

Spam or Ham

Michael Fong
Dept of Computer Science
mcfong@iastate.edu

ComS 572 - Introduction to Artificial Intelligence Project

December 12, 2008

Contents

1	The SPAM Phenomenon	3
1.1	Source of Spam	3
1.2	Type of Spam	4
2	Spam Control	5
2.1	Learning Based Spam Filtering	6
2.1.1	Naive Bayesian Classifier	6
2.1.2	K-Nearest Neighbor Algorithm	7
2.1.3	Support Vector Machine	7
2.2	Feature Selection	8
3	Adversarial Activity	9
3.1	Remedies	9
4	Experiment	12
4.1	Data-Set	12
4.2	Flow of Preprocess Work	12
4.3	Required Libraries	13
5	Classification Result	14
5.1	Text Classification (Subject plus Body)	14
5.1.1	Naive Bayes	15
5.1.2	Support Vector Machine	15
5.2	URL Classification (Email address plus URL address)	15

5.2.1	Naive Bayes	16
5.2.2	K Nearest Neighbor	16
6	Conclusion and Future Work	16

Abstract

SPAM was originally abbreviated for "Self Promotional Advertising Message", but now the most popular definition characters as "unsolicited bulk mail". [1]. Email spam especially is the major problems, bringing an increase in network bandwidth and financial damage to individual users. Among the approaches developed to stop spam, filtering is an important and popular one. In this project, we give several state-of-the-art machine learning approaches, and also a brief description of other branches of anti-spam protection.

1 The SPAM Phenomenon

The problem of undesired electronic messages is nowadays a serious issue. As spam constitutes more than 70% of total amount of email messages worldwide, and this crisis is projected to increase up to 80% by 2010, predicted by Trend corp[2]. Spam makes users not only waste their time, but cause loss of work productivity, and eventually lead to enormous social cost.

Imagine that if spammers sends 10,000 messages suggesting to buy their product, and only one person follows the lead. Then, they could made a small profit, however, a pure profit. Now, why not send out 10,000,000 messages and make a 100 times bigger profit. There are many spammers out there, all wanting to make a big profit. The result is as of estimated 70+% of all email traffic is occupied by undesired email.

1.1 Source of Spam

- BotNet

It was relatively easy to block spammers by checking from a handful of IPs, such as blacklist or whitelist mechanism, as spammers were getting blocked as soon as they start sending spam. However, the new threat used by spammers is the botnets. Bonet is a cluster of computers that were broken into by a person or via a virus attack (usually mounted via email). A program is installed on all these computers, which now the distributor, or often is called bot master, controls these computers remotely - those machines are usually referred to as zombies. Now a spammer who wants to send a lot of spam does not need to send it from one machine, but they would simply distribute the attack via thousands of computers, which now makes it really hard to detect, based on the volume of email coming from the same IP,

since each zombie sends just one or a few messages to the same target.

- Directory Harvest Attacks

Spammers collect the email addresses mainly by crawling the Internet and picking up mailing addresses from articles, forums, mailing list archives, etc. Some people try to obfuscate their addresses when they put it online, for instance, 'username at domain dot org', which is still circumvented by those crawlers since most people use the same obfuscation technique. The best way to protect your address to be harvested is probably to put it in an image file. Unless you never email anybody, there is no really a way to keep your address from the praying eyes of the spammers.

The more recent trend for harvesting email addresses is the DHA (Directory Harvest Attack). In most cases, many people use common names or names of common objects as their usernames, so that spammers could take a dictionary of common words and try to use those, or its various combinations as usernames, all starting with the same prefix. Nevertheless, the SMTP protocol has a special command 'VRFY', which was designed to help the senders whether the username exists before attempting to send an email. However, this is exactly the wonderful tool for spammer to perform the directory harvesting attack.

- Internet Hoaxes and Chain Letters

There are usually many hoaxes (fake stories) or chain letters (other fake stories) flowing in the internet. Those letters sometimes suggest that you will make a fortune if you forward it to at least 50 people, or sometimes tell a heart breaking story, urging you to forward the story to your friends. However, a few people actually do not realize this helps providing spammers with fodder for their activity. The spammer can use the same DHA trick mentioned above to collect massive of email address from these kind of chain emails. The reason is average people usually forward the email using CC email header (and not BCC header), which cause the email address of everyone you are sending to being exposed to others.

1.2 Type of Spam

- Sales-pitch (advertisement)

Unsurprisingly, spam is mostly sales advertisement (which suggests you to buy their certain product).

- Phishing mails (internet-scam)

Phishing is a relatively new phenomenon, which allows scammers to fool thousands of people at once. A typical phishing email contains a message which seems to be very similar to the usual email you get from your bank, where the difference is, however, they try to make you give them your bank credentials. The typical one is you are told that your account was hacked and they need your login/password to fix it. Once you naively give them your credentials, they go and take your money.

- Email Scam

A typical scam suggests that someone has a huge amount of money in some third world country. They beg you to help them transfer the money out of that country, and offer you a huge cash reward. If you respond, they will ask for a relatively little amount of money so that they can open a new account on your name and where they would transfer the money to. If you ever do that, say bye bye to your money. Neither you will get the promised bounty, nor you will ever again see your money. ebay is a very common place to get scammed. Especially from those vendors who have hundreds of "excellent vendor" comments and votes, many of those are fake users created to inflate the reputation of the scam artist. Hover, if someone reports it as a scam, it will disappear in the sea of positive votes.

2 Spam Control

One of the proposed ways of stopping spam is to enhance or even substitute the existing standards of email transmission by new, spam-proof variants. The main drawback of the commonly used Simple Mail Transfer Protocol (SMTP) is that it provides no reliable mechanism of checking the identity of the message source. The approach is to use simple tests that allow the system to distinguish human senders from robots by asking the user to answer a moderately easy question before granting him the permission to sent a message ¹. Another proposal [3] was to establish a small payment for sending an email message, neglectable for a common user, but big enough to prevent a spammer to broadcast millions of messages. An interesting version of this approach is Zmail protocol ², where a small fee is paid by the sender to the receiver, so that a common user who sends and receives

¹The Captcha project <http://www.captcha.net>

²ZMail - Free Secure Email: <http://free-secure-email.com/default.aspx>

nearly equal amount of messages gets neither damage no profit from using email, while spamming becomes a costly operation.

2.1 Learning Based Spam Filtering

The idea of automated methods for filtering such spam from legitimate Emails is becoming necessary. The automatic classification can be done with machine learning techniques. In general, such filter needs a collection of labelled trained data with pre-collected reliable judgments, so that a spam filter can classify a new message to be a spam or non-spam, based on these previous experience. In practice, many tools have been developed to help users organize their mailbox, like Hotmail's³ "Junk Mail" folder. Emails in these folders were classified as spam, by built-in classifiers, and used as training sample for future classification.

2.1.1 Naive Bayesian Classifier

In 1998 the Naive Bayes classifier was proposed for spam recognition [4, 5]. This new trend classifier became widely known and used due to Paul Graham's popular article "A Plan for Spam" [1]. This classifier, when applied to text, can be considered an improved learning-based variant of keyword filtering. From Bayes theorem, a document d , represented with feature vector $\vec{x} = \langle x_1, \dots, x_n \rangle$, belongs to category c is:

$$\begin{aligned}
 c_{NB} &= \operatorname{argmax}_c P(c|x_1, \dots, x_n) \\
 &= \operatorname{argmax}_c P(x_1, \dots, x_n|c) \cdot P(c) && \text{(Baye's Theorem)} \\
 &= \operatorname{argmax}_c \prod_1^n P(x_i|c) \cdot P(c) \\
 &&& \text{(independence assumption of NB)}
 \end{aligned}$$

$$\forall c \in \{spam, leg\}$$

, where c_{NB} denotes the target value output by the Naive Bayesian Classifier.

The Naive Bayesian classifier assumes that all the features are statistically independent, that is, the probability of observing the conjunction x_1, \dots, x_n , $P(x_1, \dots, x_n)$ is just the product of the probabilities for individual attributes. In practice, $P(c)$ are easy to estimate from the frequencies in which category c occurs in the training corpus.

³<http://www.hotmail.com>

$P(x_i|c)$ means given a category c , the probability of feature x_i occurs in the document. Typically, this conditional probability is represented with document frequency (or TFIDF), where each word is treated as a feature. However, the drawback of this classifier is that X_1, \dots, X_n are conditionally independent is usually overly simplistic in practice.

Naive Bayesian Classifier is an easy to understand an easy to implement classification techniques. Despite of its impractical independence assumption, this classifier often performs surprisingly effective due to its simplicity, elegance, and robustness. Thus, it is widely used in areas of text classification, such as spam filtering.

2.1.2 K-Nearest Neighbor Algorithm

The k-Nearest Neighbor (k-NN) classifier was proposed for spam filtering by Androutsopoulos et al. [6]. With this classifies the decision is made as follows: the algorithm assigns to each new unseen instance class from the majority vote among the k training instances that are closest to the unseen instance. Euclidean Distance is typically used to calculate similarity between two given instances. For instance, $x_i = \langle x_{i1}, x_{i2}, \dots, x_{in} \rangle$ and $x_j = \langle x_{j1}, x_{j2}, \dots, x_{jn} \rangle$, their distance is:

$$d(x_i, x_j) = \sqrt{\sum_{k=1}^n (x_{ik} - x_{jk})^2}$$

However, the choice of k is one of the key issues that affect the performance of this algorithm. If k is too small, then the result can be sensitive to noise points; if k is too large, then the neighborhood may include too many points from other classes. In general, k-NN is very simple to understand and implement. Nevertheless, the drawback is that k-NN is very sensitive to redundant features and noise data, so we might be able to improve the accuracy, if we adopt some noise reduction techniques for k-NN.

2.1.3 Support Vector Machine

Another classifier proposed for spam filtering is Support Vector Machine (SVM), which was proposed by Vapnik [7] in 90's, and ever since many scientists have been working on related research about SVM on text classification. Given the training samples and a predefined transformation $\phi : R^d \rightarrow F$, which maps the features to a transformed feature space, the classifier separates the samples of the two classes with a hyperplane in the transformed feature space, building a decision

rule of the following form:

$$f(\vec{x}) = \text{sign}\left(\sum_{i=1}^n \alpha_i \cdot y_i \cdot K(\vec{x}_i, \vec{x}) + b\right)$$

where $K(\phi(u), \phi(v)) = \phi(u) \cdot \phi(v)$ is the kernel function and $\alpha_i, i = 1, \dots, n$ and b maximize the margin of the separating hyperplane. The value -1 corresponds to c_{leg} , 1 corresponds to c_{spam} . In other words, SVM aims to find a maximum margin separating hyper-plane between two classes of data set. That is we are given training data to maximize the nearest distance between a point in one separated hyperplane and a point in the other separated hyperplane.

Several remarkable properties of SVM listed below showed its substantial performance in text classification[8]:

1. SVM is able to learn independent of the dimensionality of the feature space.
2. SVM shows efficient performance, even if the size of the feature space is large.
3. Document vectors are usually tended to be sparse.

Nevertheless, the original SVM is designed for binary classification and a direct solution to the multi-class problem using SVM is usually avoided, due to the various complexities. The typical approach is to use a combination different classifying methods (usually accompanied with SVM) to solve a multi-class classification problem.

2.2 Feature Selection

Measure of feature relevance used for ordering features. Each measure applies to a feature, and c_{spam} and c_{leg} are the labels of spam class and legitimate mail class correspondingly. However, the typical way of representing the frequency of a word occurred in the document, might lead to a problem. For frequent terms, such as "the", would have a lot higher occurrence than those for rare terms, like "unsubscribe". The common words appears everywhere in the document, nevertheless, are rather meaningless than the rare but highly suspicious terms. Therefore, we adopted another similar scoring function to weigh each word in the document:

- Term Frequency Inverse Document Frequency (TFIDF)
The name Term Frequency-Inverse Document Frequency (TFIDF) actually applies to a term weighting scheme, which is defined as follows:

$$w_{t,d} = tf_{t,d} \cdot \log \frac{n}{df_t}$$

where $w_{t,d}$ is the weight of term (token) t in the document (message) d ; $tf_{t,d}$ is the number of occurrences of the term t in the document d ; whereas df_t is the number of messages in which the term t occurs, and n , as above, is the total number of documents in the training set.

3 Adversarial Activity

The more spam a user gets, the less likely he will be to notice one innocent mail sitting in his spam folder. And strangely enough, the better your spam filters get, the more dangerous false positives become, because when the filters are highly effective, users will be more likely to ignore everything they catch. The methods of spamming are improving together with the methods of spam filtering. Spammers try to attack filters, namely to decrease filtering effectiveness. We can categorize several common attacks on spam filters as following:

- Tokenization attacks, when the spammer intends to prevent correct tokenization of the message by splitting or modifying features, for example putting extra spaces, or symbols like '-' or '#', in the middle of the words, 'U_N_\$\$U_B_S_C_R_I_B_E', for instance.
- Text Obfuscation attacks, when the content of the message is obscured from the filter.
 - deliberately misspelling certain key words, such as 'Wildcatt', 'weeek', 'sunsit', 'veeery'.
 - transposing letters or by substituting letters with lookalike numbers and symbols, for example, 'drgs' or 'UN\$UBB\$CRIBE'.
 - miss used accented characters are common to appear.

3.1 Remedies

- Text De-obfuscation
Spammers often substitute characters (sometimes called LEET) for other characters that look alike in order to avoid known spamming keywords. For example, if you have a content filter that traps messages containing "viagra" (because of the flood of viagra spasm), all the spammer has to do is spell it as "v1@gr@" or "iagra" or even "viagara" and it will probably sneak past.
- Spell Correction
In September 2003, a PhD student at Cambridge University had discovered that as long as the first and last letters of a word

remained in place , ordering for the others does not matter. For instance, people are able to read the word "sfotwrae" or "sofwtare".

The original text passed around reads as follows:

Aoccdrnig to rscheearch at Cmabrigde uinervtisy, it deosn't mtttaer waht oredr the ltteers in a wrod are, the only iprmoetnt tihng is taht thefrist and lsat ltteres are at the rghit pclae. The rset can be a tatol mses and you can sitll raed it wouthit a porbelm. Tihs is bcuseae we donot raed ervey lteter by it slef but the wrod as a wlohe.

The actual origin of the theory appears to be an unpublished PhD thesis [9].

After the technique became known on the internet, some people have written simple scripts ⁴ that can randomize text excluding the first and last letters.

Nowadays, spammers are constantly searching for new means to obfuscate a word in order to fool spam filters, and this technique often appeared in spam text:

Do you wnat to l00k c00l and w3althy but do not have the m0ney to aff0rd a= sweeeeet n3w R0lex wtach? Get a 98% L00kalike R0lex watc_h here! We have replika_s of all the fines_t bran_d watc_hes. Check the_m out here!

- Edit Distance

The Levenshtein distance is a metric for measuring the amount of difference between two sequences (of characters). The difference between two strings is given by the minimum number of operations needed to transform one string into the other, where an operation is an insertion, deletion, or substitution of a single character.

⁴<http://www.jwz.org/hacks/scrmable.pl>

```

LEVENSHTEIN-DISTANCE(chars[1..m], chart[1..n])
1 Initialize  $d[0..m, 0..n] = 0$ .
2 for  $i \leftarrow 0$  to  $m$ 
3     do  $d[i, 0] = i$ 
4 for  $j \leftarrow 0$  to  $n$ 
5     do  $d[0, j] = j$ 
6 for  $i \leftarrow 1$  to  $m$ 
7     do for  $j \leftarrow 1$  to  $n$ 
8         do if  $s[i] = t[j]$ 
9             then  $cost = 0$ 
10            else  $cost = 1$ 
11             $d[i, j] = \min($ 
12                 $d[i - 1, j] + 1, //$  deletion
13                 $d[i, j - 1] + 1, //$  insertion
14                 $d[i - 1, j - 1] + cost) //$  substitution
15 return  $d[m, n]$ 

```

- Probabilistic Approach

Here are three essential statements that would needs to be addressed:

- $P(w)$, is the probability of a given words w to appear in an English text.
- $P(c)$, is the probability that a proposed correction c stands on its own. This is equivalent to the statement "how likely is c to appear in an English text?" In that sense, $P(\text{"the"})$ would have a relatively high probability, while $P(\text{"aaaaabbbbbccccc"})$ would be near zero.
- $P(w|c)$, is the probability that w would be typed in a text when the author meant c . That is, "how likely is it that the author would type w by mistake when c was intended?"

Given a word w , we are trying to choose the most likely spelling correction for that word . Our goal is to find the correction c , out of all possible corrections, that maximizes the probability of c given the original word w : $P(c|w)$. This is equivalent of saying "What is the most likely spelling correction c , if user types a word w ". (which could mean itself, if no better correction can be found)

$$\operatorname{argmax}_c P(c|w)$$

By Bayes' Theorem this is equivalent to:

$$\begin{aligned}\operatorname{argmax}_c P(c|w) &= \operatorname{argmax}_c \frac{P(w|c) \cdot P(c)}{P(w)} \\ &= \propto \operatorname{argmax}_c P(w|c)P(c)\end{aligned}$$

(Since $P(w)$ is the same for every possible c , we can ignore it)

4 Experiment

4.1 Data-Set

TREC 2006 Public Spam Corpora ⁵

In these experiments, we have used the Spam Corpus provided from the TREC 2006 Spam Filtering Track.

4.2 Flow of Preprocess Work

My Spam filtering architecture works as a generalized form for spam classification. Given an input text we perform the following sequential operations:

- **MIME Normalization**
To unpacking MIME encodings to a common representation. In particular, parsing the MIME Header along with body text is extremely useful. In addition, we check the character set to be most meaningful to us, for instance, a typical English based user, this is base ASCII (also known as Latin1) where accents are not significant.
- **Tokenized Transformation**
We use a regular expression to segment the incoming text into tokens (text strings) and perform following transformation:
 - **Unicode Normalization**
In order to fix accented character, we transforms this accented text into an equivalent composed or decomposed form in normal English.
 - **HTML de-obfuscation**
In some rare cases, HTML is an essential part of the message, but HTML also provides an incredibly rich environment to obscure content from a machine classifier while retaining the content for a human viewer. In particular, spammers often insert nonsense tags to break up otherwise recognizable words, and use font and foreground colors to hide

⁵<http://plg.uwaterloo.ca/~gvcormac/treccorpus06/>

hopefully dis-incriminating keywords. We adopt HTML-Parser (.jar) package to parse the body text out. This package is extremely useful, when we try to fetch URL and email address for later experiment.

- Spell-Correction
From the theory mentioned above, $P(c)$ can be fetched by reading a default dictionary files in training, and enumerate all possible corrections of $\{c_1, c_2, \dots, \}$ for w , the misspelling word, by checking the edit distances between w and the corrected $\{c_1, c_2, \dots, \}$. However, this mechanism is not completely implemented by the time the project is due, and this would be a future improvement for this email filter.
- Feature Extraction
 - The first step in text classification is to convert documents, represented by strings of characters, into a suitable format for the classification task. During the conversion, we determine the key phrases by viewing the occurrence of word in the feature space(vocabulary). We use the StringToWord-Vector filter in WEKA to convert the original article data into a new data set with word frequency for each word. The more frequent a word appears in the data set, the more important the word represents to the document.
 - However, words with high frequency are commonly the daily words we use everyday (like 'a', 'the', etc). Those stop words, also supported by WEKA, should be omitted, and the experiments have shown stop words occupy one-fourth of the whole feature space.
 - Last but not least, we filter numbers, punctuation, and etc to reduce the degree of obfuscation to the classifier.

4.3 Required Libraries

Here are the list of libraries that we have adopted for this project

- HTMLParser(v 1.6) ⁶ is a fast Java package used to parse HTML documents. Primary features include data transformation, data extraction, filters, custom tags, and easy-to-use JavaBeans.
- WEKA (v 3.5.7) ⁷ is a well-developed Java package that supports a collection of machine learning algorithms for data mining tasks.

⁶<http://htmlparser.sourceforge.net/>

⁷<http://www.cs.waikato.ac.nz/ml/weka/>

WEKA supports data pre-processing, classification, regression, clustering, association rules, and visualization.

- JavaMail ⁸ provides a platform-independent and protocol-independent framework to build mail and messaging applications.
- Snowball ⁹ is a popular package to perform stemming process, which reduces inflected (or sometimes derived) words to their stem or base form.
- libSVM ¹⁰ is an integrated software for support vector classification, regression, and distribution estimation (one-class SVM). It also supports multi-class classification.
- WLSVM ¹¹, abbreviated as Weka LibSVM, combines the merits of the WEKA and libSVM. WLSVM can be viewed as an implementation of the LibSVM running under Weka environment.

5 Classification Result

As the messages in this corpus (TREC2006) are in chronological order, we run three major simulations over 500, 1000 and 1800 consecutive samples, where 75% of them are spam, and 25% of Ham. For testing, we perform a 10-cross validation and measure the average accuracy for each run. The detail results of this experiment are listed below.

5.1 Text Classification (Subject plus Body)

In the first experiment, we classify the message (subject plus body text) with Naive Bayes and Support Vector Machine:

⁸<http://java.sun.com/products/javamail/>

⁹<http://snowball.tartarus.org/index.php>

¹⁰<http://www.csie.ntu.edu.tw/~cjlin/libsvm/>

¹¹<http://www.cs.iastate.edu/~yasser/wlsvm/>

5.1.1 Naive Bayes

# of document	# of features	Accuracy
400	3333 (occurrence > 2)	95.8449 %
400	1302 (occurrence > 5)	95.8449 %
800	5445 (occurrence > 2)	91.9668 %
800	2240 (occurrence > 5)	91.9668 %
1200	10533 (occurrence > 2)	85.0734 %
1200	4795 (occurrence > 5)	85.4305 %
1800	11819 (occurrence > 2)	86.1173 %
1800	5281 (occurrence > 5)	86.1173 %

From the result above, obviously, when the number of feature raises, when the number of feature raises, the accuracy eventual drop downs, because of growing number of noise in data.

5.1.2 Support Vector Machine

We adapt WLSVM, known as a wrapper library for libSVM, because of its great performance (runs much faster than Weka SMO) and flexible supports to several SVM methods (i.e. One-class SVM, nu-SVM, and R-SVM).

# of document	# of features	Accuracy
400	1302	86.9806 %
800	2240	80.3324 %
1200	4795	82.202 %
1800	5281	83.1858 %

The results stay at an average of 80+%. Combining these two result, we observe that Naive Bayes classify more accurately than what SVM does. However WLSVM shows its extraordinary advantages, in terms of memory usage and running performance, whereas Naive Bayes Classifier takes more significant amount of time and memory on the same sample set.

5.2 URL Classification (Email address plus URL address)

Our second experiment focus on the URL address and Email address appeared in message and from the sender, in which about 60% are from suspicious spam address. We then perform classification with Naive Bayes and KNN Classification:

5.2.1 Naive Bayes

# of samples	Accuracy
1629	96.6851 %
8124	84.6847 %
12300	72.2457 %

5.2.2 K Nearest Neighbor

# of samples	Accuracy (k=1)	Accuracy (k=3)	Accuracy (k=5)
1629	97.8449 %	80.7858 %	78.76 %
8124	96.2457 %	80.6253 %	77.8449 %
12300	96.6623 %	Out of Memory	Out of Memory

From observation, the accuracy achieve the best, when k equals to 1, and eventually decrease, while we increase the value of k. The choice of k are odd number, because we want to break the tie in this binary classification experiment.

6 Conclusion and Future Work

In the experiment, we have seen the traditional way (text classification) of Bayesian filter usually outperforms other methods in term of training time and performance. This is the reason why it is widely used in commercial email filter software. In addition, we think URL classification results an interesting outcome, and is worth more investigation and experiment in future. However, in our experiment, there are several unsolved issues in the project. The most important and critical attack is Bayesian poisoning, sometimes called good word attack. This occurs when the spammer intends to skew the message statistics, by putting enough "neutral" text in your spam message, and dropping the score below the statistical threshold of such filters. This has been a good strategy to beat statistical filtering, such as Bayesian Filter, which looks at the content of a message and weigh the presence of spam-related words and phrases against the message as a whole. Other spammer tricks, such as Image Spam or Table-based obfuscation, are not within the scope of this project.

There is no doubt that spam represents a significant, global threat to every internet users. The costs associated with the problem are going out of control, as employees lose productivity and companies spend billions each year to process and store spam messages. However, the good news is current filtering technique perform highly effective, and

has strong resistance to most attack. To conclude, by this experiment, we enhance our understanding of the machine learning algorithm by coding and implementing these algorithms.

References

- [1] P. Graham, “A plan for spam <http://www.paulgraham.com/spam.html>,”
- [2] I. The Radicati Group, “Trend micro anti-spam innovative defense against evolving spam - a white paper,” tech. rep.
- [3] C. Dwork and M. Naor, “Pricing via processing or combatting junk mail,” pp. 139–147, Springer-Verlag, 1992.
- [4] P. Pantel and D. Lin, “Spamcop: A spam classification & organization program,” in *In Learning for Text Categorization: Papers from the 1998 Workshop*, pp. 95–98, 1998.
- [5] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz, “A bayesian approach to filtering junk e-mail,”
- [6] I. Androutsopoulos, G. Paliouras, V. Karkaletsis, G. Sakkis, C. D. 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 (PKDD 2000)*, pp. 1–13, 2000.
- [7] V. N. Vapnik, *The nature of statistical learning theory*. New York, NY, USA: Springer-Verlag New York, Inc., 1995.
- [8] T. Joachims, “Text categorization with support vector machines: Learning with many relevant features,” 1998.
- [9] G. Rawlinson, “The significance of letter position in word recognition.” Unpublished PhD Thesis, 1976, Nottingham University, by Graham Rawlinson.