

# **Master's Thesis**

## **Creation of a Bayesian network-based meta spam filter, using the analysis of different spam filters**

Csaba Gulyás

Hungarian student ID: UDLHNH  
Hungarian supervisor: Péter Antal  
Danish supervisor: Jyrki Katajainen

Budapest, 16th May 2006

## Table of Contents

<b>Table of Contents</b> .....	<b>2</b>
<b>Nyilatkozat</b> .....	<b>5</b>
<b>Tartalmi összefoglaló</b> .....	<b>6</b>
<b>Abstract of the thesis</b> .....	<b>7</b>
<b>PART I</b> .....	<b>8</b>
<b>1. Introduction</b> .....	<b>8</b>
<b>2. The definition of spam</b> .....	<b>8</b>
<b>3. The history of spam</b> .....	<b>9</b>
3.1 The different periods in spam history .....	9
3.2 The first period – the early years .....	9
3.3 The second period – letters sent by machines .....	10
3.4 The third period – machines against machines .....	10
<b>4. The necessity of spam filtering</b> .....	<b>11</b>
4.1 Overview .....	11
4.2 The users' side .....	11
4.3 Facts and figures .....	12
4.4 Summary.....	13
<b>5. Different kinds of spam</b> .....	<b>13</b>
5.1 The characteristics of spam .....	13
5.2 Advertisement spam.....	13
5.3 Financial spam .....	14
5.4 Phishing .....	15
5.5 The trends in spam messages.....	15
<b>6. The simple methods to fight against spam</b> .....	<b>16</b>
6.1 Overview .....	16
6.2 Keywords.....	16
6.3 Blacklist.....	17
6.4 White list .....	17
6.5 Throttling.....	18
6.6 Collaborative filtering .....	18
6.7 E-mail interferometry.....	19
6.8 Filtering on the network level.....	19
6.9 Fake worker .....	19
6.10 Project Honey Pot .....	20
6.11 Electronic stamp.....	21
<b>PART II</b> .....	<b>22</b>
<b>1. Overview</b> .....	<b>22</b>
<b>2. Machine Learning</b> .....	<b>22</b>
<b>3. The Bayesian network</b> .....	<b>23</b>
3.1 The Bayesian network in general.....	23
3.2 The naive Bayesian model.....	24
<b>4. Bayesian network-based spam filter</b> .....	<b>24</b>
<b>5. The algorithm of the Bayesian network-based filter</b> .....	<b>25</b>

5.1 Loading.....	25
5.2 Pre-filtering.....	25
5.3 Tokenization.....	26
5.4 Calculation.....	26
5.5 Feedback.....	27
5.6 Training and different training types.....	27
5.7 Testing.....	27
<b>6. Concrete solution of the Bayesian algorithm.....</b>	<b>28</b>
6.1 Overview.....	28
6.2 The token dictionary.....	28
6.3 Getting the result.....	29
6.4 Training.....	29
6.5 Testing and using the filter.....	30
6.6 Calculation.....	30
6.8 Concrete example.....	31
6.9 Flow Chart of the Bayesian spam filter.....	33
<b>7. Preparation for comparing spam filters.....</b>	<b>34</b>
7.1 Overview.....	34
7.2 The corpus.....	35
<b>8. Choosing the spam filters.....</b>	<b>36</b>
8.1 Overview.....	36
8.2 SpamAssassin.....	36
8.3 Mozilla Thunderbird.....	36
8.4 SpamProbe.....	37
<b>9. Training and testing filters.....</b>	<b>37</b>
9.1 Overview.....	37
9.2 The different training methods.....	37
9.3 Determination of training and testing sequences.....	38
<b>10. Analysis of the results.....</b>	<b>40</b>
10.1 Part of the results.....	40
10.2 The characteristics of the results.....	40
10.3 Another training and testing possibilities.....	41
10.4 Summary of the performed training and testing sequences.....	42
<b>11. Statistical analysis.....</b>	<b>42</b>
11.1 Time factor.....	42
11.2 Number of mistakes and error rate.....	43
11.3 Conclusion.....	46
<b>PART III.....</b>	<b>47</b>
<b>1. Introduction.....</b>	<b>47</b>
<b>2. Results analyzed with Bayesian networks.....</b>	<b>47</b>
2.1 Overview.....	47
2.2 The BNET software.....	48
2.3 The CSV file format.....	48
2.4 Structure learning.....	49
<b>3. Results of the structure learning.....</b>	<b>49</b>
3.1 Overview.....	49
3.2 Analyzing the results of the first part – whole corpus.....	49
3.3 Analyzing the results of the second part – reduced corpus.....	51
3.4 Additional analysis, in order to realize the meta filter.....	52
<b>4. The construction of the meta spam filter.....</b>	<b>53</b>
4.1 Basic ideas.....	53

4.2 The ROC-curve analysis.....	53
4.3 The meta models .....	55
4.4 Confirmation of the results .....	58
4.5 Summary.....	60
<b>Conclusions .....</b>	<b>61</b>
<b>References.....</b>	<b>62</b>

## Nyilatkozat

Alulírott, Gulyás Csaba, a Budapesti Műszaki és Gazdaságtudományi Egyetem hallgatója kijelentem, hogy ezt a diplomatervet meg nem engedett segítség nélkül, saját magam készítettem, és a diplomatervben csak a megadott forrásokat használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

-----  
**Gulyás Csaba**

## Tartalmi összefoglaló

A diplomaterv célja egy olyan meta spam szűrő megalkotása, mely több különböző spam szűrő együttes felhasználásával végzi el feladatát. Ehhez szükséges a spam küldés és spam szűrés múltjának és jelenének megismerése, valamint a spam szűrők eredményességének értékelése. A meta spam szűrő Bayes hálóban kerül reprezentálásra. A diplomaterv a következőkben részletezett három fő részből áll.

Az első rész bemutatja a levélszemét, az úgynevezett spam jelenségét. A spam fogalmának meghatározását a spam szűrés rövid történeti áttekintése követi. Majd egy magyarázat következik a spam kialakulásának okairól és szűrésének szükségességéről. Végül a spam különféle típusai és szűrési módszerei kerülnek ismertetésre.

A második rész bemutatja a gépi tanulás módszerét felhasználó spam szűrési módokat. Az első fejezet a Bayes hálókat és azok speciális esetét, a naiv Bayes hálókat ismerteti, különös tekintettel az osztályozási feladatok területén nyújtott használatukra, hiszen a legújabb spam szűrők is ezen az elven alapulnak. Ezután a spam szűrőkben működő naiv Bayes algoritmus leírása következik. Az elvégzett munka egyik fontos eleme az elterjedten használt spam szűrő programok tesztelése, melynek eredményei statisztikai elemzésben kerülnek összehasonításra.

A harmadik rész bemutat egy speciális matematikai modellező és elemző szoftvert, a BNET-et. Ezután a BNET-el előállított önszerveződő Bayes hálók elemzése következik, melynek segítségével a tesztelésben részt vett spam szűrők hasonlóságára és kapcsolatára vonatkozó következtetések vonhatók le. Ezen elemzés eredményét felhasználva lehetőség nyílik azon javaslatok kidolgozására, melyek egy meta spam szűrő előállításában játszanak fontos szerepet. A módszer segítségével megállapítható, hogy mely szűrők együttes hatása növeli az eredményességet, és melyek együttes használata bizonyul feleslegesnek. Az utolsó fejezet konkrét meta spam szűrő modellek tesztelése alapján a spam szűrő programok együttes használatával elérhető eredményességi fokok összehasonlítását és elemzését tartalmazza.

## Abstract of the thesis

The goal of the thesis is to construct a meta spam filter utilizing several spam filters at the same time. Therefore, it is important to understand the past and present of both spamming and spam filtering. Based on the best-practice solutions of present days, the meta filter is constructed as a Bayesian network. The thesis has three main parts:

The goal of the first part is to introduce the fact and phenomenon of junk e-mail also called *spam*. The first part covers what spam is and also gives a brief overview about the history of spam and spam filtering. This section is followed by the explanation why spam exists and what it causes. Furthermore, the different kinds of spam are introduced and finally the spam filtering methods are reviewed.

The second part contains the filtering methods use machine-learning techniques, Bayesian network based solutions. The first section is a general summary about Bayesian networks, followed by the naive Bayesian model, and the extended usage to solve classification problems, like spam filtering. Then a description of a Bayesian network based algorithm is given and finally the statistical results of different spam-filter software are analyzed.

The third part introduces a special tool for modeling and analyzing Bayesian networks, the BNET software. It is followed by the analysis of self-constructed Bayesian networks, what shows the relevance of the tested filters. The result of the analysis gives the answer, which filter software are substantiated to use together considering both the achieved accuracy rate in filtering and the computational resources. The final section gives ideas on the construction of a meta spam filter, using the earlier selected spam filter software, in order to achieve higher accuracy and effectiveness in spam filtering.

# PART I

## 1. Introduction

The first part introduces the fact and phenomenon of *Unsolicited Commercial E-mail (UCE)* what is often called *spam*. The first section covers the different definitions of spam and concerns the history of spam. This section is followed by the short introduction of the necessity of spam filtering and users' opinion about the topic. Then the different kinds of spam are introduced, with their trends and characteristics and finally the chapter gives a glance at the common filter techniques and the basic solutions how to handle spam.

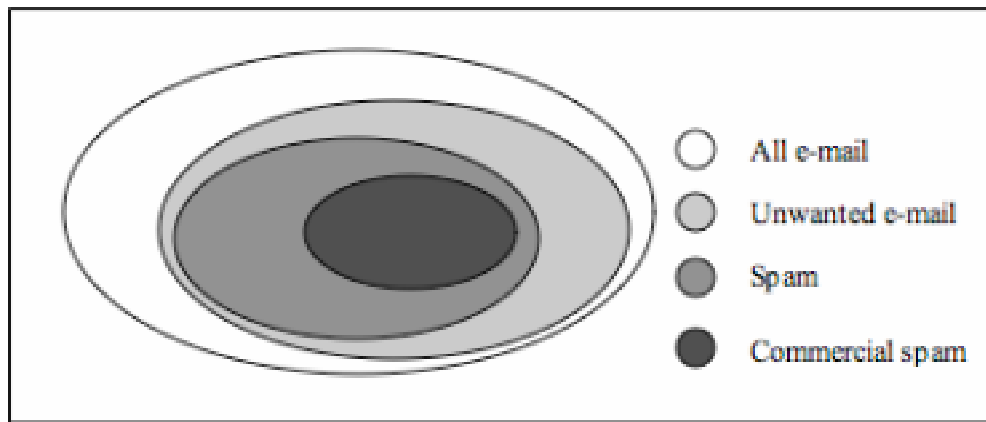
## 2. The definition of spam

There is no exact definition of spam. Most of the spam can be termed as *unwanted e-mail* but not all of the unwanted e-mails are spam. Another term would be *unsolicited commercial e-mail* [11], but unfortunately spam is not only advertising material. (The different kinds of unwanted e-mails are discussed later on.) Spam can be also defined as *junk mail* but it implicates the question: what is a junk mail? Although most of the e-mail users know what spam is, but it is not obvious how to define spam and spamming.

Wikipedia, the biggest encyclopedia on the Internet gives the following definitions: Spam: "E-mail spam (...) involves sending nearly identical messages to thousands (or millions) of recipients." [53] Spamming: "Spamming is the abuse of any electronic communications medium to send unsolicited messages in bulk." [59]

As a summary one could agree that spam is something unsolicited, unwanted e-mail what is mostly also an advertisement material. However not all unwanted e-mail letters are spam and not all spam is an advertisement. It is not an exact definition of spam these are only properties in order to explain, what is the relationship between spam and other e-mail sets.





2.1 Figure: The set of spam, compared to other e-mail sets

### 3. The history of spam

#### 3.1 The different periods in spam history

The history of spam can be separated into three different parts:

- The early years spam letters were addressed and sent manually
- Later spammers used machines, what led to the dramatic rise in the amount of spam
- The final part is when machine learning appeared at spam filtering and made the filtering substantially effective.

#### 3.2 The first period – the early years

The era of spamming began in 1978 and lasts until the middle of 90's. At the first years, spam was sent manually. Spamming needed huge amount of human resource and so it was not able to reach millions of users. Spammers used to send the messages one by one and used inner address lists of small communities. These were the tolerable years.

### **3.3 The second period – letters sent by machines [11]**

The first mentionable change happened in 1994, when a programmer was hired in order to write a spammer program. Since that time the amount of received spam letters has been growing uncontrolled. The first very famous spammer was Jeff Slaton, the “Spam King”. He was not only the first known man with a capability of sending millions of e-mails at the same time, but he also had dared to offer a remove for the users from all spamming lists for \$5.

Spamming slowly became a business. The first known e-mail list was offered for sale in 1995 with more than 2 million addresses. After the first weak attempts in 1997 the first real spam-filter software was made. Nevertheless spam turned to be totally out of control, by the end of 1997 statistics showed an exponential growth. The early spam filters were rule-based software. All the new tricks of the spammers indicated to settle up new rules into the filters. Unfortunately, the spammers were able to try the filters with their messages, so they had the ability to change the letter to get through the filters.

### **3.4 The third period – machines against machines**

The big breakthrough was in 2002, when Paul Graham published “A Plan For Spam”. This article introduced the usage of machine learning and statistical classification techniques on the field of spam filtering with Bayesian networks. Paul Graham wrote [41]: “I think it's possible to stop spam, and that content-based filters are the way to do it. The Achilles heel of the spammers is their message. They can circumvent any other barrier you set up. They have so far, at least. But they have to deliver their message, whatever it is. If we can write software that recognizes their messages, there is no way they can get around that.”

With the use of Bayesian networks, filtering not only became significantly more effective, but first time the filter learned the e-mail characteristics of the users. It was the first solution, which made it impossible for the spammers to test the filters before sending their e-mails, hence every users’ filter worked with different knowledge base.

From this time spammers could only hope that they found the right method to pass the filters, but they could not be sure they could reach a defined number of users. Even if the spammers figured out a lot of new tricks, like obfuscating or good-word-attacks, the Bayesian filters still have a good chance to filter them out.

## **4. The necessity of spam filtering**

### **4.1 Overview**

Nowadays the implementation of a reliable spam filter becomes more and more important for e-mail users since they have to face with the growing amount of uninvited e-mails. The spam is mainly a cheap and illegal form of advertisement exploiting that thousands of users are easily reachable on the Internet. Although it is illegal, most legislation enforcements fail due to the inability to identify the spammer. The deficit due to the workers time-loss can be measured in thousands of dollars. A new solution in spam filtering always indicates the spammers to try to find the way through the filters. That causes spam filtering to be a never-ending fight. Currently some of the most efficient filters are based on Bayesian network classifiers, although there are several less complex methods as well to amend the filtering.

### **4.2 The users' side**

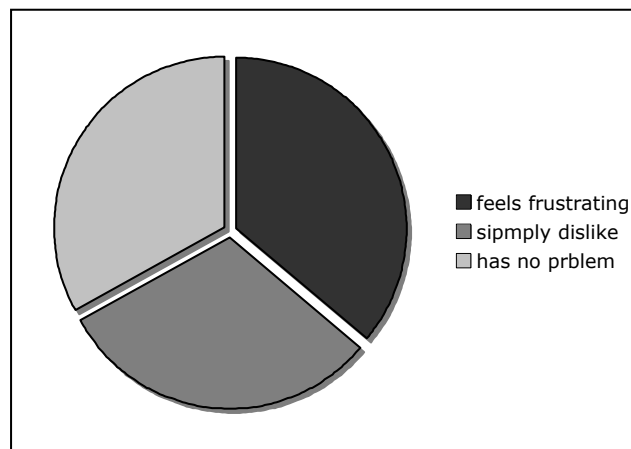
The irritating-rate of spam letters depends on the e-mail habits of the users. For example if somebody receives 10 letters every week, and there is only one spam than it could be easily imaginable that to press the delete button is quicker and more comfortable than implementing any kinds of filtering. Spam causes problem usually for users who receive hundreds of e-mails per weeks. One can find really frustrating to press the delete button hundred times every week.

In the first years, the main problem was the connection speed. Especially users connecting through modem to the Internet were suffered by waiting significantly more time, spent with downloading e-mail. To wait ten minutes for download a couple of letters can be painful, if the 70% of the letters were actually spam letters.

Nowadays, when more and more users use broadband Internet connection, the downloading time plays a smaller role. Spam is usually short message and/or smaller picture. Today the problem is that the users have to face the necessity of deleting the unwanted letters, or the difficulty to pick out the important ones from the big amount of junks. Moreover there is the annoying question: “why I have to get all these letters?”

### 4.3 Facts and figures

The statistics in the April 2005 showed, that 67% of the Internet users dislike spam and 33% of them not only dislike spam, but also feels them frustrating, while only 33% of the users have no problem with it. Although the image shows almost a balanced 33-33-33 percent, it is important to highlight, that two-thirds of the users have problem with spam [43].



4.3.1 Figure: Opinions of the users about spam

For example, if a company has no filter and a worker receives 6 spam messages daily and it takes an average of 5 seconds to read and delete each spam message this means, that the worker will spend almost 3 hours per year to read and delete spam. These estimates are conservative and don't factor with time spent discussing the latest spam trends with co-workers or contact the Help Desk, etc.

## 4.4 Summary

The main thing what is important to understand, that advertisement works only, if people are capable for it. If people buy from the offered product or if people can be fooled by phishing, spamming can survive. If nobody would response to spammers, or buy from them, surely spamming would reach its end.

## 5. Different kinds of spam

### 5.1 The characteristics of spam

Although there are at least three main different kinds of spam and especially advertising spam could be divided into many subcategories, all the spam has also a content-free characteristic. The majority of spam follows common patterns, which could be clearly identified. More than 99% of spam falls into one or more of the categories listed below [47].

The followed characteristics or trends can be for example the language. Most of the spam is written in English. The numbers are different through the year of the research or method (80%, 84%, 95%) [45] but it is clear that the rate is very high. Other such a trend is by the sending time. The spending time of spam follows the working times in the U.S. In the EST (in summer EDT) time zone (Washington DC, New York, etc.) in average 50% more spam is received between 8 a.m. and 2 p.m. than in the other hours of the day [22]. This information can become the base of further usage. For example: Non-English speaking users may receive legitimate English mails with smaller probability. Non-US residents may receive most spam in the US working time what could be nighttime in another part of the world. All of these characteristics could be the base for further filter solutions.

### 5.2 Advertisement spam

Most spam is commercial advertisement, often a direct product offer. Spam costs the sender very little to send, compared to other advertisement methods. The most common subcategories of the advertisement spam are:

- Online Pharmacy spam: Spam promoting different versions of Viagra, Cialis, Anti-depressant pills that can be purchased online.
- Penny Stock spam: Stock-encouraging spam, encouraging people to buy cheap stocks.
- Porn or (sex-) dating spam: Porn-sites and (sex-) dating sites were often marketed via spam (nowadays its rate out of all spam is getting less and less, fortunately).
- Pirate Software spam: spam offering pirate software, usually much more cheaper than the official prices.
- Online Casino spam: Spam promoting gambling in online casinos.
- Fake Degrees spam: Spammers often try to sell fake Degrees and Diplomas.
- Mule job spam: Promoting jobs 'working from home' (which are typically scams, or mule jobs, like laundering money).

### **5.3 Financial spam**

While advertisement spam have at least a little probability, that the responder could get something for the sent money, the financial spam only tries to fool people and get their money somehow, without the chance to buy anything. The most common financial spam kinds are the following:

- 419 scams: Usually a plea for help to recover millions of dollars from a bank account in a foreign country (typically Nigeria).
- Lottery spam: Similar to the 419 scam, these spam are telling, 'You have already won X Million' in order to try to extract transfer fees etc.

## 5.4 Phishing [42]

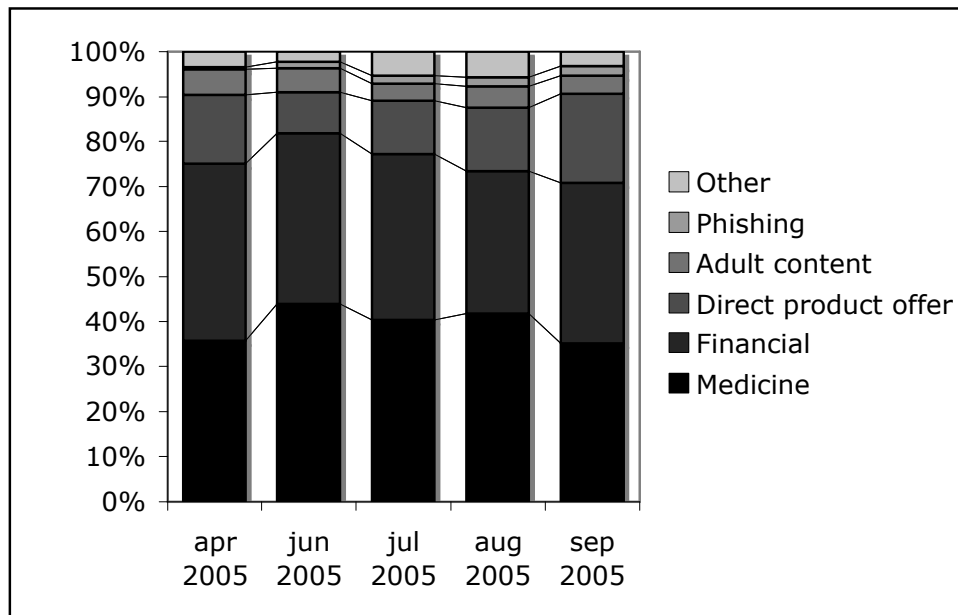
Phishing spam is fake alert from banks (mostly CitiBank), PayPal, eBay etc, and it asks for confirmation, validation or monitoring of details in order to defraud people of their personal information. Phishing spam are usually linked to fake login sites, which can be used to capture user details (e.g. passwords) in order to use this information to steal money or goods. The term phishing was coined because the fraudsters are “fishing” for personal information.

Fraudulent e-mails harm their victims through loss of funds and identity theft. They also make a draw back in on-line business, since people loose their trust in Internet transactions. Phishing e-mails use mostly the listed methods:

- Using the company’s Image: the fraudulent e-mails often contain the company’s logo and use similar fonts and colour schemes.
- Links to the real company’s site: The main link in a fraudulent e-mail sends the recipient to the fraudulent phishing web site, but many fraudulent e-mails include other links that send the recipient to sections of the real company’s web site.
- E-mail appears to be from the spoofed company: To further convince the recipient that the e-mail originated from the company, the spammers use an e-mail address that appears to be from the company (e.g., @ebay.com, @paypal.com).

## 5.5 The trends in spam messages

Although the content of spam letters are in continuous change, there is a slow flow remarkable. Analyze the data of 5 months shows, that the most common spam is the medicine product offer spam, with a significant rate one third of all spam. The second biggest category is the financial spam, where is a slight drawback from around 40% to 30%. The direct product offering follows a waving tendency and the adult contents gets fortunately less accent. The next figure shows the trends in spam



5.5.1 Figure: The trends in spam [44]

## 6. The simple methods to fight against spam

### 6.1 Overview

Several different very simple methods have been discovered during the evolution of spam filtering. These are ideal to be combined with more complex parts of spam filtering; therefore the area of spam filtering becomes very complex. The biggest effectiveness can be reached by using together small parts of filtering methods.

### 6.2 Keywords

One of the first solutions is to search for keywords in the e-mail's subject. It means to scan the subject for words, related to spam letters. This is a simple language analysis, works only by match specific phrases. This method has unfortunately several problems, since the spam letters topic changes time by time. This can be handled by a keyword list that is regularly updated, but the smallest change in the words of the subject leads to mismatch (e.g. write "softw@re" in spite of "software"). Filtering based on a single word had a potential success rate of around 80 percent.



Keyword searching was based on known list of words and phrases which most of the word believed only existed in spam. The simplicity of these spam filters led to a high false-positive rate and it had also a significantly high maintenance rate.

### **6.3 Blacklist**

There is a need, to make difference between two levels of blacklisting: the network-level and the address-level blacklisting. The network-level blacklisting is based on creating intentional network outages. The method has the ability the detect spam letters based on its origin rather than its content. Unfortunately new spam hosts can pop up instantly and the propagation time could be a significant weakness. Moreover if a legitimate user was accidentally blacklisted, there is no way, to get off the blacklist, hence all mails were rejected from the blacklisted part of the network. Spammers have learnt quickly how they can get around blacklisted networks.

The address-level blacklist is an updated list of known spam sender addresses. There are on-line accessible blacklists and the user can administrate personal blacklist as well. By receiving a letter, a simple search engine tries to find the address of its sender in the list. If it matches, the letter should be surely marked as spam, or with more strict rules, it could be deleted immediately. The “answer” of the spammers for this solution is to try sending the letters from addresses, containing random parts within, to avoid the match with the blacklist entries (e.g.: John87sd4vu8@hotmail.com).

Both blacklisting solutions implicated only a small change from the spammers’ side and made no big change of the amount of spam, but it is still a very good technique to combine with other filter methods.

### **6.4 White list**

White listing is the opposite of blacklisting. Content filtering identifies spam, while white listing requires identifying users. A white list is a collection of reliable contacts. If e-mail comes from the members of this list, it should be marked automatically as legitimate letter what is also called *ham*. Just as the blacklisting, the white list also needs a continuous upgrade and refreshment.

Rejecting all e-mails from unknown senders is a far too strict. A more appropriate usage can be achieved by sending an auto-reply for every unknown user with an ask for authentication (challenge/response). This can cause limited speed and rely on the sender's will. After all of these drawbacks, white listing is used only for classify letters as legitimate mails, and has nothing to do when the sender is unknown. If the blacklist and white list methods are used together, further filtering is only required for letters that do not match any of the entries in the two lists.

## **6.5 Throttling**

The throttling simply slows down the rate at which a single network or host can send traffic. Probably this is the most sensible way to fight spam. For example a legitimate mailing list may send out huge quantities of mail, but each messages are addressed to different users on different networks. A spammer on the other hand may use dictionary attack, and tries to find valid e-mail addresses on one network. In this case throttling can lead to a drawback for the spammers, but it also uses more resources from the legitimate senders. Unfortunately spammers more likely collect e-mail addresses for sending spam letters to, and they use dictionary attack rarely what makes throttling rather a theoretical then a practical solution.

## **6.6 Collaborative filtering**

To filter in collaborative way relies on "two hands are better than one" philosophy. It will allow individuals in trusted groups to share message inoculations with the other group members. For example if ten users agree to share their training errors and to see a new spam type or a new mutation has the probability 0.9 for each person. The person, who received the letter first trains not only his, but the filter of all other users. After all if one received 1 spam out of 100 legitimate e-mail, after sharing the training one will receive 1 spam out of 1000. The weakness of collaborative filtering is the participating community itself. In large communities, there is both a high maintenance loop and the risk of false positives.

## 6.7 E-mail interferometry

Spammers have always been able to adapt their strategies to fool content-based filters. It makes necessary to develop new content independent filters. E-mail interferometry is a collaborative filtering way, ideal for big e-mail providers. The solution relies on recognizing identical URLs; similar message body looks, sending time periods and nearly identical header patterns. E-mail interferometry is a really new way to fight spam and its accuracy strongly depends on the community what shares the information. It is applicable primary for companies to amend their existing filters.

## 6.8 Filtering on the network level

The Simple Mail Transfer Protocol (SMTP) is the way mail servers communicate with each other. This protocol was designed to function anonymously to guarantee the privacy of Internet users. Spammers have taken the advantage of this aspect of e-mail servers to send spam anonymously. Originally the Authenticated SMTP thought to be an answer to spam, but it turned out to be useful only to identify legitimate senders of mail on a system. Authenticated SMTP requires users to provide their password before they are allowed to send mail. Many spammers today build their own mail servers and host them on unsuspected networks in order to send out the mail, thereby bypassing any authenticated sending. [11]

SMTP opens several different opportunities for further usage. One of them is a new algorithm for learning the reputation of e-mail domains and IP addresses. This could be done based on the analysis of the paths used to transmit e-mails. Combined with machine learning techniques, the result is an effective algorithm what gives the information needed for domain authentication. It has the opportunity to make filtering decisions. [46]

## 6.9 Fake worker

The main idea of the solution is to create a fake e-mail address that functions like a trap. This can be used especially at companies (e.g.: Xyz@company.com). After generating such a fake e-mail address, one can be sure that no real sender should contact this address. In this case every mail, what this fake-worker receives, is probably spam. Using this information can help filtering the content of the inboxes from other workers.

Very similar to the fake worker idea is what Hotmail uses. MSN Hotmail has more than 130,000 trap mailbox accounts [46], on which Microsoft's protective spam filters are not working – what means, the trap accounts are exposed to the Internet and receive all mail sent to them. The letters arrived to the trap e-mail addresses help Hotmail in filtering the inboxes of real users. They surely need it since about 4 million spam letters are arriving to hotmail.com e-mail addresses daily.

## 6.10 Project Honey Pot [32]

Project Honey Pot is the first system that not only filters spam or tries to defend, but also fights back spammers. It is capable to identify spammers and the robot programs that they use to harvest e-mail addresses from websites. The spammer's robots are scanning the websites and are searching for *username@domain.com* –like addresses. Once a robot visited a whole website, it follows the search on all links what are published on the site. This is the most common way, how spammers collect e-mail addresses. The first idea was, to fool the robots with sites, where fake e-mail addresses are published. The background of the idea is the following: if the spammers send messages to non-existing addresses, it will draw back their efficiency. Later it turned out that the idea hides more opportunities, since the letters arrived on fake e-mail addresses are definitely spam.

Project Honey Pot system offers a program what every user can run on their website. The system publishes the fake e-mail addresses and if one of these addresses begins receiving e-mail the program can tell the exact moment when the address was harvested and the IP address that gathered it. To participate in Project Honey Pot, webmasters need only install the Project Honey Pot software somewhere on their website. All the rest is handled automatically.

Project Honey Pot also works with law enforcement authorities to track down and prosecute spammers. Harvesting e-mail addresses from websites is illegal under anti-spam laws, and the data what Project Honey Pot results are critical for finding those breaking the law.

The statistics of the project are the following: The average time from being harvested to first received spam is: 38 days. In the slowest case it was 1,5 years and in the fastest case only 1 second.

6.10.1 Table: Address harvesting

1	United States	27.2%
2	Romania	17.0%
3	China	11.6%
4	Japan	10.0%
5	Germany	8.5%
6	Netherlands	6.9%
7	United Kingdom	6.4%
8	Spain	4.4%
9	France	4.2%
10	Canada	3.8%

6.10.2 Table: Spam sending

1	China	27.8%
2	United States	24.3%
3	Korea	18.1%
4	Japan	7.1%
5	Brazil	5.1%
6	Taiwan	4.6%
7	France	4.0%
8	Spain	3.3%
9	Poland	3.1%
10	United Kingdom	2.5%

## 6.11 Electronic stamp<sup>1</sup>

Other possibility for the future could be to use some kind of post offices for e-mail delivering. The idea of post offices would cause the end of free and anonymous e-mailing, and so now it is more a fiction than a way to go. The basics are the following: all the e-mail traffic should be controlled by big intelligent nodes on the network, and every e-mail would cost for the sender a little money, for example 0,01 euro (1 euro cent). If the receiver signs back that the e-mail was not spam, the sender gets back the 99% of the cost of the e-mail. The rest 1% goes for the post office nodes budget. For legitimate senders it could be still affordable and still the cheapest solutions, while spammers, who want to send millions of letters in a single day, should pay big amounts of money. To avoid unnecessary money transfers, it would be suitable to build a pre-paid system, and so money transfer would be enough once monthly. The idea holds the possibility of ending spam in an unbelievable high cost, by changing the whole architecture on network level, and making the users to pay for sending e-mails. Even so, these fictions are really important on our way, to find the solutions for the problem called spam.

---

<sup>1</sup> Based on the idea of Jyrki Katajainen, University of Copenhagen

# PART II

## 1. Overview

The idea to use Bayesian network based classifiers in spam filtering originates from David Heckerman's article (2002) "A plan for SPAM" [41]. The article summarized, how it is possible to use naive Bayesian network-based statistical classification method to achieve a reliable and learning filter. Several different modifications were published since that time, but most of them use the same base, the same engine. The new method gives the ability for computers to make their own decisions rather than being told how to response, like at rule-based solutions.

The appearance of machine learning techniques in the spam filtering effected a significant improvement in filtering. The spammers are not anymore able to test the filters before sending out the messages, since every user's filters have its own knowledge base. Although spammers are figuring out new ideas to pass through filters, the accuracy of filtering after satisfying learning cycles could easily reach the 99% with a well-trained Bayesian filter. The goal now is the next step 99.9%.

## 2. Machine Learning

Machine learning is the development of algorithms and techniques, which allow computers to "learn". It is a wide area of artificial intelligence. Machine learning has a broad spectrum of applications including search engines, medical diagnosis, bioinformatics, detecting credit card fraud, stock market analysis, classifying DNA sequences, speech recognition, computer games, robot locomotion and spam filtering.

### 3. The Bayesian network

#### 3.1 The Bayesian network in general [51]

A Bayesian network is a form of probabilistic graphical model, named after Thomas Bayes. Structurally, a Bayesian network is a directed acyclic graph where nodes represent variables and arcs represent dependency relations between the variables (nodes). An arc from node A to another node B is called: A is a parent of B. A node can represent any kind of random variable.

A Bayesian network with parameters is a graphical representation of the joint distribution over all the variables represented by nodes in the graph. If the variables are  $X_1, \dots, X_n$  we let “parents(A)” be the parents of the node A. Then the joint distribution for  $X_1$  through  $X_n$  is represented as the product of the probability distributions:

$$P(X_1, \dots, X_n) = \prod P(X_i \mid \text{parents}(X_i)) \text{ for } i = 1 \text{ to } n.$$

In order to fully specify the Bayesian network and to carry out numerical calculations, it is necessary to further specify for each node X the probability distribution for X conditional on its parents. In this way a Bayesian network could be used to perform any probabilistic inference over the domain variables.

Other important usage of the Bayesian networks is modeling, where the structure of the Bayesian network is generated by software. Learning the structure of a Bayesian network is a very important part of machine learning. To find the structure of the network, a scoring function should be maximized through a search algorithm. Unfortunately this leads to a super-exponential exhaustive search. [54]

Bayesian networks are used for modeling knowledge in many domains with uncertain knowledge, like medicine, engineering, text analysis, image processing, data fusion, decision support systems, and spam filtering.

### 3.2 The naive Bayesian model

The Bayesian network is called naive, if only one parent node is contained in the network and all the other variables are the children of the parent node. If the parent variable is “ $X_p$ ”, the joint distribution formula in that case is the following:

$$P(X_p, X_1, \dots, X_n) = P(X_p) \prod P(X_i | X_p) \text{ for } i = 1 \text{ to } n.$$

A naive Bayes classifier is a simple probabilistic classifier. Its main advantage is that naive Bayes classifiers can be trained very efficiently in a supervised learning. Naive Bayesian classifiers are used for parameter estimation in numerous practical applications.

## 4. Bayesian network-based spam filter

The Bayesian network-based spam filter uses the content of the e-mail. The main phases of the Bayesian network-based solution are the following: First of all there is a need to tokenize the e-mail, which means to separate it into small parts that are used further in the process. These tokens can be sentences, or word-pairs, but usually single words are used to define tokens. After that step the value of every token is determined by looking up in an updated table, what we call the token-dictionary. In this table one or more values are stored for each token. After getting the value of each token, there is a way to calculate a probability of the e-mail being a spam or a legitimate. Most implementations do not deal with all the values, usually in order to save processing time they calculate only with the most relevant values. The values of the relevant tokens have the largest distance from the neutral value and so they are close to be an obvious mark of spam or legitimate letters. These values are used to establish the so called decision matrix. Most Bayesian filters limit the number of tokens in the decision matrix, usually to 15 or 27 of the most interesting tokens. [11] The evaluation of the decision matrix is given in the section 6.6 in part II.

The final step is to modify the values of the tokens in the dictionary, this gives the continuous learning capability with the feedback; and the final binary result is produced.



If not only a binary result is required, it is also easily reachable to give the result as probability of being spam. This probability can be used in many different ways. For example, limits can be settled this way generating categories and different actions can be performed for them (e.g.: under 40% – put it in the inbox; 40-80% – put it in the inbox, with a marked subject: ”possible spam”; above 80% – put it into the spam folder; etc.).

## **5. The algorithm of the Bayesian network-based filter**

### **5.1 Loading**

The first step is loading the e-mail. The most comfortable filters are combined with e-mail programs and the downloaded message is filtered in real-time.

### **5.2 Pre-filtering**

Every single method based on text string searching needs a really reliable pre-filtering to make it easier match the words in the previously stored list the token-dictionary. Without pre-filtering, the tokens kept in the token-dictionary could reach an unlimited size, since the spammers realized in a very short time, that writing the words in their dictionary-form leads to an easy filtering, and so they began to change some letters or put extra characters in (e.g.: softw@re, r\_i\_c\_h). This has an effect not only on the keyword searching methods but also makes statistical filtering more difficult. The pre-filters role is to try changing the words back to their original form and to make the search engines able to find them. An interesting newer method does not try to change the words back, rather tries to find similar words. These methods are calculating a value that is similar to the “Hamming distance” to find relevant words in the token-dictionary. Even so many of the filters are not using pre-filtering and so those deal with larger sets of tokens.

### 5.3 Tokenization

The Bayesian filter works with the individual small parts of the text, the so-called tokens. This very simple hierarchic model fits to the naive Bayesian model and it does not count with the dependency between tokens. Many filters work with simple word-by-word tokenizing, where the text is separated into words and the words have their own values in the token-dictionary. In this case, one word will be one token. However, some filters offer word-pair tokenization, which requires more maintenance and more resources for calculating, but it leads to better filtering accuracy. For example: the word “software” and the word “cheap” can be also found in legitimate e-mails, but the word-pair “cheap software” can be a strong indicator of being spam. The dictionary should be updated with all the new tokens from the processed e-mail. After the tokenization, the tokens are handled separately, what means that the order of the tokens is taken into consideration.

### 5.4 Calculation

After the pre-filtering and tokenization the values of the tokens are looked up and a decision matrix is built from the most relevant token's values. Usually the filters use only a fixed number of values, the ones, which are farthest from the neutral value. In an environment supported by unlimited hardware resources surely all the tokens could take part into the calculation. With the limited number of tokens, only the significant tokens are taken into consideration and this leads to relevant faster running time due to the reduced complexity in calculation. If a token does not exist in the dictionary, it is added as a new token to the token-dictionary with a neutral value. Using the values of the tokens and other statistical data (like number of all letters, number of spam letters, etc.) makes available to calculate the final probability of being a spam. The result is a number, but it could be easily turned into a binary value as well.

## 5.5 Feedback

After getting the final result, whether the e-mail was a spam or not, the stored values in the token-dictionary have to be changed. Note that a change is necessary only for the tokens that existed in the letter. This gives the continuous learning capability of the method, since tokens, which are often in spam letters, will get higher and higher values, while tokens related to legitimate mails are getting smaller values.

## 5.6 Training and different training types

Every Bayesian network based-solution needs a training period. If the tokens were having neutral values, like just after adding them to the token-dictionary, it would not be possible to make correct decisions in the first times. Training could be achieved by two different ways. First, by manipulating the values in the token-dictionary, to strengthen or weaken the effect of the words. This solution is more ‘ad hoc type and leads to unpredictable results. The second method, a more adequate training could be achieved by adding pre-categorized “spam” and “legitimate” messages on the input of the filter. In this case the algorithm skips the calculation part; the usage is limited to the feedback, since the program gets the results in the same time with the letters. This training method uses the steps, described in the 6.5.1 Table in part II. Repeating the above described sequence for bigger amount of e-mail will fine-tune the values in the token-dictionary. As a result, the words relating to spam will get higher values and legitimate letter indicators will get lower ones. Note that words that are very often in both spam and legitimate messages, will keep their neutral value and so it will not play an important role in the decision.

## 5.7 Testing

After the learning part is it is possible to test the filter. It is important to note that testing the filter before training makes no sense. Although a filter learns further during its usage and gets more and more personalized, to compare the result of different filters there is a need to train them with the same e-mail set before testing. There is always the opportunity to review and correct the result of the filters, achieving better functioning in the future. That is called the users’ feedback and it helps to achieve better machine learning.

## 6. Concrete solution of the Bayesian algorithm

### 6.1 Overview

This chapter contains an algorithm that provides the basic of Bayesian classification in the field of spam filtering. The chapter does not deal with the problem of pre-processing, since this algorithm works with exact string matches. This solution uses all values from the token dictionary, while in everyday usage due to constrained computing resources filters are working with only a limited number of values.

### 6.2 The token dictionary

First of all, there is a need for a wordlist, a token dictionary. In best case the token dictionary is being updated with new words (tokens) after every processed e-mail. For simplicity let us now reduce the scope to a fixed dictionary, with no updating. It means that the dictionary is pre-created and no additional tokens are added later. In this token dictionary two values are recorded for every entry (every token). Let them call “ $N_1$ ,  $N_2$ ”. All are natural numbers (counters). Note that the numbering must be started from one to avoid dividing by zero in the future calculations.

6.2.1 Table: Meaning of the values recorded for each token in the dictionary

$N_1$	No. of <b>spam</b> letters, where the word <b>has existed</b>
$N_2$	No. of <b>legitimate</b> letters, where the word <b>has existed</b>

When a new e-mail is received, the token dictionary is searched for all of the words, included in the letter. (Note: this algorithm follows word-by-word tokenization.) Two sets of words are handled: one is the set of words matching the dictionary (an update will be necessary at value  $N_1$  or  $N_2$ ), other is the set of words not included in the letter (no update will be necessary).

The two sets are built up by every processed e-mail. It could be implemented, for example, with a binary flag that can be switched to 1 for every word, which belongs to the first set and kept 0 for every others. (Note: the flag must be reset after every processed e-mail.)

Furthermore besides the token dictionary two numbers must be stored, the number of all received spam letters and the number of all received legitimate letters. Let them call “SPAM, HAM”. (Since legitimate letters are also called ham.)

### 6.3 Getting the result

The final result of the filtering has the follow form:  $Odd = \text{Ratio of } P / (1-P)$ . This formula is a standard idealization of to express odds. In this case it looks like:

$$P ( \text{“letter is a spam”} | \text{“matching words”} ) /$$

$P ( \text{“letter is a legitimate”} | \text{“matching words”} )$ , that means the probability of the currently filtered e-mail to be a spam divided by the probability of being legitimate, taking into consideration the words included in the letter, more precisely the values of the words included in the letter. Here the “matching words” phrase means the words what are existed in both the letter and the dictionary. (Note, that in this case word-by-word tokenizing was done.) Since the result is a quotient of two probabilities, it will be a number. This value can be turned into a binary result. If the result is over 1, the probability of spam is higher than the legitimate, what means the letter is spam. If the result is between 0 and 1 the legitimate classification has higher probability so the e-mail should be categorized as legitimate. There are many other possible usage of the result, related to section 4 in part II, where the different actions were defined.

### 6.4 Training

Like every Bayesian network-based algorithm there is an obvious need to train the filter before usage. In this case the result probability is given by the user or simply the message is pre-classified, and the value modifications (mentioned below) in the token dictionary are exactly the same, as if the final probability would have been calculated by the algorithm (detailed in the section 6.6, part II).

## 6.5 Testing and using the filter

By testing and also by using the filter, first the resulting probability is calculated the way described below, and according to its result the records in the token dictionary are modified. At this point it is already known for each word in the dictionary, whether they are included in the e-mail or not and after calculating the resulting probability the filter classifies the e-mail being a spam or a legitimate. After all these activities one value is incremented at all entries of the dictionary, which were included in the letter; according to the facts whether the letter was a spam or not. The two different outcomes are the following:

6.5.1 Table: Performed actions at the token dictionary upgrade

Observation	Action
The letter was marked as spam and the word was in it.	Increment $N_1$ by 1.
The letter was marked as legitimate and the word was in it.	Increment $N_2$ by 1.

Note that no change is required in the token dictionary at the words, which were not included in the letter. Moreover these non-matching tokens are not used in the calculation part. After the letter was categorized, one of the SPAM or HAM values must be also incremented by 1.

## 6.6 Calculation

The calculation of the final value is done as follows: first the value is initialized to one (for the case, where none of the words are matching from the token dictionary). The calculation involves the steps listed below:

- ALL (No. of all e-mails) = SPAM + HAM (number of legitimate letters, added to the number of all spam letters).
- Let us call a word “matching word”, if the word has existed both in the letter and in the token dictionary.
- $P(\text{“matching words”} \mid \text{“letter is spam”}) = \prod_{\text{for all matched word}} (N_1 \text{ value of the current word} / \text{SPAM})$

- $P(\text{"matching words"} \mid \text{"letter is legitimate"}) = \prod_{\text{for all matched word}} (N_2 \text{ value of the current word} / \text{HAM})$
- $P(\text{"letter is spam"}) = \text{SPAM} / \text{ALL}$
- $P(\text{"letter is legitimate"}) = \text{HAM} / \text{ALL}$
- $P(\text{"letter is spam"} \mid \text{"matching words"}) =$   
 $= P(\text{"letter is spam"}) * P(\text{"matching words"} \mid \text{"letter is spam"})$
- $P(\text{"letter is legitimate"} \mid \text{"matching words"}) =$   
 $= P(\text{"letter is legitimate"}) * P(\text{"matching words"} \mid \text{"letter is legitimate"})$
- Final result:  $P(\text{"letter is spam"} \mid \text{"matching words"}) /$   
 $P(\text{"letter is legitimate"} \mid \text{"matching words"})$

## 6.8 Concrete example

The next section gives a made-up example, to explain how Bayesian filter works. Let us imagine, in the database the result of 240 spam and 814 legitimate letters are stored (1054 letters in total). It means the spam rate is 23%, while the legitimate rate is 77%. The received letter is quoted below:

-----  
 Date: 19.05.1908 11:14  
 From: Kathrine Witt <alm@0451.com>  
 Subject: All products for your health!

<http://pcvija.seescum.biz/?70573075>

Suffering from pain, depression or heartburn? We'll help you!

All verified dr@gs collected at one LICENSED online store! Great choice of wonderful meds to give you long-awaited relief! Operative support, fast shipping, secure p@yment processing and complete confidentiality!

The store is VERIFIED BY BBB and APPROVED BY VISA!

<http://pcvija.seescum.biz/?70573075>  
 -----

The spam filter separates the letter into tokens and the values of the tokens are looked up in the database. The meaning of the  $N_i$  numbers is described in part 6.2. The table below shows the tokens and values. Many different conclusions can be made based on the values in the table. For example the words *for* or *you* are neutral and they exist in spam just as in legitimate. However *pain* or *dr@gs* are obvious marks of spam. Additionally the W/S (word has existed/spam) column means the number of spam messages, where the current matched word has existed divided by the number of all spam letters. The W/H (word has existed/ham) column means the number of legitimate messages, where the current matched word has existed divided by the number of all legitimate letters. Note that in order to avoid dividing or multiplying by zero, the numbers start from one, instead of zero. Also note that the last two columns contain rounded numbers.

6.8.1 Table: The example token dictionary

Word	$N_1$	$N_2$	W/S	W/H
all	54	512	0.23	0.63
products	22	210	0.09	0.26
for	129	447	0.54	0.55
your	98	398	0.41	0.49
health	81	23	0.34	0.03
Suffering	20	2	0.08	0.00
from	57	306	0.24	0.38
pain	77	14	0.32	0.02
depression	34	7	0.14	0.01
or	126	583	0.53	0.72
heartburn	1	1	0.00	0.00
we'll	18	39	0.08	0.05
help	60	100	0.25	0.12
you	142	517	0.59	0.64
verified	1	3	0.00	0.00
dr@gs	37	1	0.15	0.00

The next step is to calculate the  $P$  ( “letter is spam” | “matching words” ) probability by multiplying  $P$  ( “letter is spam” ) – that is the 23% and  $P$  ( “matching words” | “letter is spam” ) – that is the product of the W/S column. After, the same calculation is done for  $P$  ( “letter is legitimate” | “matching words” ) what uses the 77% legitimate rate and the W/H column.



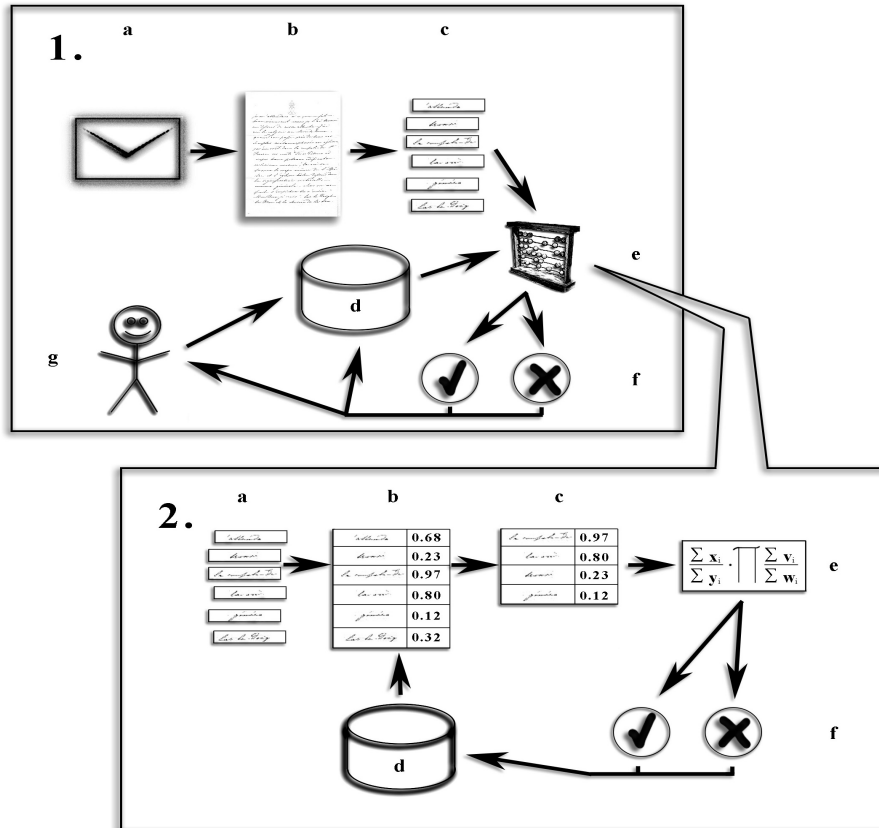
The final result comes with the quotient of the two above calculated probabilities. If the letter is more like spam, the result will be more than 1. If the letter is more legitimate, the result will be between zero and one. In this example the result is around 29million what strongly determines that the e-mail should be categorized as spam.

## 6.9 Flow Chart of the Bayesian spam filter

- 1a: the letter is received.
- 1b: it is used as a plain text (including body and header).
- 1c: the letter is segmented into tokens (tokenization).
- The stored values are looked up from 1d database.
- 1e: the calculation is done.
- 1f: the result is sent to the user, while it updates the stored values in the database (1d).
- 1g: the users' feedback in case of misclassification, which helps to update the database.

The calculation part is detailed on picture 2.

- 2a: the tokens are received.
- 2b: from the database (2d) the tokens get their stored values.
- 2c: only the most relevant tokens are used further.
- 2e: the calculation is done using the values of the most relevant tokens and other statistical data.
- 2f: the final decision is made, and the database (2d) is updated relating to the result.



6.9.1 Figure: Flow chart of the Bayesian spam filter

## 7. Preparation for comparing spam filters

### 7.1 Overview

After recognizing the necessity of spam filtering, the next very important step is to find a suitable filter. Already comprehensively used programs were compared, with the hope that the results of the comparison will help to find the filters of our future or to create a meta-filter from the best parts of the different, existing filters. Since most spam letters are in English, it seems a logical choice to reduce the training and learning parts to English e-mail only. The goal is to compare the filters by several different ways, in order to achieve better results.

It is important to note the fact that most spam filters are calibrated a little off-balanced. It means that making of a mistake has not the same probability for spam and legitimates. The explanation for this fact is that to get a spam in our inbox (also called *false negative*) is less bad, than putting an important letter to the spam folder (also called *false positive*).

## 7.2 The corpus

To be able to compare and analyze the results of the filters, first of all there is a need for a large amount of e-mail. Furthermore, the e-mail corpus has to be separated into sets of legitimate and spam, in order to make the training possible. The first idea to get a corpus was to search for newsletter or mailing list archives, but unfortunately it did not lead to a satisfying result, since hopefully there were no spam letters at all or only very few; and these e-mail archives may contain information only about a specified area or subject.

After a significant research and analysis of different corpuses, it appeared that the best should be the corpus of the SpamAssassin open source spam-filtering program, which is reachable for everyone. [21] There are not only binary separations, but the e-mails are already classified into 3 groups: spam letters, easily recognizable legitimate letters, and hardly recognizable legitimate letters.

7.2.1 Table: The amount of letters contained in the used corpus

Easily recognizable legitimate	10647
Hardly recognizable legitimate	153
Spam	2176

Such a large number of messages gives the opportunity to work with different training and testing sets. Training could be successful, since the corpus covers a lot of different subjects and themes. As it later turned out, from some aspect the corpus seemed more than enough what led to train and test the filters only with a part of the corpus.

## 8. Choosing the spam filters

### 8.1 Overview

For testing and comparing purposes widely used programs were chosen that already have proved their appropriateness. Fitting to the users' needs, it is worth to compare freeware programs. Although there are quite lot of different filters on the market, due to the limited resources, only three of them were compared. In order to support adequate result, big efforts were made to choose the most relevant and most widely used filters. The chosen filters are the following:

### 8.2 SpamAssassin [33]

The most known and widely used open source program SpamAssassin was first published in 2001. That time it used only heuristic rules, but it has recently adopted a Bayesian filter part to its decision matrix, in an effort to boost accuracy. *SpamAssassin is generally regarded as one of the most effective spam filters, especially when used in combination with spam databases.*[60] SpamAssassin is easy to parameterize for special and individual needs, and could be used together with other programs. From single users to bigger companies a large scale of different needs is satisfied by this solution. It was natural to choose SpamAssassin for testing.

### 8.3 Mozilla Thunderbird [31]

The second program was Mozilla Thunderbird. After recognizing the Internet Explorer's weaknesses more and more users are changing to Mozilla or Firefox web browsers. As a result of this changing flow, people tend to switch from Outlook and Outlook Express to Mozilla or Thunderbird to send e-mail. Mozilla's product has an already implemented, well-designed filter. The filter gives no automated opportunity for training, but it learns quickly manually trained and gives also a satisfying result in everyday situations. Wikipedia [55] gives the following description about the built in spam filter system of Thunderbird:

“Thunderbird incorporates a Bayesian spam filter, a white list based on the included address book, and can also understand classifications by server-based filters such as SpamAssassin.”

## 8.4 SpamProbe [34]

The third program was SpamProbe, another open source spam filter, which is a statistical spam filter, based on Paul Graham's research. SpamProbe implements 27 tokens in the decision matrix and it also allows duplicate entries in the matrix. Since the decision matrix deals only with a very limited number of tokens, only the most meaningful data is used to populate the decision matrix. This allows the analysis engine to focus on the most useful parts of e-mail without being confused by less important text. It features also multiword tokens based on the concept and uses a modification of Graham's original algorithm to calculate the final probability.

“Instead of using pattern matching and a set of human generated rules SpamProbe relies on a Bayesian analysis of the frequency of words used in spam and non-spam emails received by an individual person.” [11]

## 9. Training and testing filters

### 9.1 Overview

Like all of the Bayesian network-based solutions, there is a need for training the software before use. Finding the ways of training is not only important to prove the filters' capability of classification, but there is also a need to train the different filters in the same way to make the comparison feasible. Pre-classified e-mails are added to the inputs of the programs. During the training process the values of the tokens are changed time by time in the token-dictionary of the filters in order to achieve higher and higher accuracy. It makes the spam recognizable for the programs. There is a need to note that the learning does not stop after finishing the training, it continues every time when the filter is used. Moreover, the filter gets more and more personalized and related to our needs during usage.

### 9.2 The different training methods

All the Bayesian-based filters need training and the training could be done in several different ways. The main principles in training are listed below:

“Train on everything” or “unsupervised” learning: all the new tokens are added to the dataset, and the values of the tokens are updated after each processed e-mail. This solution has the ability to follow quick changes in e-mail characteristics, but it needs significant resources after every received e-mail for handling all the tokens and the rapidly changing values.

“Train on error” learning: it follows the idea that rapidly changing values could lead to more errors. In this case the values of the tokens in the dataset are modified only if an error is noticed due to the user’s feedback. It deals with a significantly lower disk writing need, compared to the “train everything” solution. The negative side of the learning method is the lack of quick adaptation for new characteristics. For example a new kind of spam can take a while until being correctly handled.

“Train until mature” learning: this solution gives the composition of the two methods listed above. It follows the ideas of “train on everything” in the first section and after getting enough knowledge about the user’s habits it trains only if an error is occurred, like in “train on error”. This combined learning method gives the advantages of both above listed learning solutions, the only problem is to define, when the filter already got enough knowledge to change the learning method.

### **9.3 Determination of training and testing sequences**

In this comparison “train on everything” learning method was used. Note that in this comparison training was done after every e-mail, but in real life situations it should be done only daily, to keep resources for peak hours. For a single user daily training is not a really big advantage, but at company level it could be really useful in order to free calculating and disk handling resources.

At the comparison the first training rate was the widely used 70-30 percent, that means the filters are trained with the 70% of the corpus (this 70% of the corpus contains randomly selected messages, but the different filters got the same random part from the corpus). The rest 30% of the corpus is used for testing. At the testing part, the filter is used as a real user would use it. The filter uses the knowledge base from the training part to decide for every e-mail, whether it is spam or legitimate.

After every processed e-mail the filter generates a binary result. In our case 1 means the letter was categorized as spam and 0 means the letter was categorized as legitimate. This result is a binary vector what can be compared with the results of other filters and of course with the original classification. The results were above expectation; the filters worked satisfactorily, which could be caused by the huge number of training messages.

Due to the efficiency of the filters were over expectation on the whole corpus, other training/testing rates were worthwhile to deal with. This consideration led to the second idea, to train only with the 40% of the corpus, what allows testing with 60%. Although in this case the filters were tested with the 60% of the corpus, in order to achieve full comparable results, only the results of the 30% of the corpus were used. This 30% contains the same e-mail set that was used in the 70-30 percent training. This cut down at the result vector was necessary in order to let the filters be comparable also over different training types. For example, mail#1 was tested with SpamAssassin, when the software was trained with the 70% of the corpus and mail#1 also was tested with the same software, when the filter was trained only with 40%. (Note: that between the different training sequences, the knowledge base of the filters was erased.) After all of these considerations, not only the different filters were comparable, but the filters with different training amounts became comparable. In so far the statistical results with the 40-60 percent training and testing were still very suitable. More details about the results of the filters are published at the statistical results, chapter 11.2 in part II.

Finally, to try a drastically altered training/testing rate, the filters were trained with only 10% and tested with 90% of the corpus. Either the corpus seemed to be really large, or the filters were really well-designed, but even so there could be seen no considerable differences. In order to still let the filters be comparable again the 30% of the corpus was separately handled, to achieve the above mentioned comparison.

## 10. Analysis of the results

### 10.1 Part of the results

10.1.1 Table: An example part of the result (filename: spam2.csv)

Mail No.	SA-10	SA-40	SP-40	SA-70	SP-70	ORIG
323	0	0	0	0	0	0
1197	0	0	1	0	1	0
1187	1	1	0	1	0	0
1198	1	1	1	1	1	0
1305	1	1	1	1	1	1
1318	0	1	1	1	1	1
1365	0	0	1	1	1	1
1431	0	0	1	0	1	1
1547	1	1	0	1	0	1

The table is a part (different rows) from the result; it shows the typical types of mistakes. The first row is the header. Mail No. means, which processed e-mail, belongs to the row. SA is for SpamAssassin and SP is for SpamProbe, while the numbers shows the training set, for example SA-10 means SpamAssassin trained with 10%. The column ORIG is the original classification of the e-mail, zero means legitimate and one means spam. All other values in the table are the output of the filters.

### 10.2 The characteristics of the results

The typical types of results are the following: Mail 323 is where the filters made no mistakes, classification was correct (this result was get for the most of the messages). Mail 1197 is a message, which SpamProbe categorized wrong, independently of the training amount. Mail 1187 is the same like the one before, but now SpamAssassin could not handle it correctly. (Even if the training set was the same, there are significant differences between the filters.) Mail 1198 is one, which none of the filters were able to classify.



Mail 1305 is an originally spam message which was correctly classified by all filters. Mail 1318 is one, which SpamAssassin categorized wrong with 10% training set, but with 40 and 70 percent it succeeded, just as SpamProbe. At mail 1365 SpamAssassin needed 70% training, and could not be made a correct decision with less, while SpamProbe handled it correctly. Only SpamProbe classified mail 1431 correctly, while only SpamAssassin categorized mail 1547 correctly. Even if the filters made more than 80% of the classification correctly, these above mentioned typical mistakes appeared randomly. These different mistakes make the ground for further analysis of the results, since not only the number of mistakes is important but also the place of them.

Despite the filters made some significant differences at filtering, the effect of the different training amount was not enough detectable. To draw the inference, the huge number of e-mail messages could cause the small differences between the filters. Even at 10 percent training the training set seemed to be too big (approximately 1300 letters). To make the differences clearer and more detectable, it seemed obvious to try reducing the body of the corpus. The 70% of the corpus was passed by, and the above listed methods were tried using only the 30% of the original messages. The messages were selected randomly. Even though the corpus became significantly smaller, the filters still worked with high accuracy.

### **10.3 Another training and testing possibilities**

Since the Thunderbird program was already installed and used on two computers, a new testing method seemed to be interesting. It was testing without training. The software had only the knowledge got from everyday usage. The main point of this testing was not to check the filtering capability of the software without training, but to get more knowledge about the accuracy rate of filters trained only in every-day usage. It is important in order to avoid errors in the analysis due to a specific corpus, or not life-like messages used at comparison. As it later turned out, the filter with the 2year old knowledge base (even if most of the earlier processed letters were not in English but in Hungarian) could really catch up with the specifically trained filters.

## 10.4 Summary of the performed training and testing sequences

On the next figure are shown the results of the comparison of the tested filters and different train-test rates. The mark “x” means that the testing was done for that sequence.

10.4.1 Table: Training and testing sequences on the whole corpus

	70% training, 30% testing	40% training, 60% testing	10% training, 90% testing
SpamAssassin	X	X	X
SpamProbe	X	X	

10.4.2 Table: Training and testing sequences on the reduced corpus

	70% training, 30% testing	40% training, 60% testing	10% training 90% testing	0% training, 100% testing
SpamAssassin	x	x	x	
SpamProbe	x	x		
Thunderbird pc#1	x			x
Thunderbird pc#2	x			x

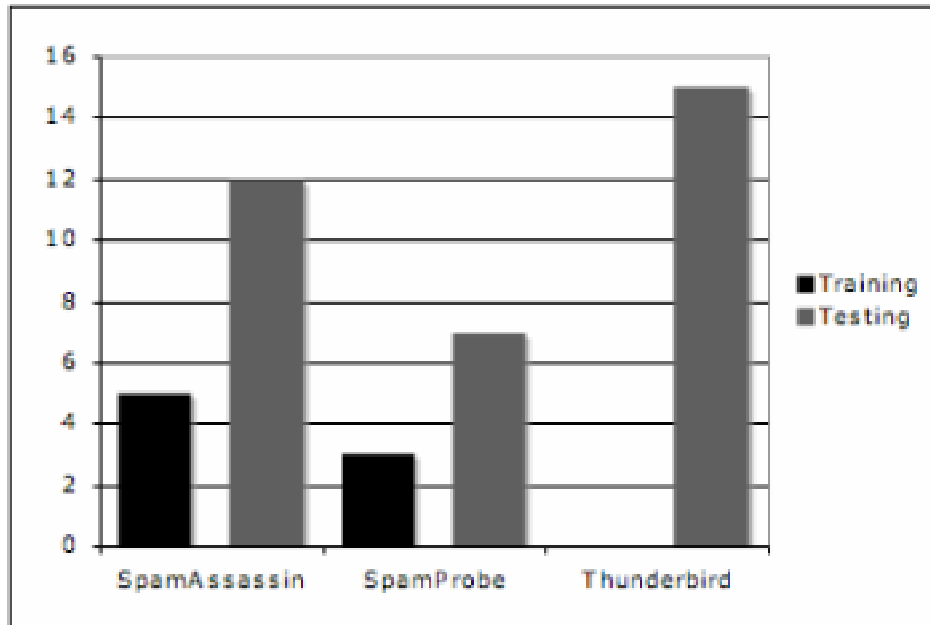
## 11. Statistical analysis

### 11.1 Time factor

The running times could not be left out of consideration, since it could be a main aspect by choosing the adequate program, especially for bigger companies. The training time compared to the testing time is also very important. At everyday usage, there is no need to train the filter anymore, so only the testing sequence will be done regularly. Running times for the different programs on the 30% of the corpus are shown on the table below. Note that the training sequence on Thunderbird was done manually, therefore no training time is available. The training and testing times were measured on an Apple laptop computer (CPU: 1.2GHz, Memory: 768MB) except the Thunderbird #2, what was made on a desktop PC (CPU: 2.4GHz, Memory: 512MB).

11.1.1 Table: Training and testing time

	70% training,	30% testing
SpamAssassin	5 min	12 min
SpamProbe	3 min	7 min
Thunderbird #1	n/a	15 min
Thunderbird #2	n/a	11 min



11.1.2 Figure: Training and testing time

The table shows that SpamProbe was the fastest in all aspects. The second was SpamAssassin, where the training and the testing times show a significant c.a. 70% more required time. The slowest seemed to be Thunderbird, what needed more than 100% extra time compared to the test-winner SpamProbe. (Note: while the first 3 cases were done on the same computer that has given a ground for comparison, Thunderbird #2 was tested on a different computer.)

## 11.2 Number of mistakes and error rate

The error rate is a way to analyze statistical results. At the comparison the number of mistakes is divided by the number of all processed letters. The error rate could be also called failure-rate. For the error rate results only the number of mistakes is relevant, and the place of mistakes is not taken into account.

As it was expected, much more spam letters could be passed through while only a few legitimate letters were filtered out. This is in accordance with the fact that filters are fine tuned under the principle that to get a spam in the inbox is less significant error than to put a legitimate e-mail into the spam folder.

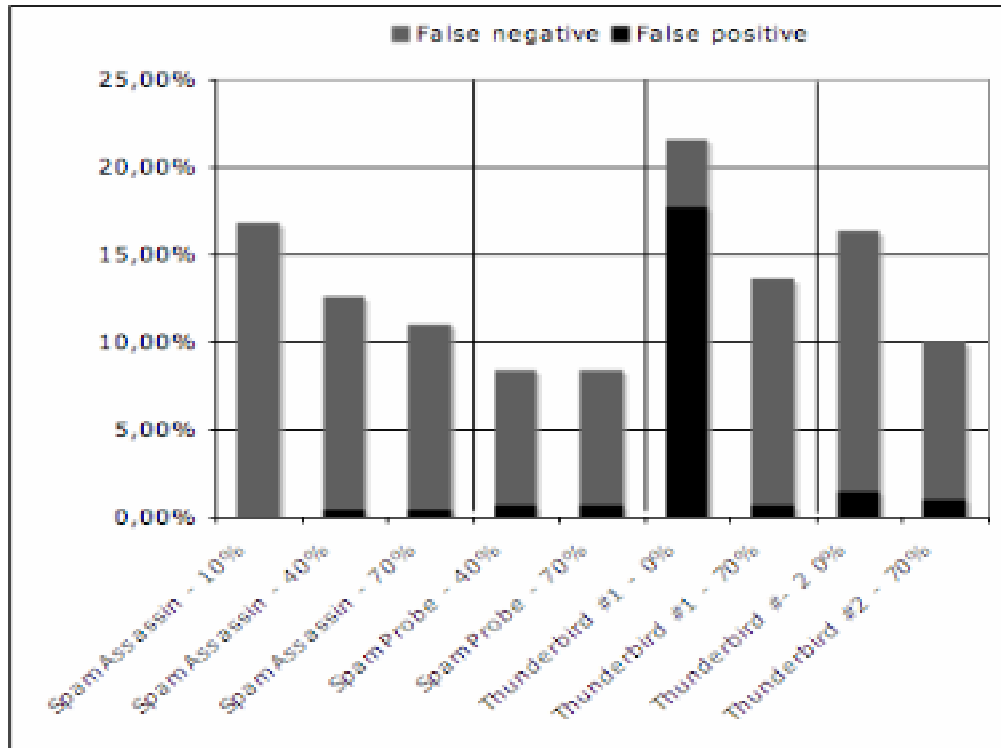
Another recognizable fact is that the Thunderbird on the computer that has been used only for 2 months (Thunderbird #1), generated very disappointing results without training. That is fairly close to the expectations, since the software was used mostly with Hungarian messages and so the spam filter could not handle English letters. However, after training the results became comparable with the other filters. The spam that could bypass the filter is called also false negatives and the legitimate letters what were filtered out are the false positives. The next two tables show the number of mistakes made by the filters and the error-rate:

11.2.1 Table: Number of mistakes on a corpus of 603 letters (413 legitimate, 190 spam)

	Training	Let through spam	Filtered legitimates
SpamAssassin	10%	32	0
SpamAssassin	40%	24	2
SpamAssassin	70%	21	2
SpamProbe	40%	16	3
SpamProbe	70%	16	3
Thunderbird #1	0%	41	73
Thunderbird #1	70%	26	3
Thunderbird #2	0%	31	6
Thunderbird #2	70%	19	4

11.2.2 Table: Rate of mistakes on a corpus of 603 letters (413 legitimate, 190 spam)

	Training	Let through spam	Filtered legitimates
SpamAssassin	10%	16,84%	0,00%
SpamAssassin	40%	12,63%	0,48%
SpamAssassin	70%	11,05%	0,48%
SpamProbe	40%	8,42%	0,73%
SpamProbe	70%	8,42%	0,73%
Thunderbird #1	0%	21,58%	17,68%
Thunderbird #1	70%	13,68%	0,73%
Thunderbird #2	0%	16,32%	1,45%
Thunderbird #2	70%	10,00%	0,97%



11.2.3 Figure: Rate of mistake in percentage for both false negatives and false positives

The figure above shows the results. All the different filters (separated by vertical lines) show a non-increasing tendency thanks to the bigger training amount from left to right. It was expected that for the Bayesian filters bigger training amount leads to higher accuracy. It is not obvious which filter should be called the best. On one hand SpamAssassin shows a very fast increasing accuracy, which allows the implication that SpamAssassin with more and more training could lead to higher and higher accuracy. On the other hand, SpamProbe follows a very balanced result, not even the higher training rate could lead to higher accuracy. It is important to note that SpamProbe produced the best results and also it was the only filter with an accuracy rate higher than 90%. Thunderbird produced an average result between the filters, but still is the easiest to use, since it is a part of an already implemented e-mail client. Thunderbird on the computer without sufficient earlier knowledge base produced a disappointing result, but it should not be taken into consideration since the example (testing without training) is not relevant for every-day life situation.

An interesting thing is that more training does not definitely lead to higher accuracy, for example the rate of false positive detected was grown at SpamAssassin with the bigger training sets. It could be caused by the not perfectly representative corpus. However, 2 outfiltered legitimate letters from 603 still do not make the ground for disappointment. Moreover, it is really important to notice, that Thunderbird with a bigger earlier knowledge base produced significantly better results after 70% training, than the one with almost none earlier knowledge base. Earlier trained Thunderbird after the 70% training also reached the 90% accuracy rate, just like SpamProbe.

### 11.3 Conclusion

My conclusion is that the tested filters have pros and cons. A quick overview of them is shown in the next table.

11.3.1 Table: Conclusions about the filters

Filter name	Advantages	Disadvantages
SpamAssassin	Good learning capability	Fairly bad results with small training set
SpamProbe	Good results also with smaller training set	Bigger training set does not led to higher accuracy
Thunderbird	Easy to use	Operates with longer running time

As it could be seen, it is not obvious how to choose the adequate spam filter. There are many different points of views that should be taken into consideration. Moreover, a forecast could be made that the filtering with only one software could not lead to such a high accuracy than filtering with more. The above mentioned deliberation has built the base for the next part of the thesis, in order to use different filters together and to establish a meta spam filter.

# PART III

## 1. Introduction

The goal of the third part is to create a meta filter and achieve higher accuracy by using filters together. First the BNET<sup>1</sup> mathematical modeling software is introduced. Bayesian networks were created showing the coherency of the filters. The BNET gives the possibility to learn structures. With this method not only the number of errors taken by the filters, but also the characteristics of the errors i.e. the regions of errors are taking account in the comparison of filters. By analyzing the generated Bayesian networks it could be figured out, which filters are efficient to be used together and which not.

Once it is known, what filters should be used together; meta filter models (constructed in the BNET) became comparable with a ROC analysis. With this classification measure the differently parameterized meta filters became comparable. The final result is a tested suggestion, which easily usable spam filter programs should be used together to achieve a higher accuracy in filtering.

## 2. Results analyzed with Bayesian networks

### 2.1 Overview

With the BNET software the filters could be compared to each other and to the expected original results. Since not only the number of mistakes is relevant, but also their characteristics i.e. the regions of errors in the input space. Every filter's results can be viewed as a binary vector and also the original or expected values are a binary vector. These vectors were arranged in a comma-separated file (csv), what is a standard input for BNET. The program uses the filter test results form the csv files and generates Bayesian networks. These networks show the relevancy between filters.

---

<sup>1</sup> BNET software has been developed by Péter Antal

The main goal is to analyze the constructed Bayesian networks in order to have a clear look on connections between the filters. If two filters show very similar results they should not be used together in a meta filter, while if they both give high accuracy but they are not similar (they make the mistakes at different locations), there is an obvious chance to use them together in order to achieve a more accurate meta filter.

## 2.2 The BNET software

Using the BNET software, there is an opportunity to settle up self-constructed Bayesian networks using the binary results of the filters. The program uses the binary vectors (the results of the filters and the original values). In these binary vectors the value 1 means spam and value 0 means legitimate. The software analyzes the data set containing the vectors. If there is a significant direct dependency relation between the variables, there will be a connection between the variables. In graphical representation it means, that there will be an arc between the nodes. By this way the structure of the Bayesian network will be learnt. Unfortunately the structure returning algorithm is an exhaustive search with super-exponential complexity in the number of variables. [54]

## 2.3 The CSV file format

The .csv extension signs a comma-separated file. This file structure is well known and widely used. It is a plain text file, just like a .txt file with some constraints. These constraints are the follow: it consist a two-dimensional table, a matrix. The first row shows the variable names while the others are responsible for the values, related to the variables above. To generate the .csv file, the result binary vectors of the filter testing were simply put next to each other separated by commas, as the example shows below.

```
-----
SA-10,SA-40,SP-40,SA-70,SP-70,ORIG
0,0,0,0,0,0
1,0,0,0,0,0
0,0,0,0,0,1
1,1,0,1,0,1
-----
```



## 2.4 Structure learning

The program detects the dependency structure between the filters output and the original values, and generates a Bayesian network. In the generated Bayesian network every node represents a variable. These variables are the binary vectors generated from the output of the filters and the original classification of the letters. The lines between the nodes are the arcs and each one represents a connection. These connections (arcs) have directions, but in our case it is irrelevant. A connection is settled between the variables if they show a significant dependency. For example if a node is connected with the node of the original values, than the inference could be drawn that the filter represented by the node produced high accuracy rate. If the node of a variable is connected another variable's node, than the two variables produced a similar result. These two variables can be different filters, or the same filter with different training sets.

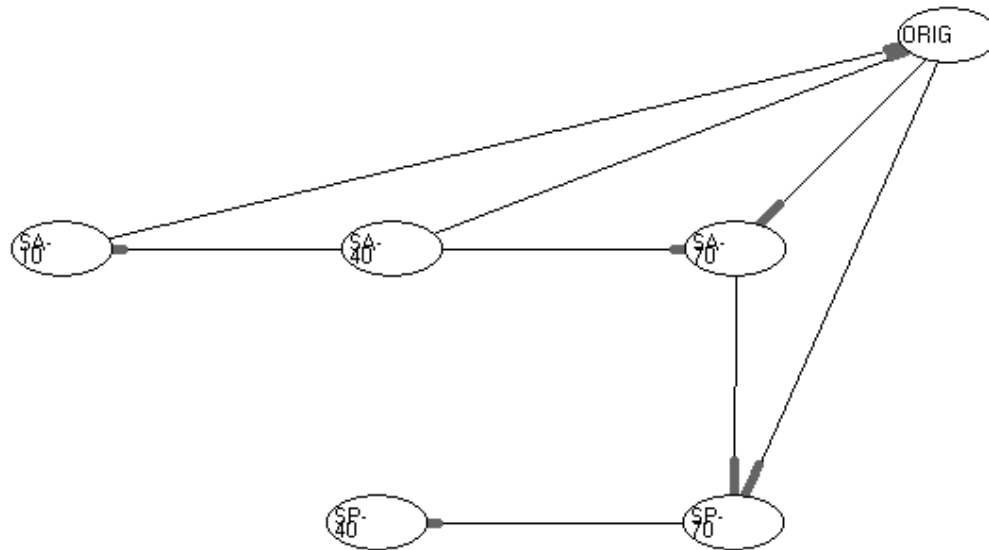
## 3. Results of the structure learning

### 3.1 Overview

The self-constructed Bayesian networks need a detailed analysis in order to get the most information out of them. This information can be used to explore the relations between the filters and training sets, and helps to understand what is important for further usage, what is important to construct a meta filter.

### 3.2 Analyzing the results of the first part – whole corpus

The first tests were done with the whole corpus. In these cases two spam filter programs were used, SpamAssassin and SpamProbe. The training and testing sequences, what had been done were described in section 10.4 in Part II. The results of the filters and the original classification of the letters established binary vectors. These binary vectors were used to build up the comma-separated file and this file was the input of BNET. With the software an exhaustive search was made and structure was learnt. On the next image is the structure of the Bayesian network, generated by the program:



3.2.1 Figure: The structure of the self-constructed Bayesian network – whole corpus

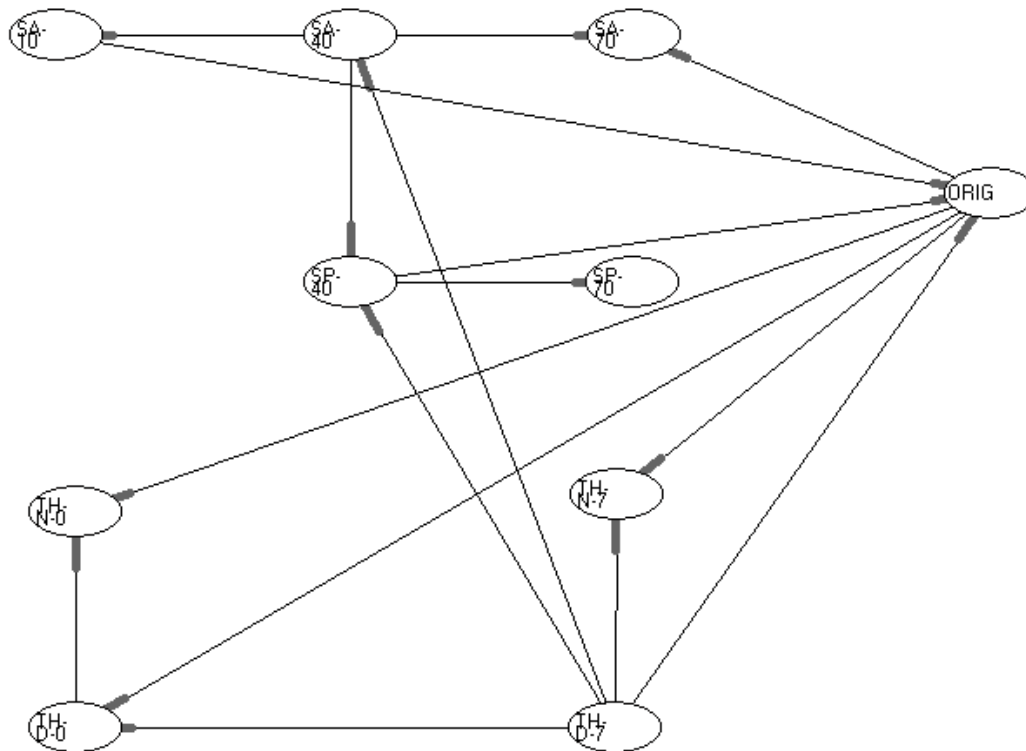
3.2.2 Table: Legend to the 3.2.1 Figure

	Legend
SA10	SpamAssassin, 10% training
SA40	SpamAssassin, 40% training
SA70	SpamAssassin, 70% training
SP40	SpamProbe, 40% training
SP70	SpamProbe, 70% training
ORIG	Original class for the e-mail

On the image there is the generated result. It shows that almost all nodes are connected with the original, what caused by the high accuracy rate of the filters. The explanation is that the corpus was really huge, and so the testing subsets were also more than satisfying. This fact led to the idea to try the same analysis with a smaller, reduced corpus. Another thing, that could lead to the large number of connections is, that there were more legitimate letters in the corpus, and since the filters made almost no mistakes at legitimate messages the “legitimate” part of the binary vectors were long and almost totally equivalent for every filter.

### 3.3 Analyzing the results of the second part – reduced corpus

After the removal of the two-third of the corpus other tests were done. The test were made with three filters: SpamAssassin, SpamProbe and Thunderbird. The next image shows the structure made by BNET:



3.3.1 Figure: The structure of the self-constructed Bayesian network – reduced corpus

3.3.2 Table: Legend to the 3.3.1 Figure

	Legend
SA10	SpamAssassin, 10% training
SA40	SpamAssassin, 40% training
SA70	SpamAssassin, 70% training
SP40	SpamProbe, 40% training
SP70	SpamProbe, 70% training
THN0	Thunderbird pc#1, 0% training
THN7	Thunderbird pc#1, 70% training
THD0	Thunderbird pc#2, 0% training
THD7	Thunderbird pc#2, 70% training
ORIG	Original class for the e-mail

In the figure is shown that most of the tested cases are related to the original values. It means that the filters still produced high accuracy rate and additional reason could be the above mentioned fact, that the filters made fewer mistakes at the longer “legitimate” part. What is really important in the figure, that there are dead-end nodes, like SP70, what means SP70 could be easily replaced by its parental node. A dead-end node does not carry more information than its parent. A variable represented by a dead-end node should be by-passed.

### **3.4 Additional analysis, in order to realize the meta filter**

The next chapter explains the further analysis of the results published in the previous chapter, in order to get closer to the outlines of the meta filter.

A mentionable fact is that there are cliques, where nodes are connected with all other nodes from the clique. A clique means that its members are similar to all others from the same group, so the members of a clique can be handled like being only one node. The largest clique in this example contains three nodes. These cliques are the following:

SA40, SP40, THD7

SP40, THD7, ORIG

THN7, THD7, ORIG

THN0, THD0, ORIG

THD0, THD7, ORIG

The first clique is interesting, since it does not contain the node ORIG what means these results are related to each other, with the lack of the fact that all of them would be related to the original. However for further usage those cliques are important, where the clique contains node ORIG, since the other two filters in the clique require further analyze in order to establish the best meta filter. If a filter generates similar result as the original than it should be taken part in the meta filter and moreover if a filter with smaller training set is connected to the ORIG just as the same filter with higher training set, then it means the filter should be the part of the meta filter with the smaller training set. These speculations help to outline the meta filter.

The most important case is, when in a clique besides the ORIG there are two nodes from the same filters. In this case only one of the nodes should be used in the meta filter since the usage of two similar nodes is just the waste of the resources. As if there would be only simple triangles, a node could be randomly passed by at the construction of the meta filter. Instead of simple triangles the image above is much more complex. It is not obvious which filters should be passed by. In one hand if a filter is selected to pass by in the future meta filter but it should not be passed by then the passing by would lead to a drop in accuracy rate. In the other hand if a filter is part of the meta filter but it should not be the part of it, then it would lead to the waste of resources. These factors together make the meta filter building process a very sensitive and fine-tune action.

## **4. The construction of the meta spam filter**

### **4.1 Basic ideas**

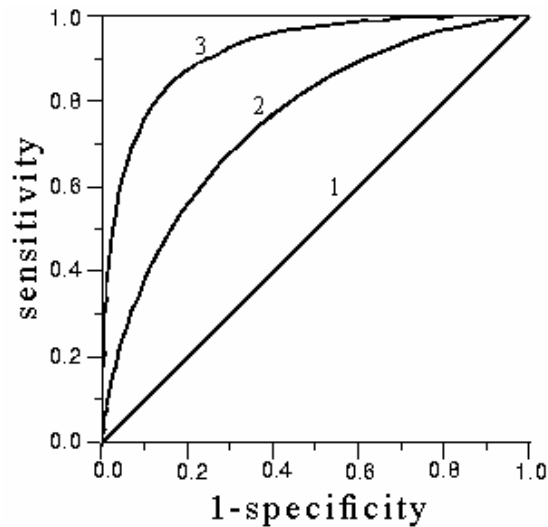
With all the above mentioned considerations in view, different meta models could be established. These different models should be tested with BNET using ROC analysis in order to show the relevancy of the meta filters.

### **4.2 The ROC-curve analysis**

To analyze the accuracy of the different meta filters the ROC is used. A receiver operating characteristic (ROC, also receiver operating curve) is a graphical plot of the sensitivity vs. (1 - specificity) for a binary classifier system as its discrimination threshold is varied. The ROC can also be represented equivalently by plotting the fraction of true positives vs. the fraction of false positives. [56]

The sensitivity of a binary classification test or algorithm is a parameter that expresses something about the test's performance.  $\text{Sensitivity} = \text{TP} / (\text{TP} + \text{FN})$ , where TP is the number of true positives and FN is the number of false negatives. Sensitivity can also be calculated by:  $P(\text{"letter is categorized spam"} | \text{"letter is spam"})$ . [57]

The specificity of a binary classification test or algorithm is the proportion of true negatives of all the negative samples tested.  $\text{Specificity} = \text{TN} / (\text{TN} + \text{FP})$ , where TN is the number of true negatives and FP is the number of false positives. Specificity can also be calculated by:  $P(\text{“letter is categorized as legitimate”} \mid \text{“letter is legitimate”})$ . [58]



4.2.1 Figure: Different ROC curves

The ROC can be interpreted as a measurement of learning capability. A completely random predictor would give a straight line at an angle of 45 degrees from the horizontal, from bottom left to top right: this is because, as the threshold is raised, equal numbers of true and false positives would be let in. This curve is marked with 1 in Figure 4.2.1. Results below this no-discrimination line would suggest a detector that gave wrong results consistently.

Every ROC above the 45 degrees straight line means a classification capability better than random. If ROC lies higher, it means a quicker learning cycle of the classifier. Or as restatement, the quicker the ROC reaches the maximum, the smaller training set is satisfying. However the characteristic of the classifier can be on the ROC seen, it is not easy to compare the curves, or define what quick raising means. To turn the curve into a number, the integral of the curve is used.

The ROC is often used to generate a summary statistic with the area under the ROC curve, what is called AUC (Area Under ROC Curve). The AUC value is simply the integral of the ROC curve. This measure can be interpreted as the probability of that when we randomly pick one positive and one negative example, the classifier will assign a higher score to the positive example than to the negative. Higher ROC curves lead to higher AUC values, which is an exact and comparable measurement of learning capability.

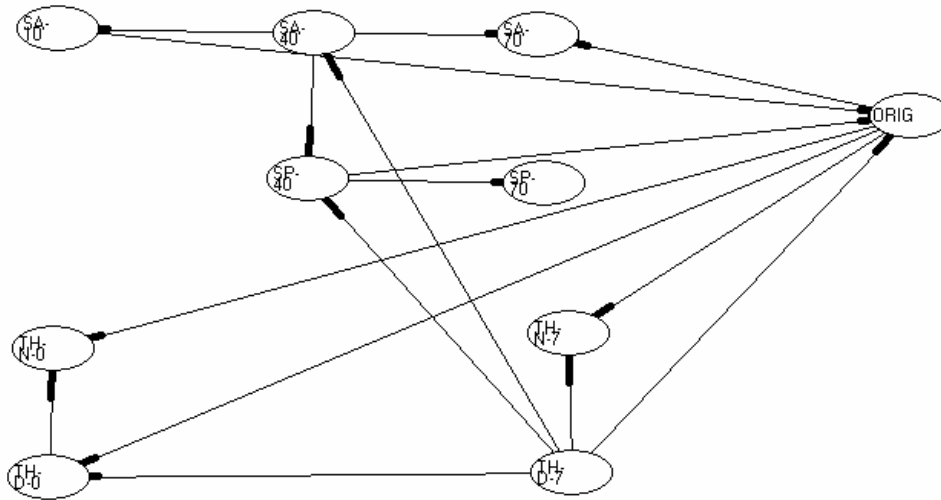
### 4.3 The meta models

The following section describes the construction ideas of meta models. These models are compared by using the AUC values of them. Since AUC is a measure of learning capability and at Bayesian spam filters the learning capability determines the accuracy of the filters, it seems reasonable to analyze the AUC values of the meta models. BNET gives an opportunity to calculate AUC values also for manually built up meta models. To calculate AUC of ROC only the target value must be defined. Since in this case the goal is to know how good predictions could be made on the ORIG node (the original classification), the target (or the output) is the binary vector of ORIG and all other values are the input for the ROC and AUC calculations.

4.3.1 Table: Legend to the following section

	Legend
SA10	SpamAssassin, 10% training
SA40	SpamAssassin, 40% training
SA70	SpamAssassin, 70% training
SP40	SpamProbe, 40% training
SP70	SpamProbe, 70% training
THN0	Thunderbird pc#1, 0% training
THN7	Thunderbird pc#1, 70% training
THD0	Thunderbird pc#2, 0% training
THD7	Thunderbird pc#2, 70% training
ORIG	Original class for the e-mail

First of all take a look again at the original structure of filters, which was learnt by BNET:

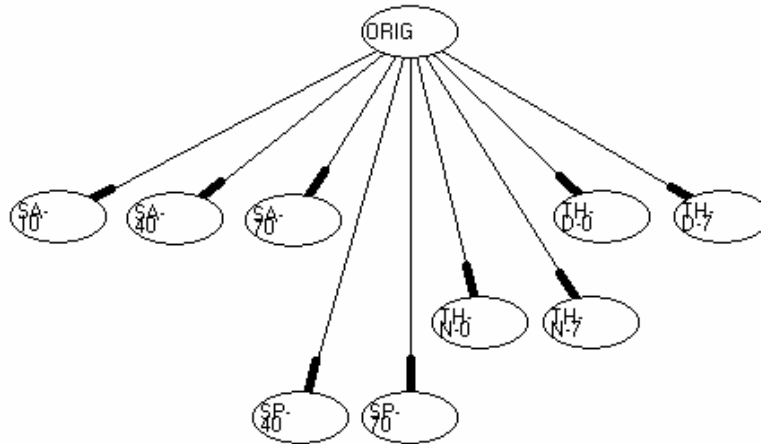


4.3.2 Figure: 1<sup>st</sup> model. The structure of the self-constructed Bayesian network – reduced corpus (same as in 3.3.1 Figure)

The AUC value of the model above is 0.9997. This value is extremely high. The explanation of it could be that almost all nodes are connected to the ORIG node and there are also additional connections between the nodes. This very high AUC value matches the estimations, since a very high accuracy can be made with the combined usage of all filters and a very fast learning can be achieved.

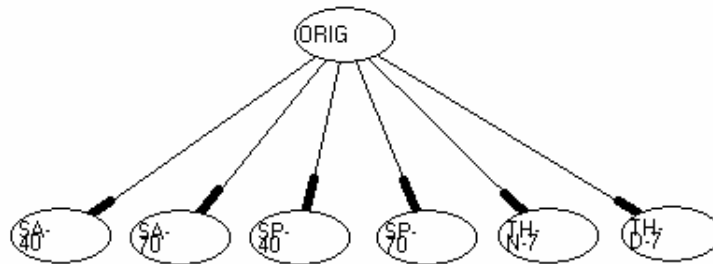
To build up meta models, a reduced scope is used, in which the meta models are only naive Bayesian models. A naive meta model presumes no connection between the filters. To confirm the propriety of meta models, take a look on a full naive Bayesian meta filter, where all the previous nodes are used. According to the naive Bayesian model, there will be only one parental node, the ORIG and all others will be children of it.





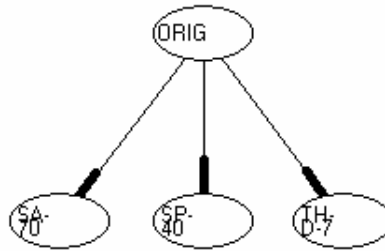
4.3.3 Figure: 2<sup>nd</sup> model. The naive Bayesian meta model with all filters

The AUC value for the full naive model is 0.9994. Compared to the AUC of the originally learnt structure (0.9997) only 0.0003 is the difference. It gives the base to the idea, that no connection is required between the filters. Although this AUC value is impressive, the goal is to find a meta model that uses not all of the nodes. The next model is a slightly reduced one, with only six nodes in it.



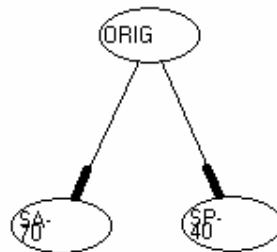
4.3.4 Figure 3<sup>rd</sup> model. The reduced naive model.

The reduced model produced a significantly smaller AUC with 0.9762. Additional reduction could be made, if only three nodes are used. The best selection is seemed to be SA70, SP40, THD7 since these three nodes are all connected to the ORIG and they are not in one clique (see Figure 4.3.2). Furthermore these nodes are connected with almost all (except two) nodes.



4.3.5 Figure: 4<sup>th</sup> model. Naive model with three nodes

The AUC for this model is 0.9722. This value is still acceptable and uses only a reduced number of nodes. Compared to the model above with six nodes (0.9762), the AUC differs with 0.004, while the number of nodes is only the half of it. If a concrete suggestion must be made on a meta spam filter, this should be one of the preferred meta models. If only two filters are selected, the two most suitable are SA70 and SP40.

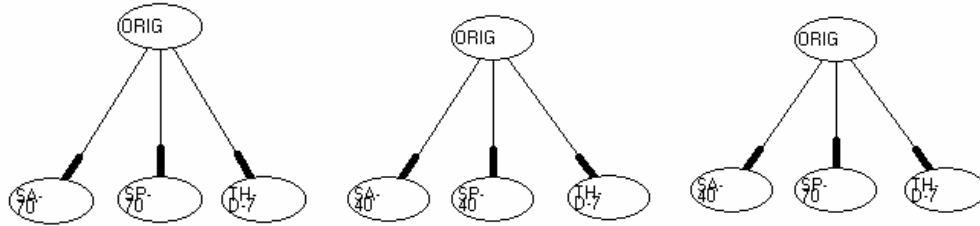


4.3.6 Figure: 5<sup>th</sup> model. Naive model with two nodes

The AUC for this model is only 0.9451. Although there is no such a threshold for AUC that could be called enough or satisfying, this solution produced under 0.95, what is quite far from the values above. As a resource-saving solution this model is important, but its accuracy lags behind the other, more complex models.

#### 4.4 Confirmation of the results

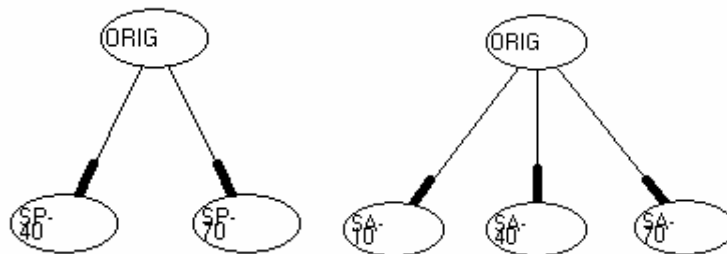
The above mentioned naive model with the three nodes was mentioned as best-seemed selection. To prove the relevancy of this model other three-node models were tested.



4.3.7 Figure: 6<sup>th</sup>, 7<sup>th</sup>, 8<sup>th</sup> models. Additional three-node naive models

The model on the left side (SA70, SP70, THD7) has the AUC of 0.9636. The one in the middle (with the SA40, SP40, THD7) nodes has the AUC of 0.9601, while the one on the right (with the SA40, SP70, THD7) nodes has the AUC of 0.9706. Although the last meta filter produced a result near to the above mentioned, principally best filter, all models are lag behind that (summary table of models and AUC values will be given later on).

It is not enough to make suggestion on a three-node meta spam filter model, it is also important to see, that this meta model is better than the filters alone. For this statement naive Bayesian models were built from the best filters, the SpamAssassin and SpamProbe, not only with the biggest training sets, but also with the smaller training sets, in order to achieve the highest AUC values, what the filters alone are capable for.



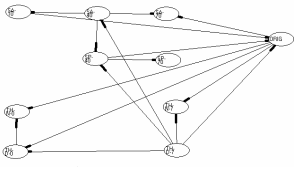
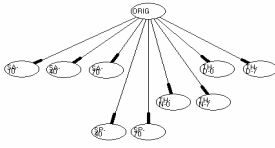
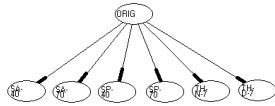
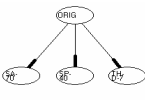
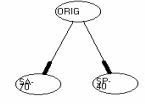
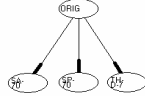
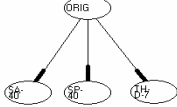
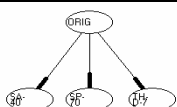
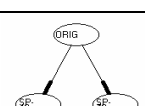
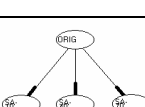
4.3.8 Figure: 9<sup>th</sup>, 10<sup>th</sup> models. Naive models of the single filters

The AUC of SpamProbe is 0.9120 and of SpamAssassin it is 0.8859. These values are obviously less than any of the values of the three-node models. This statement is important in order to proof the justification of the multi-filter meta model.

### 4.5 Summary

The final section gives a summary table about the analyzed models. In the table the most relevant (by complexity, accuracy, etc.) model is highlighted.

4.5.1 Table: All tested models (with rate compared to the 4<sup>th</sup> model)

No.	Included nodes	Structure	AUC	Rate
1 <sup>st</sup>	SA10, SA40, SA70, SP40, SP70, THN0, THN7, THD0, THD7		0.9997	102,83%
2 <sup>nd</sup>	SA10, SA40, SA70, SP40, SP70, THN0, THN7, THD0, THD7		0.9994	102,80%
3 <sup>rd</sup>	SA40, SA70, SP40, SP70, THN7, THD7		0.9762	100,42%
<b>4<sup>th</sup></b>	<b>SA70, SP40, THD7</b>		<b>0.9722</b>	<b>100,00%</b>
5 <sup>th</sup>	SA70, SP40		0.9451	97,21%
6 <sup>th</sup>	SA70, SP70, THD7		0.9636	99,11%
7 <sup>th</sup>	SA40, SP40, THD7		0.9601	98,76%
8 <sup>th</sup>	SA40, SP70, THD7		0.9706	99,84%
9 <sup>th</sup>	SP40, SP70		0.9120	93,81%
10 <sup>th</sup>	SA10, SA40, SA70		0.8859	91,13%

## Conclusions

As a result it turned out that SpamAssassin trained with 70% and SpamProbe trained with 40% plus Thunderbird trained with 70% gives a very good combination as a meta spam filter. The three nodes are a real reduction compared to the original nine nodes and furthermore not all of the used filter must be trained with 70%. This solution makes a very good compromise between complexity factor, learning capability and accuracy rate.

To summarize the results of the thesis it can be said, that the creation of a meta spam filter makes sense and has its grounds. Although the thesis deals with concrete spam filters and e-mail corpus, the above described methodology can also be applied for other filters as well. Analysis of Bayesian networks has provided a good basis for the creation of a meta spam filter.

# References

## Book

- [11] Ending Spam - Bayesian Content Filtering and the Art of Statistical Language Classification by Jonathan A. Zdziarski

## Data sources

- [21] SpamAssassin public e-mail corpus,  
<http://spamassassin.apache.org/publiccorpus/>
- [22] Spamcop statistics, <http://www.spamcop.net/spamstats.shtml>

## Official homepages

- [31] Mozilla Thunderbird, <http://www.mozilla.com/thunderbird/>
- [32] Project honey pot, <http://www.projecthoneypot.org/>
- [33] SpamAssassin, <http://spamassassin.apache.org/>
- [34] SpamProbe, <http://spamprobe.sourceforge.net/>

## Papers

- [41] A plan for spam by Paul Graham, <http://www.paulgraham.com/spam.html>
- [42] Anatomy of a Phishing E-mail by Christine E. Drake, Jonathan J. Oliver, and Eugene J. Koontz
- [43] Az embereket egyre kevésbé zavarja a spam,  
[http://www.sg.hu/cikkek/36448/az\\_embereket\\_egyre\\_kevesbe\\_zavarja\\_a\\_spam](http://www.sg.hu/cikkek/36448/az_embereket_egyre_kevesbe_zavarja_a_spam)
- [44] Sophos: messze még a spam halála by Gyurkity Péter,  
[http://www.sg.hu/cikkek/41288/sophos\\_messze\\_meg\\_a\\_spam\\_halala](http://www.sg.hu/cikkek/41288/sophos_messze_meg_a_spam_halala)
- [45] Spam Summit calls for global coalition to fight junk e-mail by Gregg Keizer,  
<http://www.techweb.com/wire/26801924>
- [46] SMTP Path Analysis – Exposing Zombie Spammers by Aaron E. Kornblum in CEAS 2005
- [47] What is spam, and how do I tell whether a message is spam or not?,  
<http://www.clearswift.com/support/technotes/item.aspx?ID=1561>

**Wikipedia**

- [51] Bayesian network, [http://en.wikipedia.org/wiki/Bayesian\\_network](http://en.wikipedia.org/wiki/Bayesian_network)
- [52] Bayesian inference, [http://en.wikipedia.org/wiki/Bayesian\\_inference](http://en.wikipedia.org/wiki/Bayesian_inference)
- [53] E-mail spam, [http://en.wikipedia.org/wiki/E-mail\\_spam](http://en.wikipedia.org/wiki/E-mail_spam)
- [54] Learning Bayesian network structure,  
[http://en.wikipedia.org/wiki/Learning\\_bayesian\\_network\\_structure](http://en.wikipedia.org/wiki/Learning_bayesian_network_structure)
- [55] Mozilla Thunderbird, [http://en.wikipedia.org/wiki/Mozilla\\_Thunderbird](http://en.wikipedia.org/wiki/Mozilla_Thunderbird)
- [56] ROC, [http://en.wikipedia.org/wiki/Receiver\\_operating\\_characteristic](http://en.wikipedia.org/wiki/Receiver_operating_characteristic)
- [57] Sensitivity, [http://en.wikipedia.org/wiki/Sensitivity\\_%28tests%29](http://en.wikipedia.org/wiki/Sensitivity_%28tests%29)
- [58] Specificity, <http://en.wikipedia.org/wiki/Specificity>
- [59] Spam (electronic), [http://en.wikipedia.org/wiki/Spam\\_%28electronic%29](http://en.wikipedia.org/wiki/Spam_%28electronic%29)
- [60] SpamAssassin, <http://en.wikipedia.org/wiki/SpamAssassin>