.1 Online SVM

Batch spam filtering, using SVM, has suffered from the dynamical nature of spam email characteristics. In batch model, training and testing sets are random samples drawn from a common source of populations which have been used for learning phase. In reality, spam filtering is a continuous task, the data is often collected continuously in time, and more importantly, the characteristics of spam emails evolve over time and no such common population exists [14]. Moreover, it is difficult to identify sufficient samples prior to spam filtering. Recently, more efforts have been spent in the development of online SVM learning algorithms [67] [68],

where incremental SVM [66] provides a framework for exact online learning.

Online learning has been shown to be efficient for boosting spam filtering performance. In contrast to batch model, the online model presents to the filter a sequence of emails W_1, $m_0 = (x_0, y_0)$ through $m_{k-1} = (x_{k-1}, y_{k-1})$, where sequence order is determined by the design (i.e. it might be in chronological order or even randomized). We adopted a simple algorithm introduced in [80] to adapt batch model to online model. The algorithm is summarized in Algorithm 1. Initially, suppose a spam filter is trained on training set $TR = \{(x_0, y_0), ..., (x_{i-1}, y_{i-1})\}$. The hyperplane with maximum margin is fully defined through small portion of emails called "Support Vectors" SVs. Next, when a new email is presented to the filter, corresponding to the KKT conditions, is poorly classified, then a new hyperplane is needed. Re-training of SVM from scratch involves solving a quadratic programming problem which is cost prohibitive. In SVM model, examples closer to the hyperplane are most uncertain and informative. Moreover, SVs are able to summarize the data space and preserve the essential class boundary. Consequently, in our model, using SVs as seeds (starting point) for the future retraining and discarding all non-SVs samples would not effect SVM classification performance.

3.2 Transductive Support Vector Machine (TSVM)

In Chapter 2, we deployed inductive SVM in which the learner attempts to build a model to approximate data from the whole problem space, and then using this model to predict output values for a new input vector. In other words, the learner has focused on constructing a universal model for spam, and general purpose strategy in detecting spam emails. However, individual's email has different characteristics. Indeed, it is difficult to use publicly labeled emails to classify individuals inbox, although may most of users agree that such emails are spams, the remaining may still wish to receive such emails [62]. In addition, acquiring labels is time prohibitive. TSVM model, in contrast to inductive SVM, estimates the value of a classification function at given points. In particular, TSVM model constructs a maximum margin by employing the large collection of unlabeled data jointly with a few labeled examples for improving generalization performance. In literature, several approaches have discussed TSVM from different perspectives, such as margin-based classification [58] [59], graph-based methods [60], and information regularization [61].

Algorithm 1: Online Support Vector Classifier

Set $W_1 = x_k, y_k$, for $k = 0, ..., i-1$. and $|E_2| = 0$
Train and obtain an optimal boundary f_1
for $k = i, ..., l.$ do
    obtain a new example $S_k = \{x_k, y_k\}$
    if $y_k f_{k-1}(x_k)$ violates the KKT conditions
    or $|E_{k-1}| > 0$ then
        if $y_k f_{k-1}(x_k)$ violates the KKT conditions or
        $|E_{k-1}| = 0$ then
            $W_k = \{Sx_i^{k-1}, Sy_i^{k-1}\}_{i=1}^{|SV_{k-1}|} \cup S_k$
        end if
        if $y_k f_{k-1}(x_k)$ satisfy the KKT conditions or
        $|E_{k-1}| > 0$ then
            $W_k = \{Sx_i^{k-1}, Sy_i^{k-1}\}_{i=1}^{|SV_{k-1}|} \cup E_{k-1}$
        end if
        if $y_k f_{k-1}(x_k)$ violates the KKT conditions and
        $|E_{k-1}| > 0$ then
            $W_k = \{Sx_i^{k-1}, Sy_i^{k-1}\}_{i=1}^{|SV_{k-1}|} \cup E_{k-1} \cup S_k$
        end if
        Re-train to obtain an optimal boundary f_k with W_k
        $E_k = \{x_i, y_i | y_i f_k(x_i) violates the KKT conditions\}_{i=1}^{k}$
    end if
end for
while $|E_1| > 0$ do
    $W_1 = SV_1 \cup E_1$
    Solving quadratic programming problem to obtain an optimal boundary f_1
    $E_1 = \{x_i, y_i | y_i f_l(x_i) violates the KKT conditions\}_{i=1}^{l}$
end while

In TSVM, suppose that the set of random independent identically distributed (i.i.d) training vectors drawn according to $P(\overrightarrow{x}, y) = P(y|\overrightarrow{x})P(\overrightarrow{x})$ belonging to two separate classes. Suppose we have a hyperplane $\overrightarrow{w}.\overrightarrow{x}_i + b$ which separates the two different classes and a sample set train S_{train} of n training examples $(\overrightarrow{x}_1, y_1), (\overrightarrow{x}_2, y_2), ..., (\overrightarrow{x}_n, y_n)$. Each training example consists of a feature vector $\overrightarrow{x} \in X$ and a binary label $y \in \{-1, +1\}$. In contrast to the inductive setting, the learner is also given a sample set test S_{test} of k test examples $(\overrightarrow{x}^*_1, y^*_1), (\overrightarrow{x}^*_2, y^*_2), ..., (\overrightarrow{x}^*_n, y^*_n)$ where y^*_i are the labels of $\overrightarrow{x}^*_i$ the classifier have to predict. In the linearly separable case, the optimization problem is solved by minimizing over $(y^*_1, ..., y^*_n, \overrightarrow{w}, b)$ [69]:

$$\frac{1}{2}||\overrightarrow{w}||^2 \tag{1}$$

subject to

$$\forall_{i=1}^{n} \quad : \quad y_i[\overrightarrow{w}.\overrightarrow{x_i} + b] \geq 1 \tag{2}$$

$$\forall_{j=1}^{k} \quad : \quad y_j^*[\overrightarrow{w}.\overrightarrow{x_j}^* + b] \geq 1 \tag{3}$$

$$\tag{4}$$

Where $\overrightarrow{w}$ is the normal of the hyperplane.

In the non-separable case, the optimization problem can be represented by solving the trade-off between maximizing the margin and minimizing the number of misclassified examples; this is given by minimizing over $(y_1^*, ..., y_n^*, \overrightarrow{w}, b, \xi_1, ..., \xi_n, \xi_1^*, ..., \xi_k^*)$ [69]:

$$\frac{1}{2}||\overrightarrow{w}||^2 + C\sum_{i=0}^{n}\xi_i + C\sum_{j=0}^{k}\xi_i^* \tag{5}$$

subject to

$$\forall_{i=1}^{n} \quad : \quad y_i[\overrightarrow{w}.\overrightarrow{x_i} + b] \geq 1 - \xi_i \tag{6}$$

$$\forall_{j=1}^{k} \quad : \quad y_j^*[\overrightarrow{w}.\overrightarrow{x_j}^* + b] \geq 1 - \xi_i^* \tag{7}$$

$$\forall_{i=1}^{n} \quad : \quad \xi_i > 0 \tag{8}$$

$$\forall_{j=1}^{k} \quad : \quad \xi_j^* > 0 \tag{9}$$

where ξ_i and ξ_i^* are slack variables which measure the violation of the constraints (related to labeled and unlabeled examples), C and C^* are parameters which control the tradeoff between the penalty and margin (chosen by the user).

3.3 Active Learning

Traditionally, the task of spam filtering involves a manual assignment of labels to emails in the data set. On one hand, manual labeling is error-prone, time and cost prohibitive. In reality, training data may not well represent the future emails to be classified, for instance, new user may not receive emails similar to what they have used to train their filters, while other users who received spam emails more often may not label each email they received [62]. This fact is exploited by spammers by establishing millions of emails in a "never-before-used" format sent to defeat spam filters prior the new format is learned. To this end, active learning provides an appealing solution to overcome labeling cost by identifying informative emails for which

labels are requested [70]. In this section, we describe several active learning strategies for SVM.

Pool-based approach, is a common approach developed in machine learning to reduce the labeling effort required by humans [72]. Pool-based model can be explained as follows. Suppose we have a two pools of labeled emails L and unlabeled emails U. Assume that emails $\overrightarrow{x}$ are i.i.d according to some underlying distribution $P(\overrightarrow{x})$, and the labels are distributed according to some conditional distribution $P(y|\overrightarrow{x})$. An active learner has three components: (f, q, L). The first component is a classifier, $f : L \longrightarrow \{-1, 1\}$, trained on the current set of labeled data L and unlabeled instances $u \in U$. $q(L)$ is the querying function that, given a current labeled set L, decides which instance in U to query next. In online setting, the active learner returns a classifier f after each pool-query, or after some fixed number of pool-queries in passive settings [65].

There are several methods for selecting these unlabeled batches $u \in U$, such as speculative sampling, batch-simple, and error-reduction [64]. Such methods can reduce labeling effort dramatically, without significant reduction in classification performance. However, prior applications of active learning in this setting have been both computationally expensive and prone to selecting redundant examples which have harmed classification performance. Angle diversity [63] approach introduces a strategy in attempt of diversity to select emails to label.

The main idea of angle-diversity is to select emails close to the hyperplane with high angle diversity [71]. Assume, two samples $\overrightarrow{x}_i$, $\overrightarrow{x}_j$, their normal vectors are given by $\phi(\overrightarrow{x}_i)$, $\phi(\overrightarrow{x}_j)$. The similarity between $\phi(\overrightarrow{x}_i)$ and $\phi(\overrightarrow{x}_j)$ is measured using cosine distance, given by [63]:

$$|\cos(\angle(h_i, h_j))| = \frac{|\langle\phi(\overrightarrow{x}_i, \overrightarrow{x}_i)\rangle|}{\|\phi(\overrightarrow{x}_i)\|\|\phi(\overrightarrow{x}_i)\|} = \frac{|k(\overrightarrow{x}_i, \overrightarrow{x}_j)|}{\sqrt{k(\overrightarrow{x}_i, \overrightarrow{x}_i)k(\overrightarrow{x}_j, \overrightarrow{x}_j)}} \tag{10}$$

In order to balance the distance to the classification hyperplane and the diversity of angles among samples, [63] introduces λ parameter. Incorporating λ (trade-off factor), the final score for the unlabeled instance $\overrightarrow{x}_i$ is given by:

$$arg\min_{x_i \in U}(\lambda|f(x_i)| + |1 - \lambda|) * (\max_{x_j \in S} \frac{|k(\overrightarrow{x}_i, \overrightarrow{x}_j)|}{\sqrt{k(\overrightarrow{x}_i, \overrightarrow{x}_i)k(\overrightarrow{x}_j, \overrightarrow{x}_j)}}) \tag{11}$$

where S is the sample set which is provided to users for label (feedback), $\overrightarrow{x}_i$ is the selected sample which will be added into S, $f(x_i)$ represents the distance from $\overrightarrow{x}_i$ to the hyperplane,

U represents unlabeled emails set, and λ is the parameter to adjust the weight of the angle diversity and the distance to the hyperplane. According to Eq. 11, we find unlabeled email $\overrightarrow{x}_i$ with smallest value, which guarantees that $\overrightarrow{x}_i$ is closer to the hyperplane, and at the same time, far away from the existing training emails. Thus, the email $\overrightarrow{x}_i$ will be included in the training set.

For Online Active SVM learning [65] messages come to the filter in a stream, and the filter must classify them one by one. Each time a new example presented to the filter, the filter has the option of requesting a label for the given message. The goal is for the filter to achieve strong classification performance with as few label requests as possible.

3.4 Experimental Results

In this section, we report results from experiments testing the effectiveness of the online, TSVM, and online active learning methods, presented in previous sections, with SVM as a base classifier for spam filtering. We used string kernels, along with combinations of distance-based kernels and feature mapping from chapter 2 for our experiments. In our experiments, we have focused on combinations with $L2$-norm, no stop words and stemming employed, since they give the best performance (see table 2.2).

Data Set to evaluate the strategies mentioned in previous sections we have conducted several experiments on two publicly available data sets, trec06p and trec05p-1. Trec05p-1 is described in chapter 2 and trec06p data set has 24912 labeled spam and 12910 legitimate emails. Our evaluation is based on precision, recall and F1.

Experiments Setup the value of ρ in RBF kernels, and C for the soft margin were determined via 10-fold cross-validation by training an inductive SVM on the entire data set. For TSVM, the value of C^* is set similar to C [78]. The length of substrings used in string kernels is set to 4 (value of k parameter). For mismatch kernel (k, m), wildcard kernel (k, w), and gappy kernel (g, k) the experiments have taken place with fixed values of allowed mismatch, wild card and gaps which are $m = 1$, $w = 2$, $k = 2$, respectively.

Online SVM learning

To demonstrate the effectiveness of Online SVM learning, we applied the Online learning algorithm described in section 3.1 to trec05p-1 data set, and compared it to batch learning. We

have chosen the first 80000 emails for training and the remaining 12189 emails for testing. Emails, in training and testing processes, have been represented as a stream of chronological order, as a classifier has to classify them one by one.

Generally, results show that all classes of kernels have improved performance in online model compared to batch model. Moreover, results vary significantly among kernels, e.g. precision is between 70.41% and 91.39% , recall is between 67.88% and 92.15%, and F1 results are between 67.40% and 90.65% for distance-based kernels. For string kernels, results are rather better, where precision varies between 89.67% and 96.89%, recall results are between 90.12% and 95.31%, and F1 reported results between 91.02% and 95.20%. The classification results for distance-based kernels are listed in Table 3.1. RBF_{χ^2} weighted with logTF has achieved best performance over other distance-based kernels in terms of precision, recall, and F1. In addition, one can note that the second and third highest recall have been achieved by $RBF_{Laplacian}$.TF-IDF and $RBF_{Laplacian}$.logTF-IDF, respectively, which prove that online setting provides natural environment for spam filtering. Moreover, for string kernels, SSK kernel reported the highest precision and F1 with 95.05%, 94.86%, respectively. While spectrum kernel has achieved the highest recall with 94.78%. Obviously, spectrum kernel and SSK have a very close performance in terms of precision, recall and F1. For position ware string kernels, restricted gappy kernel has a comparable performance to SSk and spectrum kernels. Results are given in Table 3.2.

Transductive SVM

To examine the effectiveness of TSVM classification, we conducted two experiments. In the first one, we trained TSVM classifier using 30000 labeled emails (from each class) from trec05p-1 data set and 37822 unlabeled emails from trec06 data set (semi-supervised settings). Then, TSVM was evaluated using the remaining emails in trec05p-1. In next experiment, trec05p-1 was divided into halves from each class with 30000 labeled and 50000 unlabeled emails for training, and the remaining for testing. In entire experiments, emails were represented in stream and the canonical order of emails was reserved.

In Tables 3.3, 3.4, 3.5 and 3.6, TSVM yields better filtering performance than its SVM counterpart. In particular, the performance of TSVM trained with unlabeled emails from the same test set led to better filtering performance. In both cases string kernels out-performed

Kernel	Precision	Recall	F1
Polynomial.TF	0.7865	0.8173	0.8016
$RBF_{Gaussian}$.TF	0.8689	0.8416	0.8550
$RBF_{Laplacian}$.TF	0.9189	0.8628	0.8900
RBF_{χ^2}.TF	0.8976	0.8444	0.8702
Sigmoid.TF	0.8987	0.7943	0.8433
Inverse multiquadric.TF	0.7867	0.8126	0.7994
Polynomial.logTF	0.8612	0.8234	0.8419
$RBF_{Gaussian}$.logTF	0.8366	0.8347	0.8356
$RBF_{Laplacian}$.logTF	0.8852	0.8876	0.8864
RBF_{χ^2}.logTF	**0.9215**	**0.9065**	**0.9139**
Sigmoid.logTF	0.7945	0.8248	0.8094
Inverse multiquadric.logTF	0.7652	0.8589	0.8093
Polynomial.ITF	0.8267	0.8180	0.8223
$RBF_{Gaussian}$.ITF	0.8631	0.7898	0.8248
$RBF_{Laplacian}$.ITF	0.7645	0.7721	0.7683
RBF_{χ^2}.ITF	0.8972	0.8873	0.8922
Sigmoid.ITF	0.7737	0.7928	0.7831
Inverse multiquadric.ITF	0.7778	0.7603	0.7690
Polynomial.IDF	0.6788	0.7569	0.7157
$RBF_{Gaussian}$.IDF	0.7578	0.7852	0.7713
$RBF_{Laplacian}$.IDF	0.7285	0.7857	0.7560
RBF_{χ^2}.IDF	0.7321	0.7783	0.7545
Sigmoid.IDF	0.7349	0.6956	0.7147
Inverse multiquadric.IDF	0.7371	0.6740	0.7041
Polynomial.TF-IDF	0.8356	0.8089	0.8220
$RBF_{Gaussian}$.TF -IDF	0.8312	0.8264	0.8288
$RBF_{Laplacian}$.TF-IDF	0.8611	0.8931	0.8768
RBF_{χ^2}.TF-IDF	0.8077	0.7854	0.7964
Sigmoid.TF-IDF	0.7937	0.8088	0.8012
Inverse multiquadric.TF-IDF	0.7877	0.8156	0.8014
Polynomial.logTF-IDF	0.7321	0.7450	0.7385
$RBF_{Gaussian}$.logTF-IDF	0.7385	0.7467	0.7426
$RBF_{Laplacian}$.logTF-IDF	0.8178	0.8879	0.8514
RBF_{χ^2}.logTF-IDF	0.7999	0.7934	0.7966
Sigmoid.logTF-IDF	0.8096	0.7670	0.7877
Inverse multiquadric.logTF-IDF	0.7988	0.7657	0.7819

Table 3.1: The performance of Online SVM spam filtering on trec05-1

Kernel	Precision	Recall	F1
SSk	**0.9505**	0.9468	**0.9486**
Spectrum	0.9466	**0.9478**	0.9472
Mismatch	0.9012	0.9145	0.9078
Wildcard	0.9432	0.8952	0.9186
Gappy	0.9256	0.9189	0.9222
WD	0.9189	0.9067	0.9128
WDs	0.9109	0.9001	0.9055

Table 3.2: The performance of Online SVM spam filtering on trec05-1 using string kernels

distance-based kernels. Consequently, It would be of interest to adopt string kernels in constructing filters for individuals inbox. The 2006 ECML/PKDD learning challenge tested several methods for semi-supervised learning on a small data set of emails spam and ham [62]. Our initial results confirmed the promising results reported in the challenge, which show that TSVMs is strong strategy for spam filtering.

Indeed, wildcard kernel reported the highest precision among kernels with 96.89%. Meanwhile, SSK reported the highest recall, and F1 with 95.31%, 95.20%, respectively. Besides, the best results we obtained for distance-based kernels were precision of 92.01%, recall of 91.57%, and F1 of 91.79% reported by RBF_{χ^2}.logTF. $RBF_{Laplacian}$.logTF is the second best distance-based kernel. However, inverse multiquadratic.IDF is the worst performing kernel which confirms our previous results in chapter 2.

It is interesting to investigate the classification performance against the number of involved unlabeled emails over time. From figure 3.1 we can see that unlabeled data can improve the results on this problem, especially in the case of few training data. However, when enough training data is available to the filter, the improvement is not much (graph is flat) [69]. Figure 3.2 shows the effect of varying the size of labeled training emails.

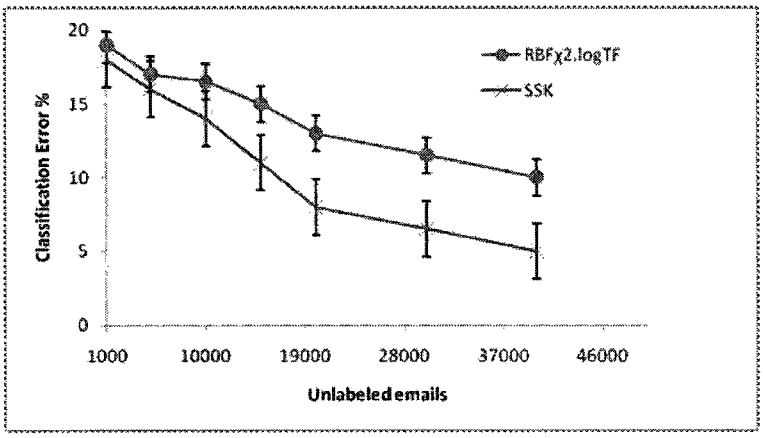

Figure 3.1: Classification Error on the trec05p-1 for RBF_{χ^2}.logTF and SSk kernels by varying the number of unlabeled emails in training for TSVM

Online Active Learning

We combined the online algorithm described in section 3.1 with angle diversity approach described in section 3.3 to perform online active learning. We chose the first 60000 emails

Kernel	Precision	Recall	F1
Polynomial.TF	0.7789	0.8045	0.7915
$RBF_{Gaussian}$.TF	0.8756	0.8501	0.8627
$RBF_{Laplacian}$.TF	0.9194	0.8794	0.8990
RBF_{χ^2}.TF	0.8930	0.8788	0.8858
Sigmoid.TF	0.8966	0.8170	0.8550
Inverse multiquadric.TF	0.7787	0.8240	0.8007
Polynomial.logTF	0.8537	0.8378	0.8457
$RBF_{Gaussian}$.logTF	0.8189	0.8490	0.8337
$RBF_{Laplacian}$.logTF	0.9012	0.8934	0.8973
RBF_{χ^2}.logTF	**0.9201**	**0.9157**	**0.9179**
Sigmoid.logTF	0.8031	0.8358	0.8191
Inverse multiquadric.logTF	0.7967	0.8731	0.8332
Polynomial.ITF	0.8345	0.8360	0.8352
$RBF_{Gaussian}$.ITF	0.8717	0.7956	0.8319
$RBF_{Laplacian}$.ITF	0.7930	0.7904	0.7917
RBF_{χ^2}.ITF	0.8989	0.8930	0.8959
Sigmoid.ITF	0.7689	0.7948	0.7816
Inverse multiquadric.ITF	0.7947	0.7879	0.7913
Polynomial.IDF	0.6831	0.7678	0.7230
$RBF_{Gaussian}$.IDF	0.7781	0.7645	0.7712
$RBF_{Laplacian}$.IDF	0.7467	0.8012	0.7730
RBF_{χ^2}.IDF	0.7189	0.7370	0.7278
Sigmoid.IDF	0.7480	0.7089	0.7279
Inverse multiquadric.IDF	0.7278	0.6978	0.7125
Polynomial.TF-IDF	0.8540	0.8230	0.8382
$RBF_{Gaussian}$.TF -IDF	0.8356	0.8345	0.8350
$RBF_{Laplacian}$.TF-IDF	0.8321	0.8890	0.8596
RBF_{χ^2}.TF-IDF	0.8163	0.7925	0.8042
Sigmoid.TF-IDF	0.7969	0.8310	0.8136
Inverse multiquadric.TF-IDF	0.7956	0.8106	0.8030
Polynomial.logTF-IDF	0.7416	0.7520	0.7468
$RBF_{Gaussian}$.logTF-IDF	0.7260	0.7689	0.7468
$RBF_{Laplacian}$.logTF-IDF	0.8269	0.8659	0.8460
RBF_{χ^2}.logTF-IDF	0.8166	0.8907	0.8520
Sigmoid.logTF-IDF	0.8133	0.7950	0.8040
Inverse multiquadric.logTF-IDF	0.7891	0.7745	0.7817

Table 3.3: The performance of TSVM spam filtering on trec05-1

for training and the remaining 32189 emails for testing from trec05p-1 data set. Emails, in training and testing processes, have been represented as a stream of chronological order, as the classifier has the choice to request the label each time. For the parameter λ used in Eq. 11, we set its value to 0.6.

As expected, the best results were obtained using string kernels and in particular SSK and Spectrum kernel. Compared to the best performance of Online SVM (94.86%) and Active Online SVM (96.59%) the latter illustrates improved performance (see Table 3.8). One can note a slight reduction for RBF_{χ^2} in term of precision, yet it has reported the best results in Online Active learning. In general, we can see the improvement of distance based kernels in

Kernel	Precision	Recall	F1
Polynomial.TF	0.7598	0.8000	0.7794
$RBF_{Gaussian}$.TF	0.8506	0.8489	0.8497
$RBF_{Laplacian}$.TF	0.9090	0.8644	0.8861
RBF_{χ^2}.TF	0.8739	0.8801	0.8770
Sigmoid.TF	0.8960	0.8032	0.8471
Inverse multiquadric.TF	0.7707	0.8103	0.7900
Polynomial.logTF	0.8367	0.8187	0.8276
$RBF_{Gaussian}$.logTF	0.8289	0.8401	0.8345
$RBF_{Laplacian}$.logTF	0.9010	0.8705	0.8855
RBF_{χ^2}.logTF	**0.9190**	**0.8937**	**0.9062**
Sigmoid.logTF	0.8001	0.8401	0.8196
Inverse multiquadric.logTF	0.7697	0.8555	0.8103
Polynomial.ITF	0.8178	0.8109	0.8143
$RBF_{Gaussian}$.ITF	0.8571	0.8095	0.8326
$RBF_{Laplacian}$.ITF	0.7693	0.7445	0.7567
RBF_{χ^2}.ITF	0.8819	0.8636	0.8727
Sigmoid.ITF	0.7871	0.7801	0.7836
Inverse multiquadric.ITF	0.8049	0.7799	0.7922
Polynomial.IDF	0.6988	0.7709	0.7331
$RBF_{Gaussian}$.IDF	0.7513	0.7534	0.7523
$RBF_{Laplacian}$.IDF	0.7259	0.8105	0.7659
RBF_{χ^2}.IDF	0.7164	0.7074	0.7119
Sigmoid.IDF	0.7355	0.6808	0.7071
Inverse multiquadric.IDF	0.7270	0.6901	0.7081
Polynomial.TF-IDF	0.8489	0.8099	0.8289
$RBF_{Gaussian}$.TF -IDF	0.8263	0.8104	0.8183
$RBF_{Laplacian}$.TF-IDF	0.8239	0.8745	0.8484
RBF_{χ^2}.TF-IDF	0.8096	0.7806	0.7948
Sigmoid.TF-IDF	0.7858	0.8321	0.8083
Inverse multiquadric.TF-IDF	0.7865	0.7913	0.7889
Polynomial.logTF-IDF	0.7391	0.7403	0.7397
$RBF_{Gaussian}$.logTF-IDF	0.7301	0.7598	0.7447
$RBF_{Laplacian}$.logTF-IDF	0.8196	0.8405	0.8299
RBF_{χ^2}.logTF-IDF	0.8023	0.8900	0.8439
Sigmoid.logTF-IDF	0.8201	0.7866	0.8030
Inverse multiquadric.logTF-IDF	0.7756	0.7601	0.7678

Table 3.4: The performance of TSVM spam filtering on trec05-1, where unlabeled training emails are from trec06 data set

Kernel	Precision	Recall	F1
SSk	0.9509	**0.9531**	**0.9520**
Spectrum	0.9657	0.9345	0.9498
Mismatch	0.8967	0.9277	0.9119
Wildcard	**0.9689**	0.9056	0.9362
Gappy	0.9678	0.9012	0.9333
WD	0.9571	0.9100	0.9330
WDs	0.9124	0.9080	0.9102

Table 3.5: The performance of TSVM spam filtering on trec05-1 using string kernels

Kernel	Precision	Recall	F1
SSk	0.9507	**0.9505**	**0.9506**
Spectrum	0.9650	0.9205	0.9422
Mismatch	0.8966	0.9170	0.9067
Wildcard	0.9501	0.8946	0.9215
Gappy	**0.9656**	0.9000	0.9316
WD	0.9570	0.9099	0.9329
WDs	0.9105	0.8909	0.9006

Table 3.6: The performance of TSVM spam filtering on trec05-1 using string kernels, where unlabeled training emails are from trec06 data set

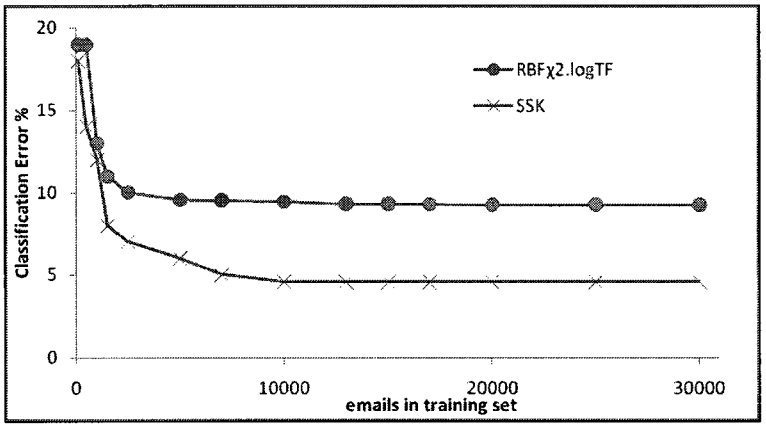

Figure 3.2: Classification Error on the trec05p-1 for RBF_{χ^2}.logTF and SSk kernels by varying the number of labeled emails in training for TSVM

online active model (see Table 3.7).

It should be noted that online active filter can also be beneficial from a computational complexity viewpoint, since the number of labels requested tends to decrease over time. It is evident that online active SVM overwhelmingly outperforms online SVM and TSVM on the trec05p-1 data set. In particular, Online Active SVM exhibits excellent performance using string kernels. However, string kernels are significantly inferior to distance-based kernels in term of computational cost. Tables 3.9 and 3.10 illustrates an improvement for both classes in time computations.

Conclusion our experimental results suggest that online active learning is able to yield a sizeable improvement in performance. For instance, string kernels, in particular SSK, yields improved performance compared to batch supervised learning, with reduced number of labels and reasonable computational time. These results are very encouraging for spam filtering where labeled data are costly while unlabeled data are easy to obtain. In addition, results show a clear

Kernel	Precision	Recall	F1
Polynomial.TF	0.7978	0.8101	0.8039
$RBF_{Gaussian}$.TF	0.8696	0.8579	0.8637
$RBF_{Laplacian}$.TF	0.9041	0.8678	0.8856
RBF_{χ^2}.TF	0.8911	0.8801	0.8856
Sigmoid.TF	0.8759	0.8204	0.8472
Inverse multiquadric.TF	0.7976	0.8678	0.8312
Polynomial.logTF	0.8501	0.8423	0.8462
$RBF_{Gaussian}$.logTF	0.8325	0.8534	0.8428
$RBF_{Laplacian}$.logTF	**0.9100**	0.8898	0.8998
RBF_{χ^2}.logTF	0.9034	**0.9167**	**0.9100**
Sigmoid.logTF	0.8145	0.8404	0.8272
Inverse multiquadric.logTF	0.7893	0.8821	0.8331
Polynomial.ITF	0.8306	0.8453	0.8379
$RBF_{Gaussian}$.ITF	0.8890	0.8011	0.8428
$RBF_{Laplacian}$.ITF	0.8356	0.7896	0.8119
RBF_{χ^2}.ITF	0.9054	0.8831	0.8941
Sigmoid.ITF	0.7965	0.7834	0.7899
Inverse multiquadric.ITF	0.7997	0.7951	0.7974
Polynomial.IDF	0.6976	0.7781	0.7357
$RBF_{Gaussian}$.IDF	0.7591	0.7798	0.7693
$RBF_{Laplacian}$.IDF	0.7598	0.8120	0.7850
RBF_{χ^2}.IDF	0.7290	0.7589	0.7436
Sigmoid.IDF	0.7987	0.7143	0.7541
Inverse multiquadric.IDF	0.7177	0.7328	0.7252
Polynomial.TF-IDF	0.8497	0.8345	0.8420
$RBF_{Gaussian}$.TF -IDF	0.8419	0.8459	0.8439
$RBF_{Laplacian}$.TF-IDF	0.8201	0.8977	0.8571
RBF_{χ^2}.TF-IDF	0.8178	0.7956	0.8065
Sigmoid.TF-IDF	0.7901	0.8422	0.8153
Inverse multiquadric.TF-IDF	0.7901	0.8200	0.8048
Polynomial.logTF-IDF	0.7678	0.7690	0.7684
$RBF_{Gaussian}$.logTF-IDF	0.7580	0.7731	0.7655
$RBF_{Laplacian}$.logTF-IDF	0.8308	0.8662	0.8481
RBF_{χ^2}.logTF-IDF	0.8201	0.8923	0.8547
Sigmoid.logTF-IDF	0.8324	0.7869	0.8090
Inverse multiquadric.logTF-IDF	0.7793	0.7856	0.7824

Table 3.7: The performance of Online Active SVM spam filtering on trec05-1

Kernel	Precision	Recall	F1
SSk	0.9590	**0.9729**	**0.9659**
Spectrum	0.9800	0.9479	0.9637
Mismatch	0.9033	0.9265	0.9148
Wildcard	0.9900	0.9112	0.9490
Gappy	**0.9943**	0.9043	0.9472
WD	0.9700	0.9190	0.9438
WDs	0.9167	0.9088	0.9127

Table 3.8: The performance of Online Active SVM spam filtering on trec05-1 using string kernels

	Polynomail	$RBF_{Gaussian}$	$RBF_{Laplacian}$	RBF_{χ^2}	Sigmoid	Inverse multiquadric
TF	0.03	0.30	2.40	0.38	0.10	3.10
logTF	0.40	1.10	4.44	5.00	6.00	2.45
ITF	1.35	1.55	**6.30**	4.55	3.03	4.00
IDF	0.46	2.00	2.00	4.50	3.01	4.05
TF-IDF	0.02	0.37	3.00	0.34	0.53	3.10
logTF-IDF	0.40	1.10	3.25	5.00	5.59	2.55

Table 3.9: Training time for different combinations of frequency and distance kernels

SSk	Spectrum	Mismatch	Wildcard	Gappy	WD	WDs
17.37	18.55	19.30	20.20	**19.32**	19.10	20.00

Table 3.10: Training time for string kernels

dominance of online active learning methods, compared to both Online SVM and TSVM.

CHAPTER 4

Conclusions and Future work

In this thesis, we described the use of string kernels in order to improve spam filter performance. We implemented, tested, integrated various preprocessing algorithms based on term frequency, importance weight with normalization to investigate their impact on classifier performance. Moreover, we applied algorithms to adapt batch theoretical models to online real world models using string kernels and well-performed preprocessing combinations, and hence maximize the overall performance. Furthermore, we applied typical evaluation criteria such as precision, recall, F1, and computational cost to test the effectiveness of potential solutions.

In chapter two, we gathered legtimate and spam emails and encoded each as a training examples using Bag of Words for distance-based kernels and k-mers for string kernels. For feature mapping, extracted features using BoW approach were weighted using one of already established weighting schemes in TC. It was found that each frequency transformation has an effect on the performance; where applying some frequency transformation the performance reaches a comparable values as using TF, logTF , while with others it reaches lower performance such as IDF. Results suggest that deletion of stop words is not necessary when using SVM to classify emails. We also evaluated how well SVM classified emails based on used kernels. The kernels that consistently perform well and tend to produce the most powerful classifier are the RBF kernels, in particular, RBF_{χ^2} and unaware position string kernels (for instance, spectrum kernel and SSk). After extensive experiments we can say that string kernels offer an excellent alternative discriminative approach for spam filtering. However, its computational costs are higher than any of other distance-based kernels, with 10 minutes as best case.

In chapter three, several algorithms were employed to support the evaluation process of

string kernels and preprocessing algorithms and to reproduce effective potential spam filter. For fair comparison, we used the same appropriate preprocessing for kernels on the same data sets. We modeled three designs: online SVM, TSVM, and Active online SVM. Online SVM supplies a real world environment where data come in stream one-by-one, human effort in labeling emails was reduced using Transductive and active models. To cope for real scenarios we implemented and tested active online model. Ultimately, active online SVM outperformed other algorithms using string kernels.

Spam filtering solutions presented in this thesis generates acceptable, accurate results, but further enhancement can be made by taking into account user feedback. Moreover, email content is richer than text, it has images, attachment, links, routing and meta information. Consequently, classifier might be improved if we consider such information.

List of References

[1] SPAMHAUS. The spam definition and legalization game. *Available at http://www.spamhaus.org/news.lasso?article=9,Accessed: 31.05.06,* 2003.

[2] C. Drake, J. Oliver, and E. Koontz. Anatomy of a phishing email. In Proceedings of the *First Conference on Email and Anti-Spam (CEAS)*, California, USA, 2004.

[3] N. Lugaresi. European union vs. spam: A legal response. In Proceedings of the *First Conference on Email and Anti-Spam (CEAS)*, California, USA, 2004.

[4] P. Denning. Electronic junk. *Communication of the ACM* 25 (3): 163 - 165, 1982.

[5] W. Cukier, S. Cody, and E. Nesselroth. Genres of spam: Expectations and deceptions. In Proceedings of the *39th Annual Hawaii International Conference on System Sciences*, volume 3, Hawaii, USA, 2006.

[6] G. Hulten, J. Goodman. Tutorial on Junk Mail Filtering. Microsoft Research. *Available at: http://research.microsoft.com/ joshuago/tutorialOnJunkMailFilteringjune4.pdf.*

[7] L. F. Cranor and B. A. LaMacchia. Spam!. *Communications of the ACM*, 41 (8): 74-83, 1998.

[8] D. Nagamalai, C. Dhinakaran, and J. K. Lee. Multi Layer Approach to Defend DDoS Attacks Caused by Spam. In Proceedings of the *International Conference on Multimedia and Ubiquitous Engineering*, pp. 97-102, Washington, DC, USA, 2007.

[9] C. Cortes and V. Vapnik. Support-Vector Networks. *Machine Learning* 20 (1): 273-297, 1995.

[10] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In Proceedings of *10th European Conference on Machine Learning*, LNCS 1398, pp. 137-142, Springer-Verlag, 1998.

[11] A. Kolcz, and J. Alspector. SVM-based filtering of e-mail spam with content-specific misclassification costs. In Proceedings of the *Workshop on Text Mining*, pp. 123-130, California, USA, 2001.

[12] D. Sculley and G. Wachman. Relaxed online SVMs for spam filtering. In Proceedings of the *30th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 415 - 422, Amsterdam, Netherlands, 2007.

[13] G. Schohn and D. Cohn. Less is more: Active learning with support vector machines. In Proceedings of the *Seventeenth International Conference on Machine Learning*, pp. 839-846, California, USA, 2000.

[14] G. V. Cormack and A. Bratko. Batch and on-line spam filter comparison. In Proceedings of the *Third Conference on Email and Anti-Spam*, California, USA, 2006.

[15] X. Carreras, and L. Marquez. Boosting trees for anti-spam email filtering. In Proceedings of the *4th International Conference on Recent Advances in Natural Language Processing*, pp. 58-64, Bulgaria, 2001.

[16] J. Rocchio. Relevance feedback in information Retrieval. In Proceedings of the *SMART Retrieval System: Expriments in Automatic Document Processing*, pp. 313-323, New Jersey, USA, 1971.

[17] I. Androutsopoulos, J. Koutsias, K. Chandrinos, G. Paliouras, and C. Spyropoulos. An evaluation of naive Bayesian anti-spam filtering. In Proceedings of the *11th European Conference on Machine Learning*, pp. 9-17, Barcelona, Spain, 2000.

[18] B. Scholkopf and A. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*, MIT Press, 2001.

[19] H. Drucker, V. Vapnik, and D. Wu. Support vector machines for spam categorization. *IEEE Transactions on Neural Networks*, 10 (5): 1048-1054, 1999.

[20] T. Joachims. Making large-scale support vector machine learning practical. In *Advances in Kernel Methods: Support Vector Machines*, pp. 169-184, 1998.

[21] T. Fawcett. ROC Graphs: Notes and Practical Considerations for Researchers. *Technical report*, Palo Alto, USA, HP Laboratories, 2004.

[22] J. Goodman. Spam: Technologies and Policies. Microsoft Research. *http://www.research.microsoft.com/ joshuago/spamtech.pdf*, 2003.

[23] I. Androutsopoulos, G. Paliouras, V. Karkaletsis, G. Sakkis, C. Spyropoulos, and P. Stamatopoulos. Learning to filter spam e-mail: A comparison of a naive bayesian and a memory-based approach. In Proceedings of the *Workshop on Machine Learning and Textual Information Access, 4th European Conference on Principles and Practice of Knowledge Discovery in Databases*, pp. 1-13, Lyon, France, 2000.

[24] A. Karatzoglou, D. Meyer, K. Hornik. Support Vector Machines in R. *Journal of Statistical Software*, 15 (9): 1-28, 2006.

[25] C. Dwork, and M. Naor. Pricing via Processing Or Combatting Junk Mail. In *12th Annual International Cryptology Conference on Advances in Cryptology*, LNCS 740, pp. 139-147, Springer, 1993.

[26] M. Abadi, M. Burrows, M. Manasse, and T. Wobber. Moderately Hard, Memory-Bound Functions. In Proceedings of the *10th Annual Network and Distributed System Security Symposium*, pp. 25-39, California, USA, 2003.

[27] A. Back. HashCash - A Denial of Service Counter-Measure. *Available at: http://cypherspace.org/hashcash/hashcash/.pdf.*

[28] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz. A Bayesian Approach to Filtering Junk E-mail. In Proceedings of the *Learning for Text Categorization Workshop*, Wisconsin, USA, 1998.

[29] B. Gates, N. Myhrvold, and P. Rinearson. *The Road Ahead.* Penguin Group Incorporated, 1996.

[30] V. Vapnik. An Overview of Statistical Learning Theory. *IEEE Transactions on Neural Networks*, 10 (5): 988-999, 1999.

[31] E. Blanzieri and A. Bryl. A Survey of Learning-Based Techniques of Email Spam Filtering. *Technical Report*, Trento, Italy, University of Trento, 2006.

[32] R. Courant and D. Hilbert. *Methods of Mathmatical Physics*. Wiley Interscience, 1953.

[33] T. Anderson and R. Bahadur. Classification into two Multivariate Normal Distributions with Different Covariance Matrices. In *The Annals of Mathematical Statistics*, 33 (2): 420-431, 1962.

[34] N. Cristianini and J. Shawe-taylor. *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge University Press, 2000.

[35] J. burges. A tutorial on support vector machines for pattern recognition. In *Data Mining and Knowledge Discovery*, 2 (2): 121-167, 1998.

[36] B. Scholkopf. The Kernel Trick for Distances. In Proceedings of the *Advances in Neural Information Processing Systems (NIPS)*, pp. 301-307, Colorado, USA, 2000.

[37] C. Berg, J. Christensen, and P. Ressel. *Harmonic Analysis on Semigroups: Theory of Positive Definite and Related Functions*, Springer-Verlag, 1984.

[38] I. J. Schoenberg. Metric Spaces and Positive Definite Functions. *Transactions of the American Mathematical Society*, 44 (3): 522-536, 1938.

[39] H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins. Text Classification using String Kernels. *The Journal of Machine Learning Research*, 2 (1): 419-444, 2002.

[40] G. Salton. Mathematics and information retrival. *Journal of Documentation*, 35 (1): 1-29, 1979.

[41] E. Leopold, and J. Kindermann. Text Categorization with Support Vector Machines. How to Represent Texts in Input Space?. *Machine Learning*, 46 (13): 423-444, 2002.

[42] G. Cormack and T. Lynam. Spam corpus creation for TREC. In Proceedings of the *Second Conference on Email and Anti-Spam (CEAS)*, California, USA, 2005.

[43] G. K. Zipf. *Human Behavior and the Principle of Least Effort*. Addison-Wesley, 1949.

[44] C. Leslie, E. Eskin, and W. S. Noble. The Spectrum Kernel: A String Kernel For SVM Protein Classification. In Proceedings of the *Pacific Symposium on Biocomputing*, pp. 564-575, Hawaii, USA, 2002.

[45] B. Vanschoenwinkle. A discrete Kernel Approach to Support Vector Machine Learning in Language Independent Named Entity Recognition. In Proceedings of the *Annual Machine Learning Conference*, pp. 154-161, Netherlands, 2004.

[46] D. Zhang and W. Sun lee. Extracting Key-substring-Group Features for Text Classfication. In Proceedings of the *12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 474-483, Pennsylvania, USA, 2006.

[47] C. Laslie, E. Eskin, J. Weston, and W. S. Noble. Mismatch string kernels for SVM protein classification. In Proceedings of the *Advances in Neural Information Processing Systems (NIPS)*, pp. 1441-1448, Massachusetts, USA, 2002.

[48] G. Rtsch and S. Sonnenburg. Accurate Splice Site Detection for Caenorhabditis elegans. In *Kernel Methods in Computational Biology*, MIT Press, 2004.

[49] G. Rtsch, S. Sonnenburg, B. Schlkopf. RASE: Recognition of Alternatively Spliced Exons in C. elegans. *Bioinformatics*, 21 (1): i369-i377, 2005.

[50] M.F. Porter. An algorithm for suffix stripping. *Program*, 14 (3): 130-137, 1980.

[51] G.L. Wittel and S.F. Wu. On attacking statistical spam filters. In Proceedings of the *First Conference on Email and Anti-Spam, CEAS*, California, USA, 2004.

[52] C. Leslie and R. Kuang. Fast string kernels using inexact matching for protein sequences. *Journal of Machine Learning Research*, 5: 1435 - 1455, 2004.

[53] N. J. Belkin and W. B. Croft. Information filtering and information retrieval: two sides of the same coin?. *Communications of the ACM*, 35 (12): 29 - 38, 1992.

[54] F. Debole and F. Sebastiani. Supervised Term Weighting for Automated Text Categorization. In Proceedings of the *ACM symposium on Applied computing*, pp. 784-788, Florida, USA, 2003.

[55] F. Sebastiani. Machine Learning in Automated Text Categorization. *ACM Computing Surveys*, 34 (1): 1-47, 2002.

[56] D. D. Lewis. An evaluation of phrasal and clustered representations on a text categorization task. In Proceedings of the *Annual ACM Conference on Research and Development in Information Retrieval*, pp. 37-50, Copenhagen, Denmark, 1992.

[57] M. F. Caropreso, S. Matwin, and F. Sebastiani. A learner-independent evaluation of the usefulness of statistical phrases for automated text categorization. In *Text Databases and Document Management: Theory and Practice*, pp. 78-102, IGI Publishing, 2001.

[58] M. Szummer and T. Jaakkola. Information regularization with partially labeled data. In Proceedings of the *Advances in Neural Information Processing Systems (NIPS)*, British Columbia, Canada, 2003.

[59] V. Vapnik. *Statistical Learning Theory*. Wiley, 1998.

[60] J. Wang and X. Shen. Large Margin Semi-supervised Learning. *The Journal of Machine Learning Research*, 8 (1): 1867-1891, 2006.

[61] X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In Proceedings of the *Twentieth International Conference on Machine Learning (ICML)*, pp. 912-919, Washington, DC, USA, 2003.

[62] C. Xu and Y. Zhou. Transductive Support Vector Machine for Personal Inboxes Spam Categorization. In Proceedings of the *International Conference on Computational Intelligence and Security Workshops*, pp. 459-463, Washington, DC, USA, 2007.

[63] K. Brinker. Incorporating diversity in active learning with support vector machines. In Proceedings of the *Twentieth International Conference on Machine Learning*, pp. 59 -66, 2003.

[64] N. Roy and A. McCallum. Toward optimal active learning through sampling estimation of error reduction. In Proceedings of the *Eighteenth International Conference on Machine Learning*, pp. 441 -448, 2001.

[65] D. Sculley. Online active learning methods for fast label-efficient spam filtering. In Proceedings of the *Fourth Conference on Email and Anti-Spam (CEAS 2007)*, Berlin, Germany, 2007.

[66] G. Cauwenberghs and T. Poggio. Incremental and decremental support vector machine learning. In Proceedings of the *Neural Information Processing Systems (NIPS)*, pp. 409-415, 2000.

[67] J. Kivinen, A. Smola and R. Williamson. Online learning with kernels. *IEEE Transactions on Signal Processing*, 52 (8): 2165- 2176, 2004.

[68] S. Riiping. Incremental learning with support vector machines. Techn. Report TR-18, Universitat Dortmund, SFB475, 2002.

[69] T. Joachims. Transductive Inference for Text Classification using Support Vector Machines. In Proceedings of the *sixteenth International Conference on Machine Learning (ICML-99)*, pp. 200-209, San Francisco, US, 1999.

[70] N. Kasabov and S. Pang. Transductive Support Vector Machines and Applications in Bioinformatics for promoter Recognition. *Neural Information Processing*, 3 (2): 31-38, 2004.

[71] E. Y. Chang, S. Tong, K. Goh and C. Chang. Support Vector Machine Concept-Dependent Active Learning for Image Retrieval. In Proceedings of the *ACM International Conference on Multimedia*, pp. 107-118, 2001

[72] D. Cohn, L. Atlas, and R. Ladner. Improving generalization with active learning. *Machine Learning*, 15 (2): 201 -221, 1994.

[73] *Available at : http://spamassassin.apache.org/tests32x.html,* 2008.

[74] P. Graham. A plan for spam. *Available at http://www.paulgraham.com/spam.html,* 2002.

[75] G. Salton and M. J. McGill. *An Introduction to Modern Information Retrieval*, McGraw-Hill, 1983.

[76] G. V. Cormack and T. R. Lynam. TREC 2005 spam track overview. In Proceedings of the *Fourteenth Text REtrieval Conference (TREC05)*, Gaithersburg MD, 2005.

[77] E. Fredkin. Trie Memory. *Communications of the ACM*, 3 (9): 490-499, 1960.

[78] O. Chapelle, V. Sindhwani and S. S. Keerthi. Optimization Techniques for Semi-Supervised Support Vector Machine. *The journal of Machine learning Research* 9: 203-233, 2008.

[79] G. V. Cormack and A. Bratko. Batch and on-line spam filter comparison. In Proceedings of the *Third Conference on Email and Anti-Spam, (CEAS)*, Mountain View, CA, 2006.

[80] K. W. Lau and Q. H. Wu. Online training of support vector machine. *Pattern Recognition*, 36 (8): 1913-1920, 2003.