Na tomto místě bude oficiální zadání vaší práce

- Toto zadání je podepsané děkanem a vedoucím katedry,
- musíte si ho vyzvednout na studiijním oddělení Katedry počítačů na Karlově náměstí,
- v jedné odevzdané práci bude originál tohoto zadání (originál zůstává po obhajobě na katedře),
- ve druhé bude na stejném místě neověřená kopie tohoto dokumentu (tato se vám vrátí po obhajobě).

Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Computer Science and Engineering

Master's Thesis

Illegal newsletter detection unit

Bc. Tomáš Gogár

Supervisor: Ing. Jan Šedivý, CSc.

Study Programme: Open informatics, Master

Field of Study: Artificial Intelligence

May 11, 2014

Aknowledgements

I would like to thank my supervisor, Ing. Šedivý CSc., for the guidance, encouragement and great advices, which he has given to me throughout the work. I also want to thank my parents for their support throughout my studies and my whole life.

Declaration

Abstract

The task of filtering email spam (also known as *Unsolicited Bulk Mail*) has been an important topic in the machine learning community during the last two decades. Although the existing techniques are able to filter the majority of spam, more sophisticated spammers are constantly refining their strategy so that their messages pass the existing filters. One example of such behaviour is a new spamming trend observed by the largest freemail service provider in the Czech Republic. The unsolicited bulk mail which resembles a standard newsletter is delivered to the email addresses, which come from illegally obtained databases. We use a term *Illegal newsletters* for such behavior. The problem of such emails is that we cannot use the content filters to detect them and that the senders mask themselves by changing IP addresses and domains. Currently the contracting email service provider is manually gathering information about the senders of *Illegal newsletters*, so it can develop defence algorithms. Because manual analysis is highly inefficient, we propose a system which should help human operators in identifying new spammers by highlighting suspicious emails. We propose features that should describe normal behavior of the senders and we use anomaly detection algorithms to detect abnormal emails. The preliminary results from testing on small set of labeled emails suggests that the majority of anomaly emails represents unsolicited bulk mails and that such approach should help the operators in identifying a significant portion of delivered spam.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The first electronic mail systems appeared before the inception of the internet and date back to the sixties of the 20th century [34]. The email system as we use it now, was introduced in 1982 in RFC 821 [26]. In this document the smtp protocol was defined and it was later updated in 2001 and 2008 by RFC 2821 and 5321 [21] [22]. Although there appeared other technologies for electronic communication in the recent years, the popularity of email system is still growing. The Radicati group estimates that 182 billion emails were sent per an average day in 2013 [18].

1.1 Bulk Emails

Since the email system provides very cheap and easy way of communication, it is very often used for sending messages to the large audience. A message which is sent to large number of recipients with none or only small number of changes in its body is called a Bulk email.

1.2 Unsolicited Bulk Emails

When a bulk message was not requested by the recipient, we refer to it as Unsolicited Bulk Mail (UBE) or Email spam[1].

The first internet spam was most probably sent in 1978 by Gary Thuerk to 600 users of ARPANET [5]. Since that time the problem of spamming has grown significantly, and according to Kaspersky Lab it made about 70 percent of overall email traffic in the second quarter 2013 [20].

Nowadays, most of the spam emails contain URL to one or multiple websites, which offer commercial products to its visitors. Since senders of such emails try to reach as many mailboxes and keep the price for infrastructure as low as possible, they often utilize botnets

[1] It is named after Spam, which is a luncheon meat, popular in the UK during the World War II. This name was chosen for unsolicited bulk mails because of its role in one Monty Python sketch, in which Spam is annoyingly included in every dish (see youtu.be/anwy2MPT5RE). Although the term nowadays refers to any kind of unsolicited message (i.e. IM spam, newsgroup spam, fax spam, letter spam etc.) in this work we will use it for the email spam only.

instead of usual mail servers. On the other hand huge amount of unsolicited bulk emails cause significant costs on the recipients side (costs for the infrastructure and storage), as well as it reduces the usability of the email system itself. Therefore the efforts to automatically filter UBEs is one of the main topics within the email developers community.

1.3 Grey Emails

In addition to spam, other types of bulk emails are sent. The term grey mail is used for emails which we have subscribed for. The problem is that many users did it accidentally or unknowingly. In such cases, recipients consider the email as spam, although according to the law it is a legal behavior. These bulk emails should contain links for unsubscribing the delivery and are usually sent from sender's own mailservers. Typical examples of grey emails are newsletters, social sites notifications, websites feeds, watchdogs, etc.

It is difficult (even for a human operator) to determine, whether the grey email is delivered only to the subscribed users, since no one has an exact history of the user's behavior. So the grey emails are usually delivered and only if a large number of users report it as unsolicited, the sender can be retrospectively marked as a spammer.

1.4 Illegal newsletters

Illegal newsletters is our term for a new spamming trend which was observed by the contracting company which is the biggest freemail provider in the Czech Republic. Although we have data only from one provider, we expect such trend to be widespread and global. The provider observed that unwanted emails whose content resembles usual newsletter, are sent to email addresses from illegally obtained databases. Emails contain html links, which direct the victim to the promoted webpage. The sender uses such illegal newsletters to promote various products and services and receives a small amount of money for every visit of the page. Sometimes even the companies which are being promoted in such campaigns do not know that it is being done in an illegal way, even though they are the ones who pay for the marketing. These companies very often outsource their marketing campaigns and therefore the final sender can be well hidden behind a long supply chain of marketing services. One of the main problems in identifying this kind of messages is its similarity with other legal newsletters, so it is difficult to filter them out with traditional antispam techniques.

Chapter 2

Our goal

During the last year the Czech email service provider (ESP) identified two biggest senders of Illegal newsletters and developed a system which is able to block these spammers. Unfortunately, in the training phase this approach requires a set of labeled emails from the particular spammers. Such identification is not trivial and requires work of a human operator, who manually analyzes received emails. The goal of this work is to propose a system which will help the human operator in distinguishing *Illegal Newsletters* from other *Grey Emails*. Our effort to find an semi-automated solution for this task is important because we expect there is more similar senders which ESP have not identified yet and manual search is highly inefficient.

Why is classification of illegal newsletters a challenging task is summarized below:

- Emails are not sent from company mail servers as usual newsletters, therefore it is not possible to create consistent reputation statistics for promoted companies.
- Senders are trying to hide themselves by means of changing their fingerprints (IP addresses, domains, etc.).
- The body of emails resembles legal newsletter, so it is difficult to filter them out with content filters.

Compared to usual spam filtering task, we have very small amount of labeled data and huge amount of unlabeled data. Although the sender's behavior changes in time and the content filters do not work, we hope we can specify features that capture their masking behavior and we will be able to detect suspicious messages in the email traffic.

Chapter 3

Email characteristics

Before we dive into spam filtering techniques, we will summarize, what sort of information is available when an email arrives to the server, which is supposed to decide whether it is ham or spam. We divide data from an email into three components based on its source - SMTP envelope, Email headers, Email content.

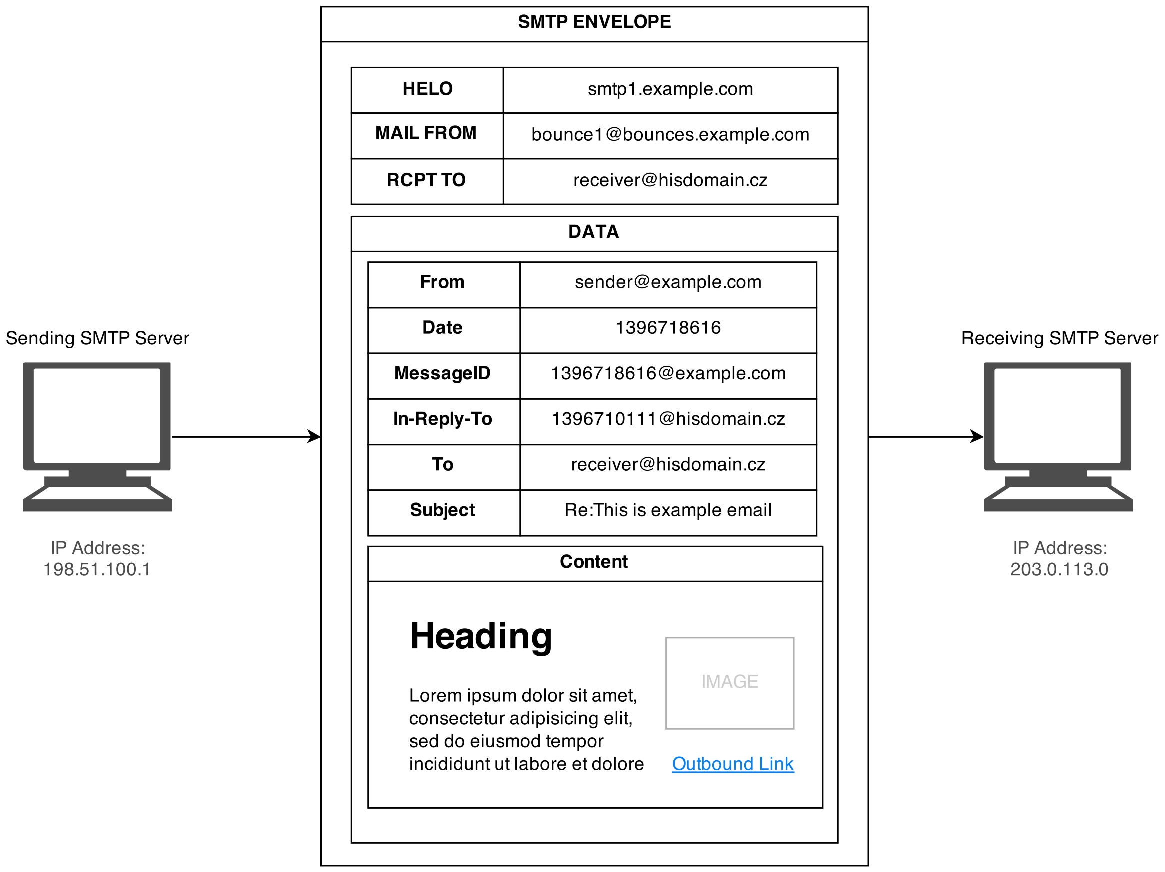


Figure 3.1: Data within an email - example

3.1 SMTP envelope

The process when two mail servers communicate in order to exchange an email is called SMTP session. At the beginning of the session the sending server (often referred to as client in this context) provides the following information:

- **HELO (or EHLO) string** String that identifies the sending server - usually with its fully qualified domain name (FQDN). EHLO string is used within new ESMTP protocol.
- **MAIL FROM** This field is used to present the originator's email address. It is also the address where notifications (bounces) of undelivered messages should be sent.
- **RCPT TO** Recipient's email address. It can be used multiple times, in case of multiple recipients.

Since these data fields serves to identify the recipient and the sender, it is often referred to as an Email (or SMTP) Envelope. After successful negotiation the sending server starts to send email data. Email data contain email headers and email content.

3.2 Email headers

The headers include all other structured information and are part of Data stream within the SMTP session. The mandatory fields include:

- **From** The field should include email address (and optionally name) of an author.
- **Date** Date and time when an email was composed.

Fields that should be included [27]:

- **MessageID** ID of an email.
- **In-Reply-To** Only reply messages include this field. It should be filled with a MessageID, that the email is replying to.

And we only mention other common optional fields: To, Subject, Bcc, Cc, Content-Type, Precedence, References, Reply-To, Sender, Archived-At

3.3 Email content

Although email system was originally designed only for plain text messages, nowadays either plain text or html is used to represent email content. Since html provides more options for graphical expression and interactivity - it is usually the sender's choice for marketing emails. Usual features found in html emails are: Formatted text, Images, Outgoing links. Emails can also include multimedia attachments of limited size.

Chapter 4

Related work

Since spam is a very concerning issue, a lot of work on spam filtering has been done in the last two decades. This chapter briefly describes history and current trends in the research area. More detailed summary with method evaluation can be found in [2] and in [15].

4.1 First filtering methods

One of the most straightforward technique of spam filtering is usage of blacklists of spam senders. Blacklisted sender is assumed to send spam and therefore all his messages are filtered out. Unfortunately the only relevant information to identify the sender is his IP address. Although we have already listed other header fields such as MAIL FROM or HELO STRING, none of these fields is protected against spoofing, and so the sender can easily forge them. Identifying senders by their IP address can be partially effective, however it also brings a whole range of problems. Spammers can change their IP addresses quickly, and so they can avoid the filters at the beginning of the attack. It can also result in filtering out good mail, when a good sender obtains already blacklisted IP address.

In order to minimize these risks, email service providers do not use only their private blacklists, but they often shared them with each other. A DNS-based Blackhole List (DNSBL) is a method of sharing blacklisted IPs through DNS servers [23]. This approach brings benefits of DNS system to Blacklists sharing such as scalability and cacheability. Since 1997 when the first DNSBL list was introduced, there have been developed many other blacklists available through DNSBL.

Besides blacklists, there exist other related methods. Whitelists are the exact opposite and include trusted senders. Or greylisting, which is method of temporary rejecting messages from unknown sender and accepting messages only if the sending server tries to send the message again after some delay. This method works because many tools for sending bulk mail do not try to resend undelivered mail.

4.2 Content-based methods

The very first methods that started to use the email content were based on a hand-built rules set, that was applied to filter junk mail. Since spam was relatively static at that time, words like sex, free, viagra etc. were good indicators of spam messages. When spammers started to adapt the filters, a need for an automatic rule creation appeared. One of the first works that applied machine learning techniques for spam filtering were [28, 24], which used Naive Bayes based on the bag-of-words model to classify messages. Machine learning methods introduced the need of training data (messages labeled as spam or non-spam), and this requirement grows ever since. With spammers quickly adapting to learning algorithms, more and more training examples were required and the filters needed to be trained more often. Comparison of Naive Bayes approaches can be found in [32].

Advances in machine learning helped to improve spam filtering as well. Methods using Support Vector Machines (SVM) or TF-IDF, significantly improved accuracy and speed of learning [12]. Also boosting, which in general combines results of multiple weak classifiers, was introduced for email classification [6].

The reaction of spammers to machine learning filters usually focuses on two vulnerabilities:

Masking critical words Since content-based methods became considerably effective, spammers try to obfuscate the content, so it is difficult for machines to get the features, while the message is still readable for humans [15].

One way of doing it, is through HTML commenting (`fr<!->whatever<!->ee`) or by using HTML ASCII codes (`fr&#101xe`). Both examples will be rendered as "free" in an email client, but for email servers it is too computationally expensive to render huge number of emails separately.

Another way is to hide the text within an image. Optical character recognition techniques (OCR) are too slow for the task of spam filtering. When image matching algorithms were introduced, in order to detect bulk messages containing the same image [36, 33, 11], spammers responded by randomizing images and randomly splitting them. Again, the whole message gets rendered correctly in email clients, but it is difficult for filtering algorithms to imitate such behavior [15].

Confusing matching systems Matching systems are supposed to detect messages with the same or very similar content. When the similar messages come from different sources (different domains, IPs, etc.), it is the only way how to detect bulk messages. Such matching is crucial for blacklisting, feature extraction, but also for reduction of computational requirements (by making intensive computations only once for the whole cluster of similar messages).

The problem of finding nearly duplicate documents has already been studied in other data-mining fields and many approaches have already been proposed and used [9, 13, 8]. On the other hand spammers constantly try to avoid matching by randomizing their messages. Randomizing is usually done by adding random text to an email, and therefore the more and more robust detection algorithms are being developed by the text-mining community [19].

4.3 Behavioral methods

Other methods, which are often used together with content-based filters, focus on behavior of senders as well as reactions of receivers. Some works attempt to create social networks from email communication graphs [4, 3]. In [3] communication graph of email users is created and groups of good and bad senders are estimated based on the properties of connected components of the graph.

Often ESPs use reactions of receivers to either improve filter's learning phase (by letting users to mark false positives and false negatives) or to create reputation database of senders. Reputation databases capture "spamicity" of individual senders over long period of time and the information is then used as a part of filtering algorithm. In [31] results of automatically created sender reputation database are presented. It shows that, if an ESP has large amount of feedback data from users, reputation database can help to filter messages effectively. In [14] combination of social network and hand-crafted sender reputation is presented.

We have already mentioned the problem of verification of email headers. However, reliable identification of a sender is crucial for building social graphs and reputation databases. Fortunately, new standards which extend original SMTP and help in solving these issues have been developed and are successively adopted by the email community.

4.4 Email standard extensions

The possibility to forge header fields, facilitates spammers hiding their true identity and it also makes phishing attacks easier. Phishing is an attempt to acquire sensitive personal information (such as credit card information, passwords, etc.) by pretending to be some credible authority (user's bank, mobile network operator, etc.). In order to limit possibilities for such attacks - SPF and DKIM systems were designed and successfully deployed. All these systems are compatible with the existing email infrastructure, and therefore we refer to them as extensions.

Sender Policy Framework (SPF) SPF was defined in [35] and allows owner of a domain to specify IP addresses, which are allowed to send emails from the domain. Lists of the allowed IPs are then published to DNS servers using special DNS SPF records. When a receiving server gets an SMTP connection, it checks its MAIL FROM domain against SPF records and when the IP address of a sending SMTP server is in the lists, it assumes the email to come truly from the domain.

DomainKeys Identified Mail (DKIM) Since SPF checks only the message envelope (MAIL FROM), DKIM was designed to validate message data (headers and content) [10]. It is based on digital signature of the message content (usually some parts - such as From, Subject and message body). Signature and other necessary information (i.e. covered fields, signing algorithm, etc.) are part of a new *DKIM-Signature* field. The receiver which wants to validate a message, gets sender's public key through DNS and verifies the truthfulness of the signature and the actual message content.

Due to DNS Survey from 2010, 15.9% of domains implemented SPF (only .com, .org and .net were tracked) [30]. We hope that adoption of anti-forge techniques will continue and it will help us in building reliable reputation databases.

4.5 SpamAssassin

Email service providers either create their own implementations of all the mention techniques or they can use and customize 3rd party solutions. One of the most popular solution, which is currently used by the Czech ESP, is SpamAssassin.

SpamAssassin is a computer program released under Apache License 2.0, that combines multiple spam-detection techniques such as BlackLists, WhiteLists, DNSBL, Regular expression filtering, Bayesian filtering, Content Matching, SPF, DKIM, etc. Examined email passes multiple tests within SpamAssassin and each tests gives it a spamscore. The higher the spamscore, the higher the probability that the email is spam. These individual scores are combined to a global spamscore, which is then compared to a threshold, in order to decide, whether the message will be labeled as spam. Usually multiple tests need to be "positive" to reach the threshold (this condition is used, in order to decrease the probability of false positives). SpamAssassin also provides plenty of parameters, that can be adjusted, so it can fit the needs of a user.

Chapter 5

Data

In this chapter we describe data available for our task. The contracting ESP is the biggest freemail provider in the Czech Republic and it receives about 120 thousand of incoming SMTP connections per minute. Approximately half of the connections are accepted and every accepted message undergoes a receiving process, which consists of many steps (STMP phase, greylisting, SpamAssassin check, etc.), before it is delivered to a user. Approximately 40 million messages are persisted per day and therefore the whole process needs to be carried out in the large cluster of computers (consisting of 1000 servers), where the individual processing subsystems are deployed independently.

Since the current email storage space is about 1.4 petabytes and I/O operations are not fast enough, it is difficult to access the emails when they are already persisted on the hard drives. Hence the ESP implemented a logging system (sometimes referred to as probe system), which logs some important data, while the email is in the receiving phase (and still in the memory of the servers).

These logs are stored on the machines where the processing subsystems are deployed. This implies that different logs for a single message are distributed over multiple servers and need to be merged once in a while. At the end, these merges result in structured logs for every single message and can be used for further analyses. These logs are stored in Protocol Buffers format and take up to 100 GB per day (more about Protocol Buffers in [17]).

Logs contain basic email characteristics (such as envelope and content headers), timestamps, matching system results, some IP statistics, SpamAssassin's results, user behavior, etc. Selected characteristics which are crucial for our tasks are summed up in table 5.1, here we provide only simplified form of the logs, the real structure of the logging format is more complex.

If we want to access email content itself, we need to specify it by its unique identifier and retrieve it from hard drives. Since we focus on the bulk messages only, we usually use data from the matching system to gather the group of similar messages and then we retrieve only one email as an example of the whole group. We consider an email as a bulk mail, if the matching system marks emails with the similar content, which are delivered to more than 500 other users. Throughout this work we refer to the set of similar bulk messages as a *bulk group*.

For the purpose of this work, we had logs for 5 days - 2 days from the beginning of the February 2014 and 3 days from the first half of the April 2014. Since our work is focused on detecting new illegal senders, we use only data for emails, which passed all the existing filters of the ESP and ended up in inboxes of the users. The statistics of delivered messages are summarized in table 5.2. You can see that there is considerably less bulk mails for the first two days. This difference is caused by the way how the bulk mails were detected in older version of the logging system.

Source: SMTP Servers	
IP	address of the sending machine
HELO string	envelope helo string
GreyListDone	whether it has passed greylists
SessionDuration	duration of the smtp session
Rcpts	envelope recipients of the message
MailFrom	bouncing address
Spf Status	whether it has passed the SPF check
Source: SCANNER (Antispam component)	
Spamscores	spamchecking modules and the received score
PepcaHash	1st hash from the matching system
SimcaHash	2nd hash from the matching system
Source: EBOX (The main email component)	
Thrash	user moved the email to thrash
Unthrash	user removed the email from thrash
MarkSpam	user marked the email as spam
UnmarkSpam	user unmarked the email
Set folder	email was moved to folder
Label set	user labeled the email
Label remove	user unlabeled the email

Table 5.1: Selected characteristics available in log files

	Date	**Total delivered messages**	**Delivered bulk messages**	**Bulk groups**
Day 1	02/08/2014	Not available	17,922,898	1,832
Day 2	02/09/2014	Not available	17,403,976	1,906
Day 3	04/08/2014	44,700,490	25,434,898	2,241
Day 4	04/14/2014	41,334,968	21,218,346	2,479
Day 5	04/15/2014	43,434,874	23,385,524	2,788

Table 5.2: Data summary

Chapter 6

Design of proposed solution

As we have already stated in Chapter 2 our task is to create a system, which will help human operator to distinguish illegal newsletters from grey emails. We also have mentioned why this task is not trivial - mainly because senders change often their identity and emails resemble usual newsletters, so we cannot the use content filters.

The ESP is building two other systems, which will work together with SpamAssassin and which should help in finding bad senders. These systems are:

- Private social graph - which should provide picture which senders behave suspiciously
- Reputation database - where reactions of receivers will be recorded and used for filtering

Both of these systems work with long-term statistics, which require enormous amount of data and therefore it needs to be implemented directly in the infrastructure by the ESP.

On the other hand our system should work complementary to the reputation database and should detect illegal newsletter even without the history data. Hence it becomes useful when a bulk mail arrives from the sender, who is not in the reputation database yet (i.e. he has never sent a bulk mail to ESP). The situation is suspicious but not illegitimate and there are actually two possible reasons for such behavior:

- It is a new legitimate sender (This does not happen very often).
- It is a sender, who is trying to mask himself and hide behind another identity.

So we want to propose features and an algorithm, which will help to distinguish these two possibilities. The role of our system within the whole process is depicted in figure 6.1.

6.1 Hypothesis: Expected behavior of a sender

We don't want to create any bias by exactly describing the behavior of the senders, but since we need to come up with distinctive features, we needed to make few assumptions, which are summarized in this section.

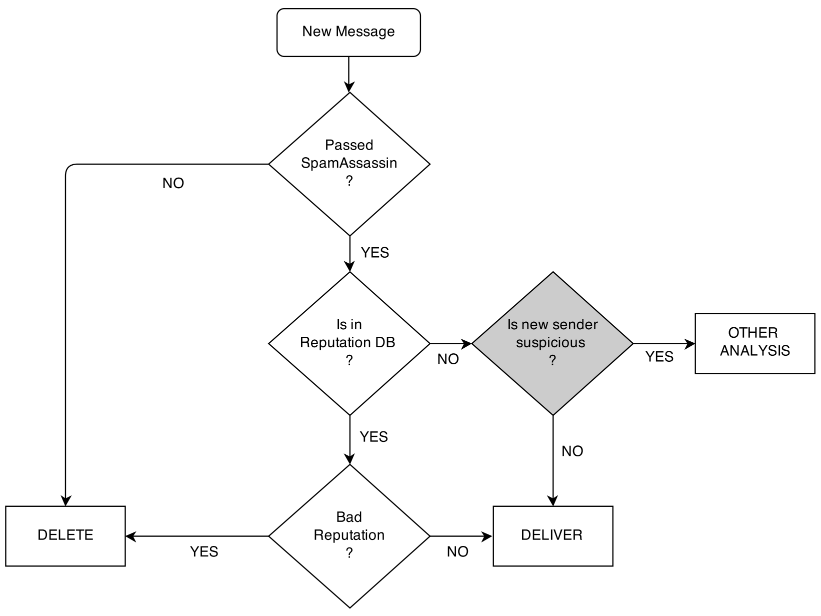


Figure 6.1: Our unit as part of the new filtering system

Illegal senders do change their behavior and so we don't want to make any assumptions about them. On the other, we think, we can quite accurately estimate the behavior of the legal ones. Such behavior will be pretty consistent and so a machine can learn it and detect anomalies.

All bulk message senders want to communicate some kind of information and so we can expect that they will (in their own interest) make use of more internet communication channels and not just an email system. Also it is logical to be consistent in their self-presentation, so their subscribers always know, where to find contacts and additional information. From these hypotheses we would expect that:

- Similar emails will have the same (or a least very similar) address in From field.
- Domain in from address provides a functional web page with relevant information.
- Majority of outgoing links point to the content-related pages.

These assumption may seem to be pretty strict, but they are based on very basic marketing rules. Even though a lot of companies nowadays use email marketing services (such as MailChimp), we can expect that from address will belong to the promoted company and not to the service provider. On the other hand Mail From addresses will belong to the sending service, but it still should be consistent.

6.2 Features

We would like to select features that describe consistency of sender's self-presentation. In order to do this we need to compute the features across the whole set of similar messages (and not only for single message separately). As we have already written we use a term *bulk group* for a group of similar bulk messages. This approach requires reliable matching system (and we presume that ESP's matching system is reliable), on the other hand it results in significant data reduction - instead of classifying millions of emails directly, we classify thousands of *Bulk groups*.

In the following paragraphs we will describe the selected features.We divide them into two groups:

6.2.1 Bulk group consistency features

The first group contains features that describe the sender's behavior consistency. The features are:

- **Number of distinct MAIL FROM domains** Mail From field serves as an address for bounce messages. Senders often want to make statistics of undelivered emails and so they don't use only one address, but rather multiple structured addresses that can look like: `bounce00123@bounces.server1.example.com`. Even though address may differ, the second level domain (here: `example.com`) can be expected to be the same (in order to describe the sender). This feature computes number of distinct second level domains, which are extracted from the domain part of the address (the part following `@`).

- **Number of distinct FROM addresses** From is the address that is displayed to the receiver and we have already discussed, why it should be consistent over the whole set of message. Therefore this feature uses the whole address and not only the second level domain.

- **Number of emails sent per IP** Since spammers need to change their IP addresses, they are often forced to use them as much as possible, which can result in abnormal number of emails sent per one IP address.

6.2.2 Domain Features

The second group focuses on the properties of domains used within an email. There are basically two sources of domains in each message - Mail From address and From address. Our goal is to describe whether the domains from these sources are somehow suspicious. What does it mean suspicious in our context is described below.

6.2.2.1 Definition: Suspicious domain

We suppose that well-known and trustful internet entities possess comprehensive websites. Such websites are usually well structured, often very vast and known by the customers. In order to estimate whether the domain is suspicious we focused on two properties - size of the website (i.e. number of unique pages within the site) and it's Page Rank.

Size of the website when we want to examine the size of the domain, we add `www.` in front of the second level domain and we visit the resulted address with our crawler, which computes the number of unique pages without leaving the domain. Number of pages is limited to 10, so we do not get stuck crawling huge sites for a very long time.

Page Rank The size of the website does not provide enough information, since there is a lot of ajax-based sites, which appear as a single page to our crawler (and javascript crawling is out of scope of this work). Therefore we add another property which should describe the relevance and credibility of the site. For this purpose we used Google Page Rank, which we obtained through their public API.

Having these two properties, we have defined the suspicious domain as follows.

$$Domain\ is\ suspicious\ if:\ [Size = 0\ \vee\ Size = 1] \wedge [PageRank = 0] \tag{6.1}$$

When the suspicious domain was defined we used the results for every message to compute the statistics for the whole *Bulk group*. The final features are:

- **Percentage of suspicious Mail From domains**
- **Percentage of suspicious From domains**

As you can see, the first group of features makes sense only for the *Bulk groups*. Although, the domain features can be computed for every single message, we also converted the results to a percentage of the whole bulk group. This approach results in significant data reduction and helps us in the classification phase.

6.3 Data analysis

In this section we will analyze individual features and discuss the possibilities for unsolicited bulk mail classification. For the purposes of analysis, we have grouped more than 16 million emails from the first two days (the beginning of February, see Chapter 5) into more than 3 thousand *Bulk groups*, we have computed the features and plotted results in histograms (see Figure 6.2).

As you can see all the histograms contains a bin, which represents absolute majority of email groups. It means that majority of senders behave similarly and as expected - most of them use one From Address, one MailFrom domain, they send similar number of messages per IP address and often use working (unsuspected) domains in the address fields. It is worth to notice that we can see significant number of emails, which do not use working domains at all. We have manually analyzed such emails and discovered that there are basically two reasons for that - either it is a spamming domain or it is domain truly owned by some trusted sender, but it does not host any web page (often domains look like: *newsletter-example.com*, *info-example.com*, etc.). Some more accurate statistics for each feature are summarized in table 6.1.

The *Bulk groups* which falls into the minority bins presents suspicious behavior, which may be the consequence of spammers masking strategy. In figure 6.3 we have plotted the bulk

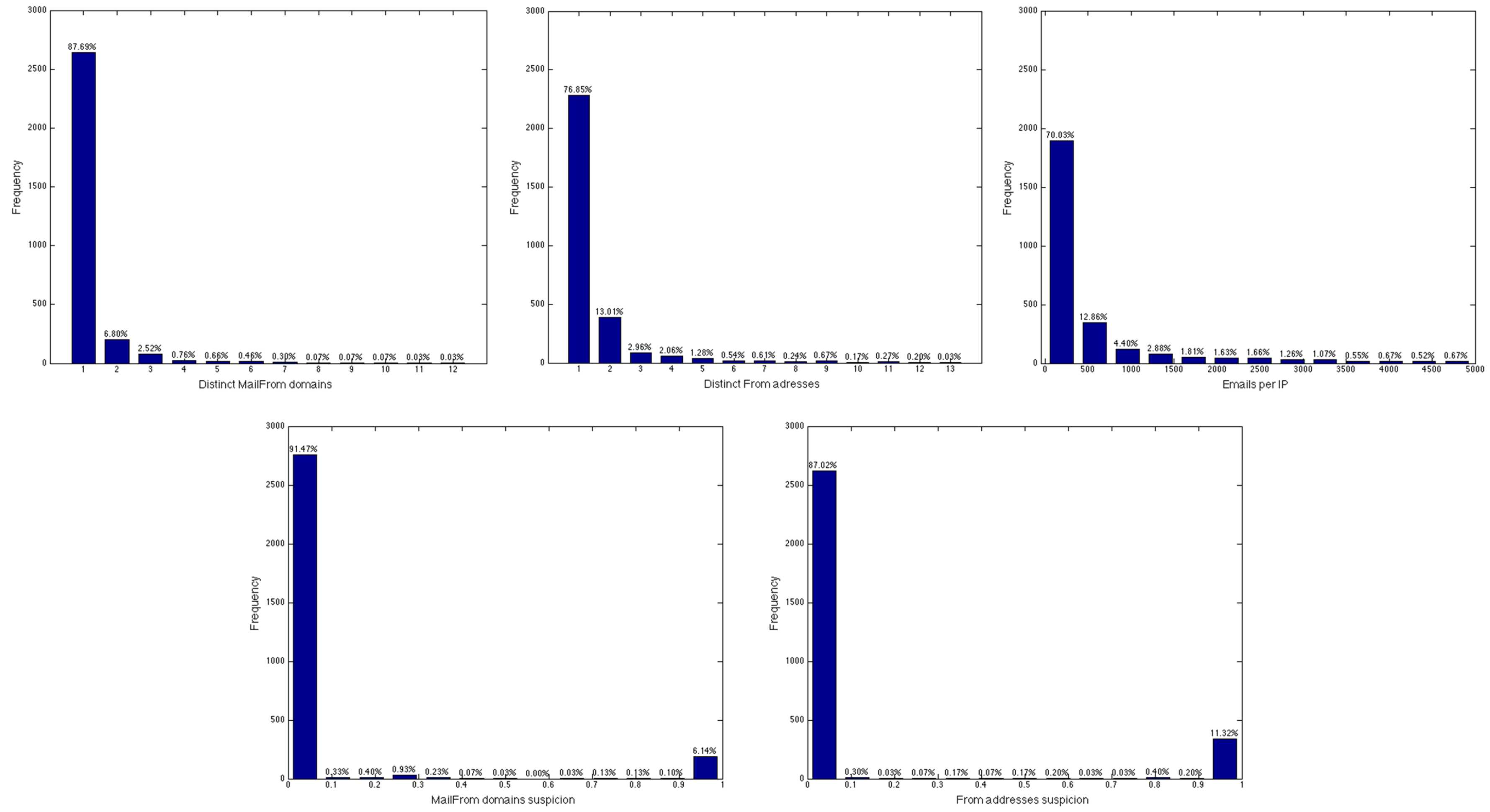


Figure 6.2: Histograms for individual features

Feature	**Statistics**
MailFrom domains	88% of bulk groups use 1 MailFrom domain. 95% of bulk groups use less than 3 MailFrom domains.
From addresses	76% of bulk groups use 1 From address. 89% of bulk groups use less than 3 From addresses.
Mails per IP	95% of senders sends less than 14,000 emails per IP.
From address suspicion	83% of senders does not use suspicious domains at all. 5% of sender use only suspicious domains.
MailFrom suspicion	86% of senders does not use suspicious domains at all. 11% of sender use only suspicious domains.

Table 6.1: Selected statistics of individual features

groups within the 5-dimensional feature space (we have used two plots to capture all the dimensions). The very small subspaces marked with green color represents the part of the feature space where 85 percent of bulk groups reside. This observation corresponds to the shape of histograms in Figure 6.2, where individual features were taken into account separately. This supports our hypotheses from section 6.1, that usual bulk mails should cluster near to each other. On the other hand, this result still does not suggests anything about spamminess of the "unusual" emails. We have manually analyzed few bulk groups that reside outside the expected cluster. An example of such group is marked in the charts with red circles - emails with the same content were send from 18 From addresses using 53 MailFrom domains, while 98% of domains in From address and 84% of domains in MailFrom domains were suspicious (i.e. do not host a reliable website). Such values suggest very abnormal behavior of the sender. When we manually analyzed the content of this particular email, we discovered that it promotes 'Jackpot in an online casino' and that it is very likely a spam (the content of the email can be found in Appendix in Figure C.1). Very similar behavior was observed with other unusual emails from the features space.

Such manual testing suggest that nonstandard *Bulk group* behavior can be related to spamminess of emails, but it does not guarantee anything. We don't know in which parts of the feature space the "ham" and "spam" emails reside and whether these parts are separable. In order to divide the feature space, linear classifiers are usually used (for example SVM). These algorithms need sufficient number of labeled data for the training, but unfortunately such information is not available for our task. The reasons for the lack of labeled data are:

- **We are analyzing "ham"** We have already mentioned in Chapter 5, that we analyze emails which passed all the filters and are delivered to inboxes - it means the current filters marked them as ham. But we know that some unsolicited bulk mail (such as illegal newsletters) is still in the set - it means that filters are not perfect.

- **Difficult manual classification** We also discussed in Chapter 1 why it is difficult to classify grey emails even for a human operator. There are emails which can be manually classified with sufficient degree of certainty, but the amount of such emails is not very high.

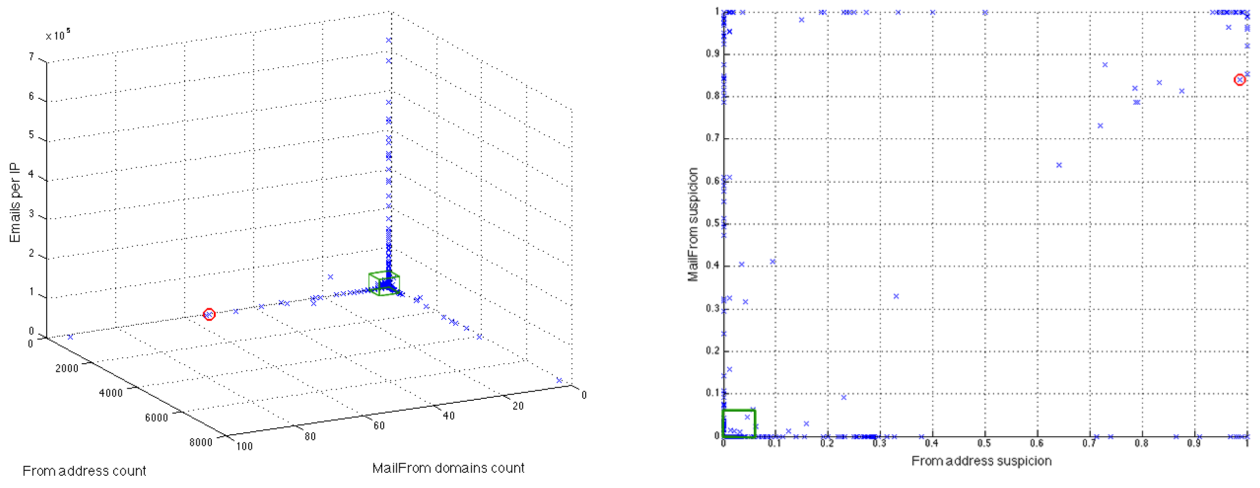


Figure 6.3: Bulk groups within the feature space

Therefore, the only information we can work with is that some senders behave unusually. We try to identify such senders and measure on the small subset of labeled emails whether such identification helps in spam recognition. Identifying unusual behavior is basically a task of anomaly detection and in the next section we will discuss the algorithms used in this work.

6.4 Anomaly detection

In general, anomaly detection refers to the problem of finding patterns in data that do not conform the expected behavior [7] . Such subtask is often part of fraud detection, intrusion detection for cyber-security or fault detection in safety critical systems. Therefore a lot of detection techniques have already been developed and an extensive analysis of such algorithms and their usage can be found in [7]. The algorithms can be divided by:

- **Nature of input data:** binary, categorical or continuous
- **Availability of data labels:** supervised, semi-supervised or unsupervised methods
- **Type of output:** anomaly score or labels

For our task we need an unsupervised algorithm which processes continuous input vector and outputs anomaly score, which is then compared to a threshold, in order to decide whether it is an anomaly or not. One group of techniques that fits our needs well, are statistical anomaly detection techniques. These approaches assume that normal data instances occur in high probability regions while anomalies occur in the low probability regions. During the training phase a statistical model is fitted to the given dataset and in the testing phase it is used to test the properties of unseen instances [7]. The techniques can be used even when the training dataset contains some anomalies, but it is necessary that large majority of training samples are normal instances. In our case, we assume that the majority of incoming mail is legal and therefore we should be able to train the model on the unlabeled dataset. The particular statistical techniques which we have used are divided into two groups:

6.4.1 Parametric techniques

The parametric techniques assume that the normal data were generated by some parametric distribution with parameters Θ and probability density function(PDF) $f(\mathbf{x}, \Theta)$, where $\mathbf{x}$ is an observation vector. The problem of the parametric models is that we need to choose a model even if we do not know the exact underlying distribution of the dataset. When we choose the model, we fit it to our dataset by estimating its parameters Θ. The resulting *Normal behavior score* of a test instance $\hat{\mathbf{x}}$ is the probability density function $f(\hat{\mathbf{x}}, \Theta)$.

In this work we have chosen normal distribution as our model, even though we are aware that the true underlying distribution is probably not gaussian[1]. The probability density function of one-dimensional normal distribution can be found in Equation 6.2. The parameters: mean μ and standard deviation σ are estimated using Maximum Likelihood Estimate. When the input is multi-dimensional vector (as it is in our case), it can be solved by either assuming that individual features are independent and we can train five gaussian distributions independently and get the resulting *Normal behavior score* as a product of individual probability density functions in $\hat{\mathbf{x}}$. If the such assumption cannot be made, we can use multivariate normal distribution whose density function is in 6.3 - where μ is mean vector, k is dimension of the vector and $\mathbf{\Sigma}$ is covariance matrix which captures the correlation of individual features. The anomaly score is then computed by using the single function.

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}} \tag{6.2}$$

$$f(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^k |\mathbf{\Sigma}|}} exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^T \mathbf{\Sigma}^{-1}(\mathbf{x}-\mu)\right) \tag{6.3}$$

When a final label needs to be assigned to a test instance, we check whether the *Normal behavior score* is higher than certain threshold. If it is higher the instance is labeled as normal, otherwise it is labeled as an anomaly. Example of two-dimensional probability density function and a threshold plane is in Figure 6.4. All instances whose PDF is on the peak above the plane would be considered as normal, those below the plane are marked as anomalies.

In this work we have implemented both (the univariate and multivariate) versions of Gaussian model and compared their performance in Chapter 8.

6.4.2 Non-parametric techniques

This group contains statistical non-parametric techniques that does not assume anything about the underlying distribution in advance and the model is created directly from the

[1]Normal distribution is often chosen when the true distribution is not known. More about the gaussian model fitting in anomaly detection problems can be found in the Coursera lecture by Andrew Ng: http://class.coursera.org/ml-003/lecture/95

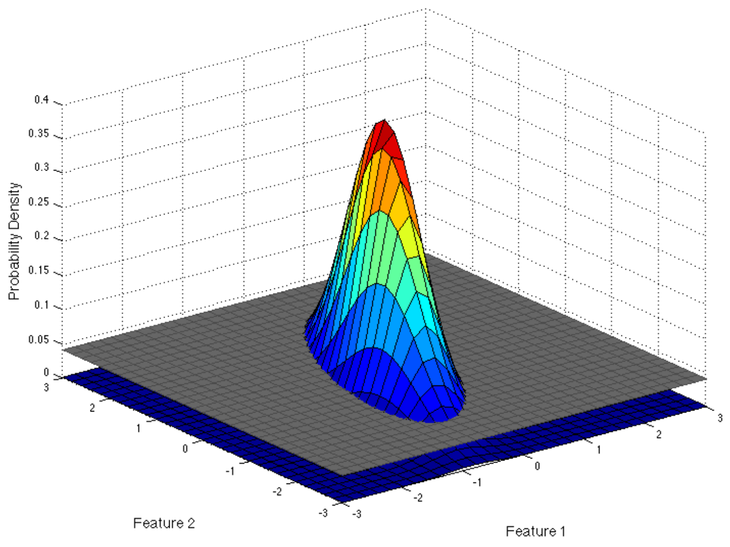


Figure 6.4: Example of 2-dimensional Probability density function with a threshold plane

training data. The disadvantage of these methods is that they cannot model multivariate data, so they cannot capture the correlation among the features and we need to estimate the final probability density function as a product of individual PDFs. It means we need to assume the independence of the features, as we did for one-dimensional gaussian distribution.

The most simple nonparametric statistical technique is to use directly the histograms of normal behavior (as they were depicted in the Section 6.3). When the histograms are used, they need to be normalized, so that they integrate to one. A disadvantage of histogram-based methods is a need of setting the bin width. If bins are too thin, the histogram is too sparse and there is higher probability that an instance falls to a zero-height bin, which can result in false positive error. On the other hand if the bins are too wide the probability of false negative error increases. We will test the influence of the bin width on performance of the classifier in the Chapter 8.

Next nonparametric model we will discuss is Parzen Window model. It is a kernel function-based method for estimating continuous probability density function [25]. The PDF is estimated from a set of n one-dimensional samples as follows:

$$f(x) = \frac{1}{n}\sum_{i=1}^{n} K(x - x_i) \tag{6.4}$$

Where K is called a kernel function - it must be a symmetric function that integrates to one. In this work we use a Gaussian distribution with mean at 0 and the standard deviation of the gaussian is set to properly smooth the feature histogram. In 6.5 you can see the final equation for estimating PDF and in figure 6.5 you can see an example plot of kernel functions (blue curves) for points $[1.1, 2.3, 4.7, 4.9, 8.1, 9.2]$ and the final estimation of the PDF (red curve), which is created by summing up the individual functions. As well as in previous case

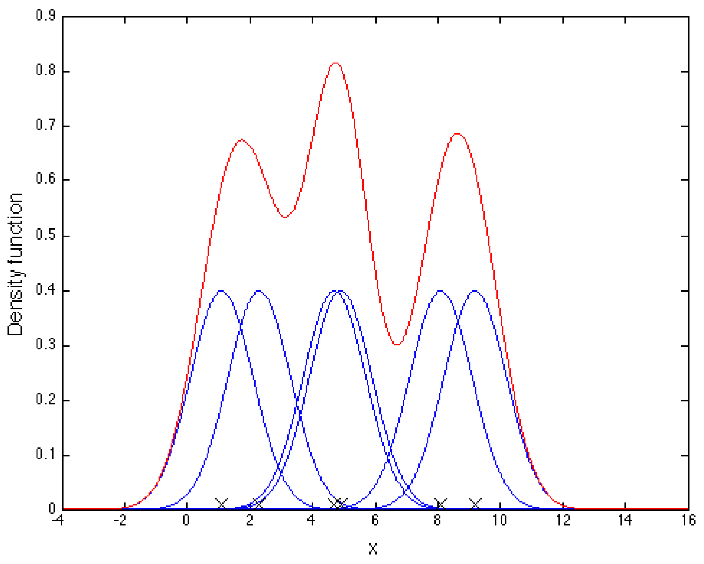


Figure 6.5: Example of estimating Density function using Parzen windows

the final probability density function needs to be created by multiplying PDFs for individual features. Another disadvantage of Parzen-windows is it's higher computational complexity in the testing phase, which is $O(n.m.f)$ while it is $O(n.f)$ for other models (n is a number of test instances, m number of training instances and f number of features).

$$f(x) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-x_i)^2}{2\sigma^2}} \tag{6.5}$$

In Table 6.2 the characteristics of the selected models are summarized. The results of the experiments and the performance of particular models can be found in chapter 8. But before diving into that, we will discuss the implementation details of the whole system.

Model	**Idependence assumption**	**Parameters**
Univariate Gaussian	YES	μ, σ (using MLE)
Multivariate Gaussian	NO	μ, Σ (using MLE)
Histograms	YES	Bin width
Parzen windows	YES	Gaussian σ

Table 6.2: Anomaly detection methods

Chapter 7

Implementation

This chapter describes implementation details of the toolkit developed throughout this work. The whole process of classification can be divided into 4 main steps :

1. **Process the log files** This tasks requires handling huge amounts of logs. The bulk messages need to be identified based on the results from the matching system and important statistics are gathered for every *Bulk group*.

2. **Get domains suspicion** When all unique domains are extracted, we need to gather Google Page Rank and visit the websites with our crawler.

3. **Merge results** The results from the first two steps are merged into final *Bulk group* features.

4. **Build models and classify** Models are built from the features and experiments are carried out.

Our goal was to propose a solution and develop a testing prototype, therefore we could divide the whole system into several 'independent' modules and implement each module with a different set of technologies (since each technology suits different needs). This approach helped us to decrease the time needed for the implementation, but it is not applicable for developing a final solution. The final system requires much more complex approach and must be implemented directly in the ESP's infrastructure. The implemented modules are **ProbeDataProcessor**, **DomainsCrawler**, **PageRankTool**, **DataAgregator** and **ModelingTool**. In Table 7.1 you can see the overview of the modules, the tasks they solve and the technologies which we have used to implement them.

The figure 7.1 depicts the roles of individual modules within the whole process. ProbeDataProccesor finds *Bulk groups*, extracts statistics for them and prepares a file containing unique domains (domains that have been detected either in mailFrom or in From fields). This list of domains is handed over to DomainsCrawler and to PageRankTool in order to gather domains information. All resulting files are then merged by DataAgregator which prepares a Feature file. The Feature file contains a list of particular *Bulk groups* and their statistics. At the end of the process it is used by the Models for classification. Since different modules often need to be launched on different machines (for example Map-Reduce tasks

Module	Technology used	Task
ProbeDataProcessor	Java, Hadoop, Protocol Buffers	Extract statistics from logs
DomainsCrawler	Python, Scrapy	Get size of websites
PageRankTool	Java	Get Google Page Rank
DataAgregator	Java	Merge data and prepare final features
ModelingTool	Matlab	Train models and classify

Table 7.1: Modules overview

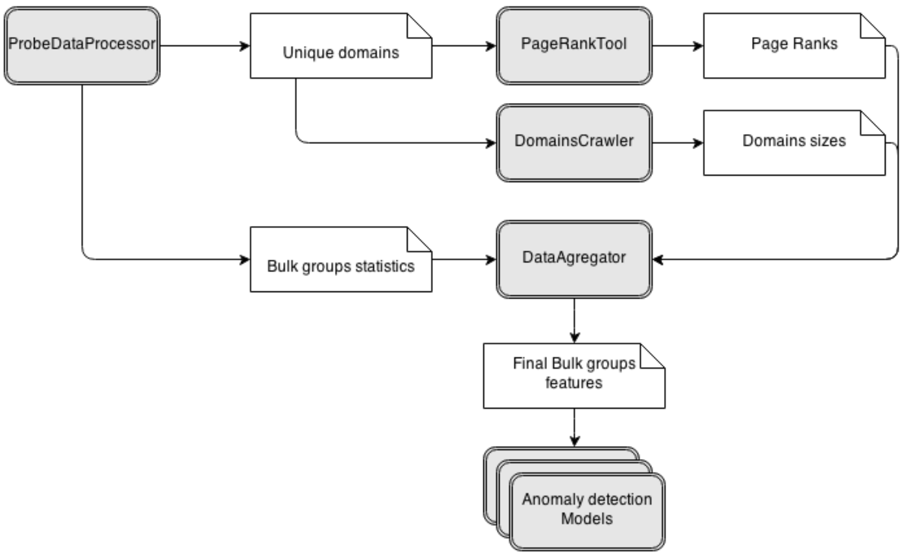


Figure 7.1: Process of feature extraction and classification.

in Hadoop cluster, crawlers on different machines, etc.), the whole system is not automated and we copy files and start modules manually. But for our purposes this does not mean a big problem.

7.1 ProbeDataProcessor

ProbeDataProcessor is a Java module responsible for extracting important statistics from log files. Before we dive into the details of the unit, we must briefly describe basic technologies used in this part of the system.

Protocol Buffers The contracting ESP stores the logs in the Protocol Buffers format. Protocol Buffers is a language-neutral, platform-neutral and extensible way of serializing structured data for use in communications protocols, data storage, etc. [17]. The format was originally developed at Google, but now it is available under open-source license. It was designed to be smaller and faster than other existing methods such as json or xml. The main building block of the method is a special schema-definition language, which allows developers to specify the underlying (platform-independent) structure of the data. Such definition is stored to *.proto* files. When a developer wants to read or write the data from the program, he can use protocol compilers (available for C++, Java, Python and various other languages) to create a language specific stub. Using the stub, data are serialized into binary format which is very compact but not self-describing and so the *.proto* file is always needed for data deserialization.

When we want to read the logs in our Java program we also need to compile a class which works as a java stub for the underlying logs structure (Our generated class can be found in package `cz.email`). The stub contains only methods for accessing the data fields. For practical usage we need methods which contain some application logic (i.e. combining values from multiple fields to discover whether an user has deleted an email without reading it, etc.) but on the other hand it is not recommended to extend the stub. So we do not use it directly, but we have created a class `RecordWrapper` (in package `cz.cvut.gogartom.probedata.entities`) which scans the stub, stores important fields and contains the application logic (this approach is known as Wrapper design pattern).

Hadoop The infrastructure used for storing and processing the log files used by ESP is Hadoop. Hadoop is a framework for distributed processing of large data sets across clusters of computers using simple programming models [1]. A detailed description of Hadoop architecture and its features is out of scope of this work. We will only mention that it provides a distributed file system (HDFS) which allows us to split large data sets across multiple machines. In order to keep the network traffic as low as possible, it exploits the locality of data and create computation as close to data as possible. Another requirement is ability to make the computations parallelized. This is achieved by using MapReduce programming model. This model is basically composed of two phases Map and Reduce. The whole model works with Key-Value pairs as follows: The input data set is split into multiple parts (and can be therefore saved to distributed file system). The first phase called Map takes one data element at a time and transforms it to Key-Value pair. Those pairs are then ordered and passed to Reduce function which merge the results and outputs the final Key-Value pair which is saved to resulting file. An example of MapReduce workflow is in figure 7.2. In this example we process email logs and compute how many similar messages we have received. At the beginning, the logs are split to smaller parts and stored on multiple machines, in the Map functions the hash string from the matching system (which identifies the content of the message) is extracted and returned as the key together with the count (which is always 1 in this example). After that, the key-value pairs are sorted and passed to the Reduce function, which computes the counts of the messages with similar hash string. These results are then merged in one output file. As you can see this programming model splits input files and works with key-value pairs. Therefore Hadoop also includes a specialized binary data format called Sequence file. This format perfectly suits the needs of MapReduce model, because it is designed to store the key-value pairs and it can be split easily (the boundaries of the individual pairs are flagged so we know where we can split the file).

Since we receive large amount of logs in Sequence files (the structure is depicted in Figure 7.3) and Hadoop perfectly suits the task of large dataset processing, we decided to use it for statistics extraction as well. The whole process carried out by ProbeDataProcessor is shown in Figure 7.4. Only few subtasks needed to exploit parallel MapReduce approach (those are

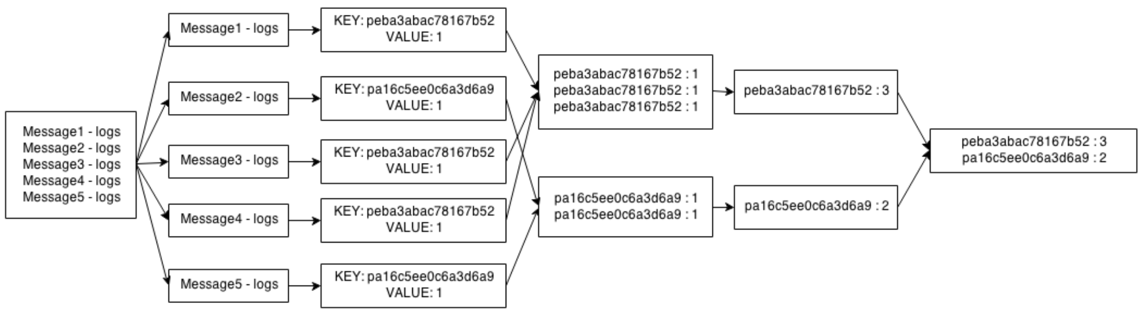


Figure 7.2: MapReduce example: counting emails with similar content

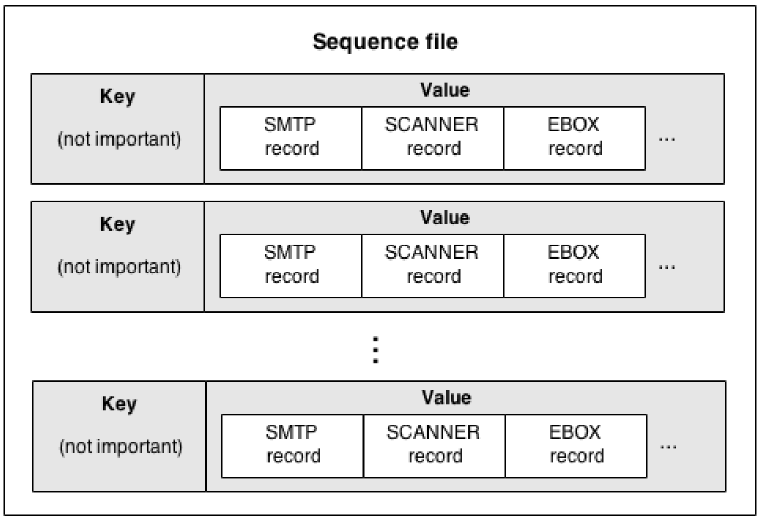


Figure 7.3: Structure of input Sequence file

marked with 'MR:' in the figure), the other tasks process smaller amount of data and could be computed locally.

The whole process begins in **FindBulkEmails** task. This Map-Reduce task counts hash strings from matching system in order to find the emails which arrived in large volume (the threshold for a bulk mail is set to 500 messages). From this step until the end of the classification process, the hash string is used as an identifier of the *Bulk group*.

The second subtask **GetStatisticsForBulkGroups** takes the hashes and gathers counts of MailFrom domains, IP addresses and From addresses for the marked bulk emails. The results after this step are stored in three Sequence files. The structure of the result file for From addresses is shown the Table 7.2[1] (the other two files have a similar structure).

We have already explained why we focus on the domains and not the whole address in the Mail From field. But extracting the second level domains from the address is not a trivial task (for example we cannot simply count dots in the address, because we would fail in

[1]Some addresses in the table were changed due to Privacy concerns.

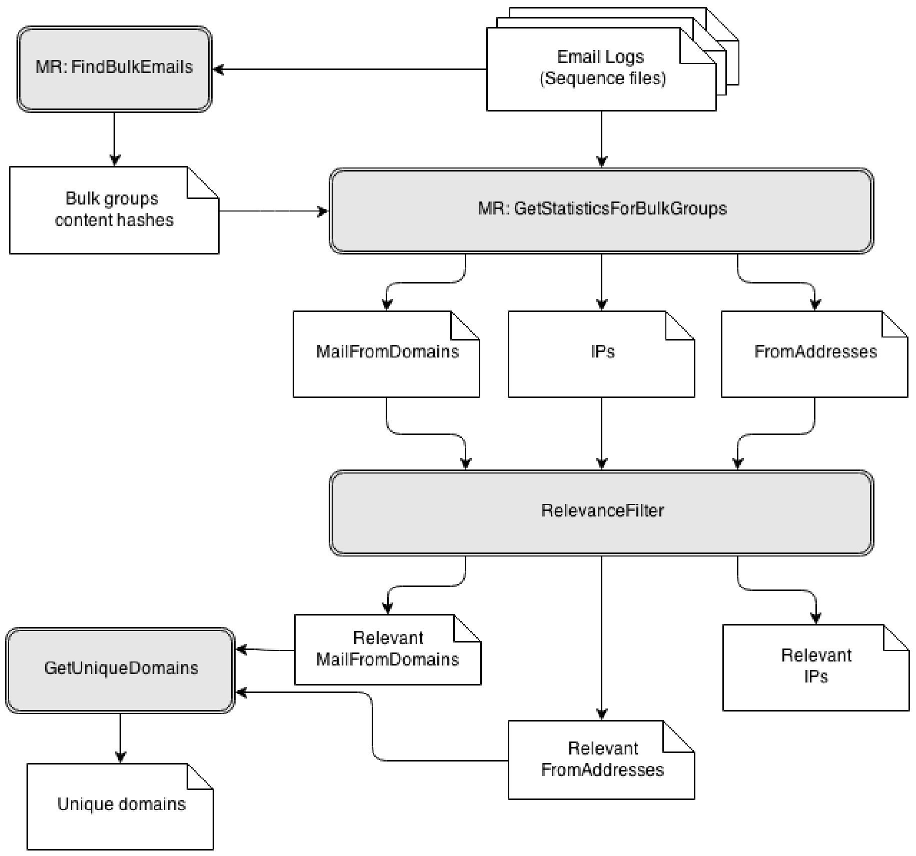


Figure 7.4: Structure of input Sequence file

extracting *co.uk* domains, etc.). Fortunately the Guava project (the open source project which publishes some Google core libraries) includes the tools for such extraction [16], so we could utilize that.

Key (Hash string)	**Value**
pd622c3f2f43b4566	info@aaapoptavka.cz : 17030 hidden.name@gmail.com : 2 hidden.name@iol.cz : 1
p9e5ca5e56950ecee	slevy@vykupto.cz : 1746
p6ab127569b0fbb04	betclic@inzerce-news.com : 5055 betclic@betclic-news.com : 768 hidden.name@quick.cz : 1
...	...

Table 7.2: Example of statistics for From addresses

Another problem which we had to solve is that the gathered statistics contained redirect noise. Redirect noise is caused by the users who set their other email accounts to redirect emails to our addresses. The contents of the messages still match the *Bulk group* but the addresses, domains and IPs are different than those of the original sender. This phenomenon negatively influences the resulting statistics as it can be seen in our example in Table 7.2 - the addresses *hidden.name@gmail.com* and *hidden.name@iol.cz=1* used to send the first email are very likely from redirects, because their usage is not very significant. So in the subtask RelevanceFilter, we filter out non-significant addresses by normalizing the counts by the maximum value and we keep only the addresses whose normalized value exceeds some specific threshold (we have used 0.02 in our tests), other addresses are discarded as possible redirects.

If all the previous tasks finished successfully, the list of unique domains is extracted. This list is subsequently used by other modules which gather the domains information. The described workflow is automatically executed from the Main class which resides in package `cz.cvut.gogartom.probedata.`, the sources for Map-Reduce tasks can be found in package `cz.cvut.gogartom.probedata.mr` and the local ones in `cz.cvut.gogartom.probedata.local`. The sources also contain other additional parts, which are not mentioned in this thesis, since they were used only for evaluation purposes and are not crucial for the classification.

Because we launch the ProbeDataProcessor repeatedly (on different data sets), we need to pass the threshold settings and paths to the input and output folders. This is done by specifying the settings in the *config.xml* file, which is shown in Appendix B. The path to input folder (the folder where ESP's sequence files reside) is set in element *probeDataPath* the output folder is specified in the *dateFolderPath* (because we compute statistics for specific dates). The other elements specify working subdirectories and thresholds.

7.2 Domains crawler

After acquiring *Bulk groups* statistics we need to decide whether the domains used to send messages are suspicious (see our definition of suspicious domain in Chapter 6). This module crawls the web sites which reside on the tested domains and estimates their size (i.e. how number of unique pages accessible from the root page). Implementing own reliable crawler is very complex problem and so we have used an open-source python library called Scrapy [29]. One of the basic class in the Scrapy framework is a Spider. This class defines what should be scrapped from the page and how the outgoing links should be followed. To serve our needs we set the Spider to:

- Stay on the selected domain
- Visit at maximum 10 pages (because we don't need to crawl the large websites)
- We count the pages returned with HTTP Status: 200 OK

The framework (with our customized spider) is launched from the bash script. Two files are passed as the parameters to the script. The first file contains the domains which need to be crawled and the second one contains already crawled domains. Since the crawling is very time consuming part of the whole process, we do not crawl domains which we have already crawled before. Another way of speeding up the process is to use multiple spiders - so at the beginning of the script the *new-domains* file is split into multiple parts and each part is crawled by one Spider.

7.3 PageRankTool

As the name of this tool suggests, it is responsible for collection domains PageRanks. It is also launched from the bash script with two parameters - paths to files with *new-domains* and *already-checked-domains*. The script starts a Java program which uses API originally used by the Google Toolbar[2]. It creates the same HTTP query as the toolbar does and gets the response from Google servers. Unfortunately this approach does not allow us to gather the PageRanks quickly. We can send one query per 2 seconds, so that our IP address is not automatically banned by Google. The script and Java program are pretty simple, the only tricky part of this module is the function used to hash the address for the query - the Jenkins hash function. The computation of the function can be found `cz.cvut.gogartom.pagerank.JenkinsHash`.

7.4 DataAgregator

DataAgregator is a simple Java program which merges the results of the previous modules. It accepts two parameters which represent paths to two folders - the first folder contains the results from ProbeDataProcessor and the second contains Page Ranks and sizes of the domains. The DataAgregator merges the results and creates a *.csv* file which contains the final features for all *Bulk groups* [3].

[2]http://www.google.com/intl/cs/toolbar/ie/index.html

[3]It also creates a second .csv file with users feedback, which is used only for data labeling purposes and it is not important for the classification.

7.5 Models

The very last module in the whole process is implemented in Matlab and it contains source codes for particular models and scripts for testing their performance. Every model was implemented as Matlab class and has two public methods: **trainModel** (which accepts training data set and estimates the model parameters) and **getScores** (which returns *Normal behavior score* for test instances). All important scripts, functions and classes are summarized in Tables 7.3, 7.4 and 7.5

Script name	Description
run	Main script successively runs training, validation and testing
loadData	Loads all available data
loadLabels	Loads labels for marked data
trainModels	Takes training data and prepares all models
validationScript	Runs validation procedures
testScript	Runs testing procedures
plotModels	Plots PDFs of all models
plotFeatureSpace	Plots training data in the feature space
modelConsistency	Plots models trained on different data sets
mergeTwoDays	Merges data from two data sets
labelEmails	Script used to manually label emails
analyzeEmails	Script used to manually analyze content of emails

Table 7.3: Scripts overview

Function name	Description
readDataCSV	Reads .csv file with features
readFeatureNamesCSV	Loads feature names from .csv
readFeedbackCSV	Reads users feedback (used for labeling only)
getClassificationStats	Returns TPR, FPR, Precision and F1 score for classifier

Table 7.4: Functions overview

Class name	Description
normalDistModel	Univariate Normal Distribution model
multivarNormalDistModel	Multivariate Normal Distribution model
histogramModel	Histogram model
parzenModel	Parzen Window model

Table 7.5: Classes overview

Chapter 8

Experiments

In this chapter we will further analyze data, train models and test them on the labeled data sets. All those steps are carried out in order to solve the following issues:

- **Is a distribution of unlabeled data consistent over some meaningful interval?** If we want to model a normal behavior of senders, we need the model to stay valid at least for few days.
- **Set the parameters for anomaly detection task.** Some parameters are set directly from unlabeled data using maximum likelihood estimate, but some parameters need to be set based on the labeled data (i.e. thresholds for the classifiers).
- **Do anomalies in sending behavior correlate with spam emails?** We have manually identified anomaly emails that were most probably spam, but does this assumption hold in the whole test set.
- **Compare the proposed models.** Each model should be able to detect anomalies, but how do they perform in spam detection task?
- **What is the expected performance of the system?** We will measure the performance of the system on the small subset of labeled data and discuss whether the technique of anomaly detection is useful for human operator.

8.1 Model consistency

Two-sample Kolmogorov–Smirnov test is a statistical test that let us estimate whether two sets of one-dimensional data were generated by the same underlying probability distribution. Unfortunately this test requires the underlying distribution to be continuous, which is not in our case. Therefore we couldn't use it and we had to check the consistency by visually comparing the data distributions for particular days. Because we use the probability density functions (PDF) for modelling the normal behavior, we have used Parzen Windows to estimate the distributions and plotted the PDFs for multiple days (see Figure 8.1). Since in the later testing phases we would like to simulate the normal scenario and use the first two days from February as our unlabeled training set, we have merged these days into one distribution. As you can see the PDF for the two days from February (red line) has very similar shape as the functions for the other days. This means that the normal behavior of the senders stayed the same for the 66 days and it suggests that we can use the models to describe the behavior. Of course for more detailed and long-term consistency testing more data is needed, but for our purposes we assume that models stay consistent.

8.2 Data splitting and labeling

When we know that data are consistent across multiple days, we can define training, validation and test sets. The Table 8.1 shows how the data will be used - we will define normal behavior on the earliest dataset from February (to simulate the real usage). We will use two days from the April as validation data to estimate the parameters and very last day will be used for testing purposes. Since we would like to give at least rough estimate how the algorithm performs, we needed to label validation and testing data. As we have said several times - this task is not easy and therefore we have not managed to label a large number of *Bulk groups*.

Labeling procedure For the labeling purposes we have used the feedback from the users - we have analyzed how they used the “mark as spam” button as a hint for manual labeling. We have manually checked the emails with highest “spamminess” and only if we were pretty sure that the email is spam, we have marked it. The same manual procedure was carried out to label ham - there we have analyzed emails with the lowest “spamminess” and only if the sender was known and reliable internet entity we have marked it as ham.

The Table 8.1 summarizes how many *Bulk groups* we have managed to label. For the validation and testing purposes, the ratio of the spam and ham should be kept as in reality. Unfortunately, no one knows the ratio in our situation, so we will use all the labelled data we have. But it’s worth to mention, that for these reasons, the numeric estimates of efficiency are rather tentative.

Usage	Statistics	Labeled Bulk groups (Ham/Spam)
Model training	02/08/2014	– / –
	02/09/2014	– / –
Validation	04/08/2014	51 / 34
	04/14/2014	57 / 35
Testing	04/15/2014	48 / 34

Table 8.1: Data usage overview

In this chapter all tests are carried out on all Bulk mails which arrived to the ESP. But as we have already stated, this component should be used together with the reputation database and only the senders which have not been seen before should be checked for anomalies. Unfortunately the reputation database is not implemented yet, so we have to test the performance on all bulk emails.

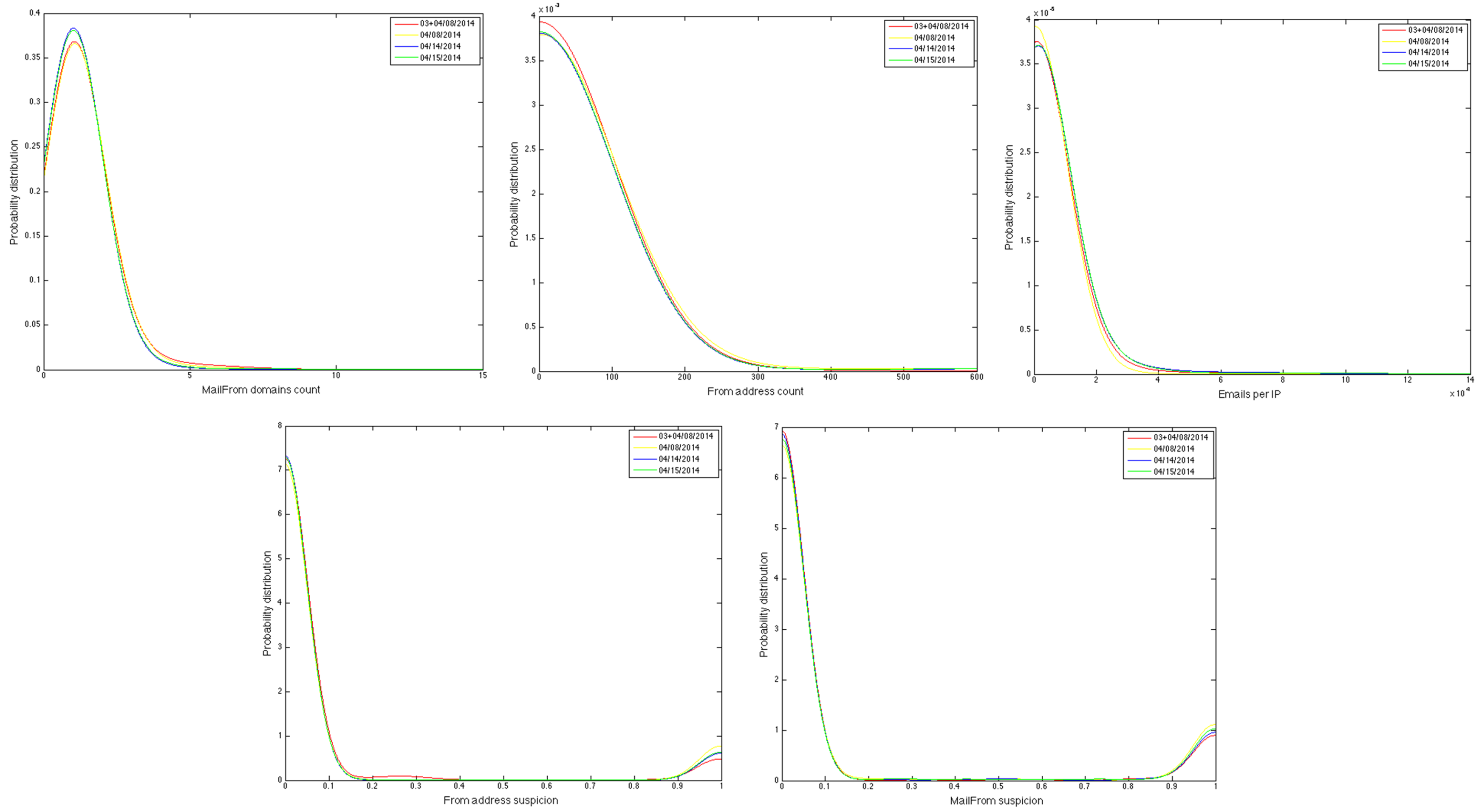


Figure 8.1: Unlabeled data distribution for multiple days

8.3 Unsupervised learning

In the first phase we have trained models on the unlabeled data set from February. We have trained all models discussed in Chapter 6 and some of them are depicted in Figure 8.2. The plots shows Probability density functions in all five dimensions for *Univariate normal distribution model* (red curve), *Histogram model* (8 bins settings, blue curve) and *Parzen windows* (green curve). As you can see all models correctly capture the peaks of standard behavior. On the other hand the Normal distribution model cannot correctly capture the smaller peaks in the 'address suspicion' features. This could be solved by using Gaussian mixtures models but that's what Parzen windows model actually does. The histogram model can detect the other peaks, but since it is not continuous it cannot describe the distribution as precisely as the other models. This could be solved by using smaller bins, but this approach brings other unwanted consequences and will be discussed later.

8.4 Model validation

The validation phase uses the performance of the models on the labeled dataset to estimate the suitable parameters for the final classifier. The parameters which we want to estimate are:

- Number of bins for Histogram model
- Classification thresholds for every model

We also have to set the standard deviations for Parzen window model. Unfortunately we do not have enough validation data to set the 5 parameters automatically. On the other hand we have pretty good picture of the normal behavior, so we could set the deviations manually. The final parameters which we have used for the Parzen window model are [1, 100, 10000, 0.05, 0.05].

8.4.1 Number of bins

We have discussed how the number of bins in Histogram model influences the anomaly detection - more bins allow more accurate distribution description, but on the other hand the sparse histogram can result in false positive errors. We have trained models with 10, 20, 30 and 50 bins and plotted their ROC curves in Figure 8.3. ROC curve plots the True Positive Rate (TPR) and False Positive Rate (FPR) for various threshold values. The models with more bins seem to perform slightly better on the validation set. The problem of false positives is not very apparent but if we plot the Precision-Recall graph (as in Figure 8.4), we can see that the curve for model with 50 bins starts at the precision lower than 0.9. This means that some ham *Bulk groups* received score 0 and will be always classified as spam (no matter on the chosen threshold). This is not a desired behavior and for our task we would rather use a less sparse model. The model with 30 bins performs just slightly worse but it didn't give any ham Bulk group score 0, so in the rest of this work we will use that model as a representative of this group.

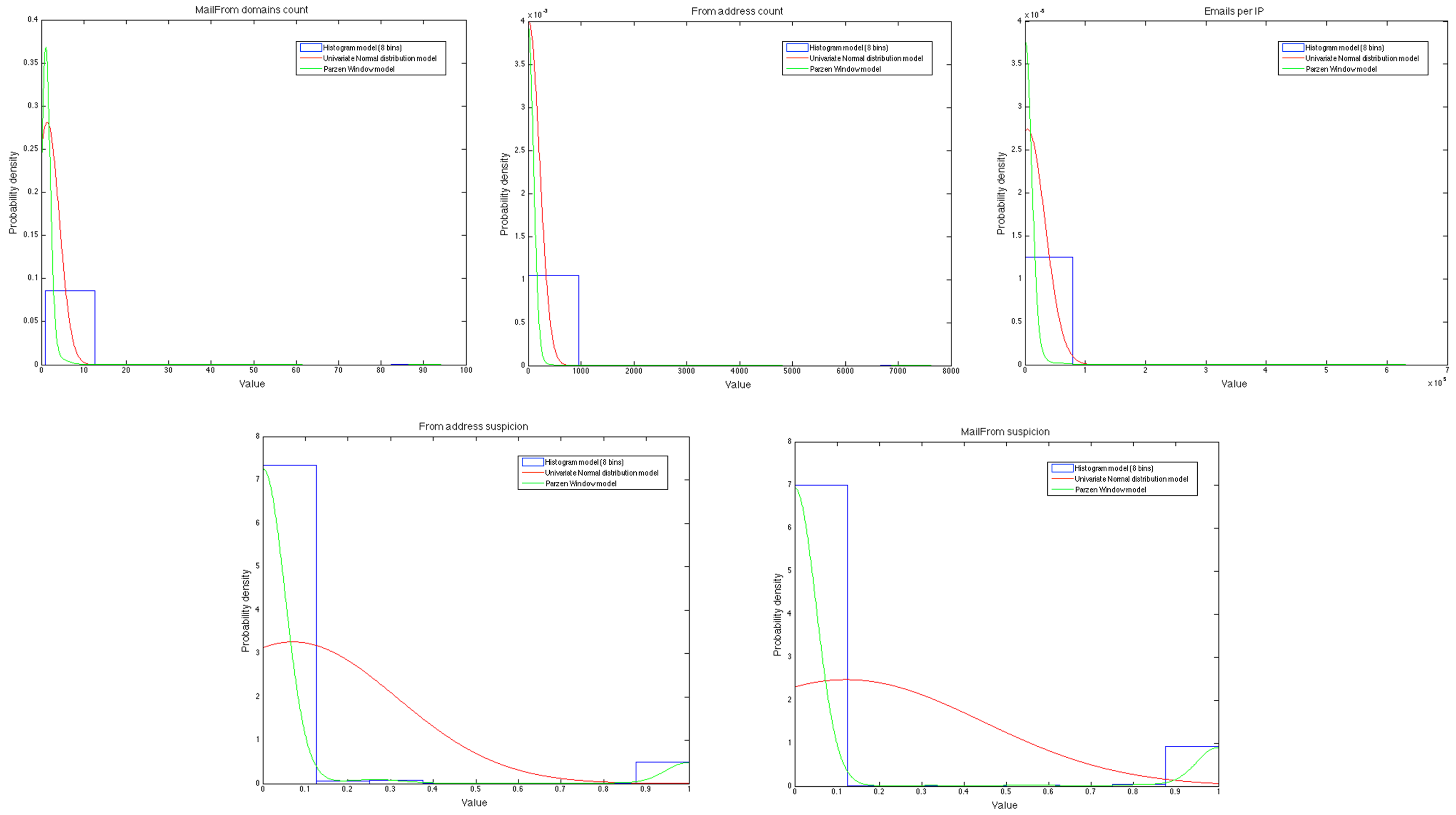


Figure 8.2: Models of normal sending behavior

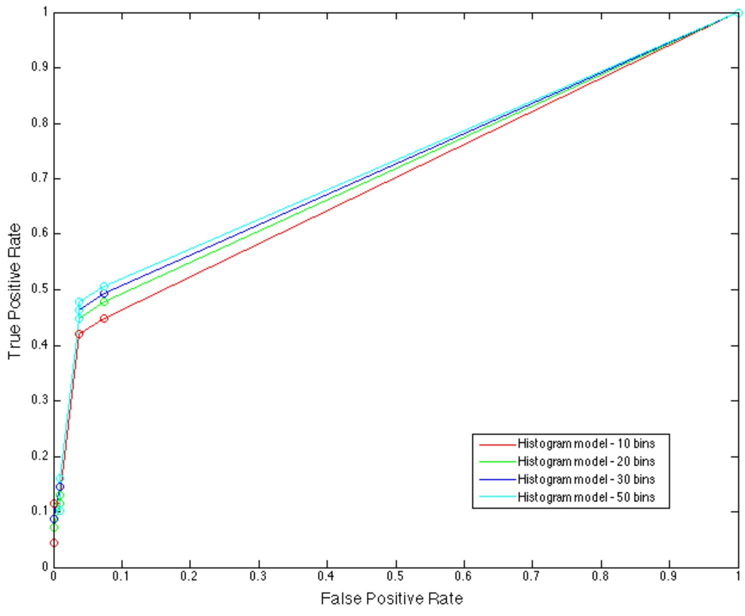


Figure 8.3: ROC curves for Histogram models

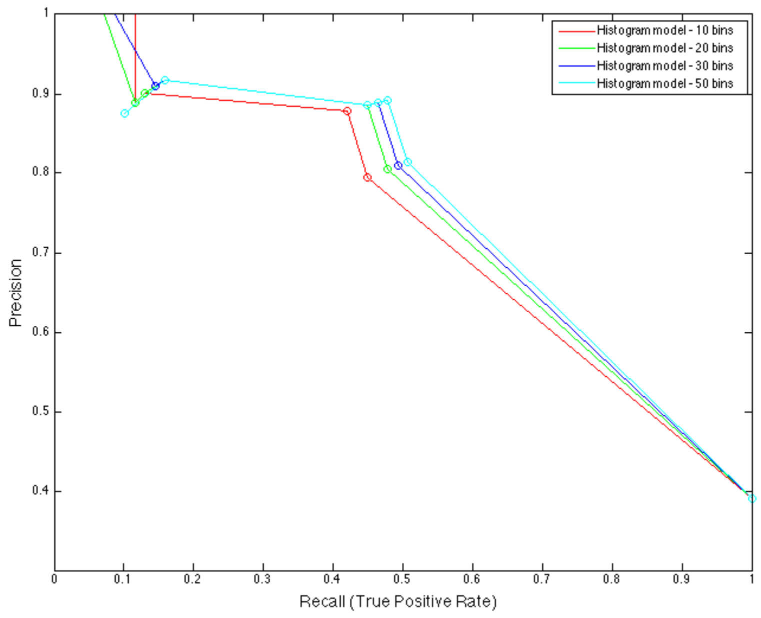


Figure 8.4: Precision-Recall graph for Histogram models

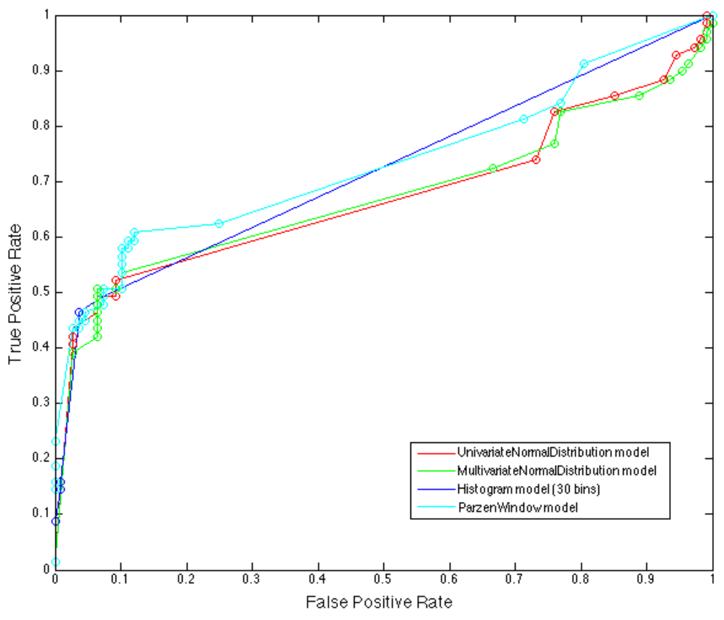


Figure 8.5: ROC curves for all models

8.4.2 Classification thresholds

In this section we focus on the proposed models and we examine how the anomalies correlate with received spam. In the Figure 8.5 we have plotted ROC curves of our models on the validation dataset. ROC curve plots the True Positive Rate (TPR) and False Positive Rate (FPR) for various threshold values. The individual values are marked as circles and are connected with lines. Since every model created different Probability density function to describe normal behavior, the maximum possible *Normal behavior score* is different for each model. Therefore we cannot use the same thresholds for all models, but we use 1000 linearly spaced values between 0 and $MaxScore_i$, where $MaxScore_i$ is the maximum possible score for model i.

If we look at the ROC curves, we can see that for the lower thresholds all classifiers performs pretty well - all of them are able to recognize more than 40% of spam (0.4 TPR) with less than 6% false positive rate. The ROC curves grows steeply at the beginning but at certain point (approximately 0.08 FPR and 0.5 TPR) the curves lay down and go linearly to the endpoint (point at [1,1] where all *Bulk groups* are classified as spam). We can also see many points at the beginning of the curves, almost no points in the center and some points at the end. This suggests that *Bulk groups* usually receive either low scores or high scores. Moreover, every curve should be formed by 1000 points, but there is not so many points. This implies that many *Bulk groups* receive similar score.

In order to validate the mentioned hypotheses we have plotted histogram of scores for our labeled validation data in Figure 8.6 [1]. We can see two peaks in the histogram - the higher peak (more than 75%) represents *Bulk groups* that received the highest scores (these senders behave normally) and the second peak (20%) which represents the *Bulk groups* with the lowest scores (anomaly senders). The vast majority of the 'anomaly peak' consists of spam and that is the reason why the ROC curves grows steeply at the beginning. On the other hand we can see that approximately the same amount of spam was sent by non-anomaly senders. These are the sender's which we will never detect with anomaly detection algorithms based on these features. Since we could bring some bias into the problem by using only 177 manually labeled *Bulk groups* (where only evident spam or ham was labeled), we have plotted the same histogram for unlabeled data set (4700 *Bulk groups*) in Figure 8.7. We can see that the score distribution is still the same and it gives us good chances that we will be able to detect the spammers which behave abnormally.

If we look at the ROC curves for particular models we can see that Univariate and Multivariate Normal Distribution models perform almost similarly - it suggests that there is not a strong correlation between the individual features. Based on the observations from histogram in Figure 8.7, the performance for the lower thresholds is crucial for our task. Parzen Window model seems to perform the best in first half of the curve and because we need to choose one model for the manual analysis, it is our choose for the next phases. The interesting threshold values and the measured performance of the Parzen Window model are summarized in Table 8.2. In this table we don't use Accuracy as a standard measure of performance, because it is not appropriate for anomaly detection problems [2]. Instead, we use $F_{0.5}$ score, which is a measure that combines precision and recall of the classifier while it puts more emphasis on the precision. The equation for the score is in 8.1.

$$F_{0.5} = (1 + 0.5^2).\frac{precision.recall}{0.5^2.precision + recall} \tag{8.1}$$

Threshold	**TPR/FPR**	**$F_{0.5}$ score**
2.18e-08	0.23/0	0.60
3.49e-07	0.46/0.05	0.74
2.71e-06	0.61/0.12	0.73

Table 8.2: Interesting thresholds for Parzen Window model

8.5 Testing and manual analysis

At the end we have used the Parzen Window model to classify *Bulk groups* from the last day. The results for three threshold settings are shown in Table 8.3. The performance for the test set confirms the results from previous sections, but it must be noted that these results are very tentative, since the performance was measured on the very small subset of labeled

[1]We have chosen Histogram model because the behavior is very apparent for its scores. But other models behave similarly.

[2]It is sufficient to classify everything as a non-anomaly and the Accuracy would be high.

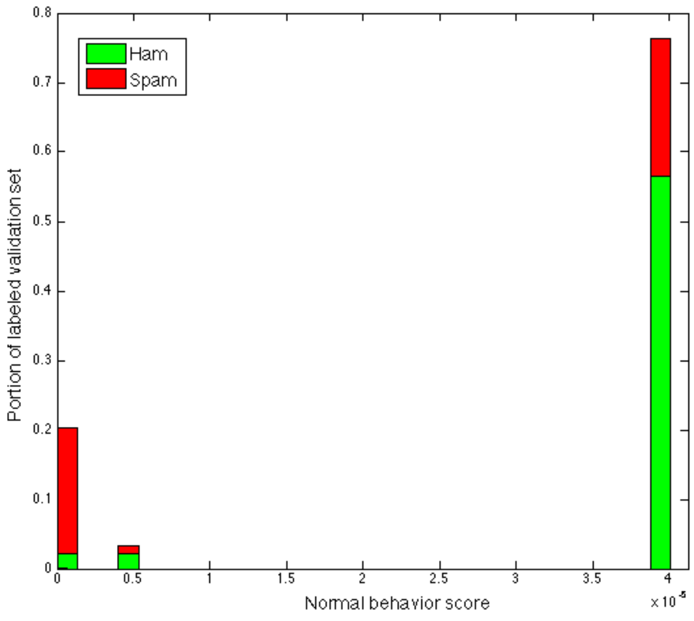


Figure 8.6: Histogram of labeled Bulk groups based on scores from Histogram model

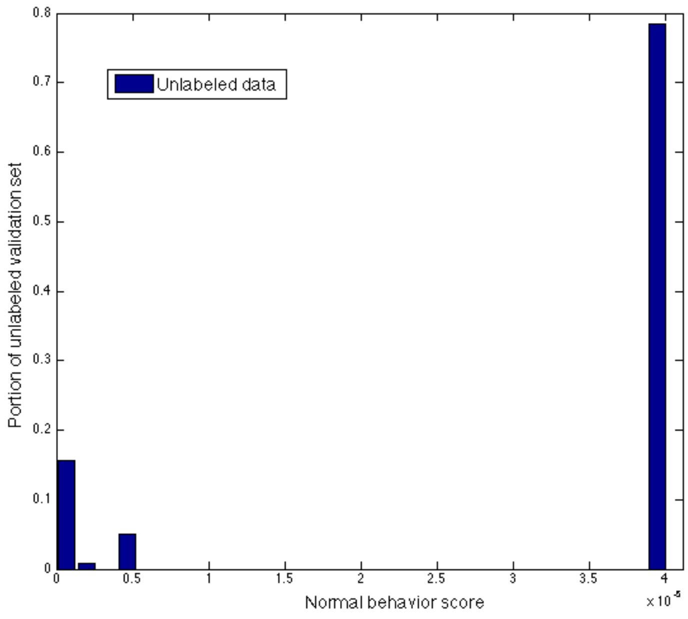


Figure 8.7: Histogram of unlabeled Bulk groups based on scores from Histogram model

data. The last column in Table 8.3 shows how many unlabeled Bulk groups were classified as anomalies. Since these are the emails that will be shown to the human operator, our goal should be not to overwhelm him with to many anomalies, because that would make the tool unusable. In other words - precision is more important than recall in our task.

Threshold	**TPR/FPR**	$F_{0.5}$ **score**	**Anomaly Bulk groups**
2.18e-08	0.32/0.04	0.64	270
3.49e-07	0.53/0.04	0.79	516
2.71e-06	0.76/0.13	0.80	1174

Table 8.3: Performance of Parzen Window model on testing data set

In the next phase we have taken the 516 *Bulk groups* which received the *Normal behavior score* smaller than 3.49e-7 and manually analyzed the content. We have focused on the topic of the email and the sending entity (the sending entities were intentionally estimated based on the content of an email, not from the headers).

Email topics The topic distribution of anomaly emails can be found in a pie chart in Figure 8.8. As we can see almost half of the anomalies advertised some goods or services available online. How many of these advertisements are illegal newsletters or legal grey emails is still the question which need to be analyzed by the professionals, but we can say that the behavior of these senders is suspicious. But there are other interesting categories of emails which promote *Online casinos*, *Dating sites*, *Earning opportunities*, *Loans*, *Pharmacy products*, *Insurance* or emails with *sexual content*. These categories make together 37% of anomalies and are very likely unwanted bulk mails. The last category *Others* contains all other content such as jokes, weather forecasts, watchdogs, informational emails, etc (lot of those emails are very likely false positives).

Sending entities While analyzing the senders based on the content, we have discovered that for 11% of the emails we cannot determine the name of the sending entity and such self-presentation is very suspicious. On the other hand we have also found some trustful companies among the anomaly emails and those emails were most probably false positivies. These emails were usually marked as anomalies because companies send emails from special domains (for example company *XY* sends emails from *news-XY.com* or *info-XY.com*), where they do not host any website. Fortunately those domains should be included in a reputation database and therefore these emails won't be passed to the anomaly detection unit in the final implementation.

8.6 Computational requirements

Since the models used for anomaly detection are pretty simple, the training and classification phases do not require much computational power. The complexity of training Parzen Window model is $O(n)$ and the classification phase is $O(nm)$ where n is a number of training

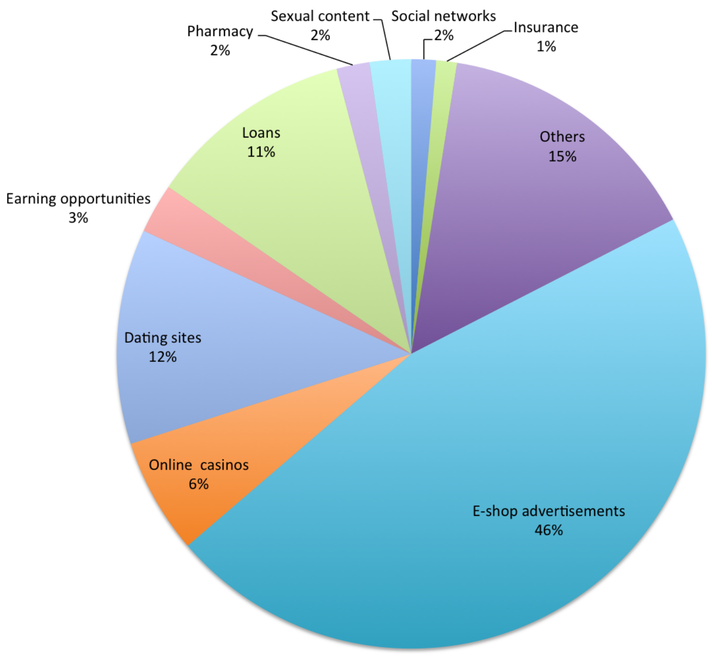


Figure 8.8: Topics of anomaly emails

instances and m is a number of testing instances. So the biggest issue (in terms of time requirements) is a process of crawling and acquiring Page Ranks. Fortunately, the contracting company has its own search engine and therefore they already have pre-crawled data with their own ranking function, so this phase can be skipped in the final implementation.

8.7 Summary

Even though we could not use statistical methods to check consistency of normal behavior, from the plotted distributions it is apparent that the behavior of senders remained very similar between the February and April 2014. The preliminary results (from the small set of labeled data) show that approximately half of the spammers behave abnormally while the vast majority of legal senders behave as expected. This observation implies that the proposed tool cannot detect all the spammers, but on the other hand it should operate with sufficient precision. High precision is a very good property for such semi-automated system, because we don't want to overwhelm the operator with too many false positives.

Chapter 9

Future work

9.1 Further testing

The first measured results look promising, but in order to get the more accurate insight into how the system performs we need to carry out further testing:

- **Testing with more feedback data** We have used feedback from users (how they use 'mark as spam' button) for data labeling purposes. Since we had the data only for particular days the resulting statistics weren't as informative as they can be, if we used the feedback for multiple days.
- **Test the algorithm with different crawlers and ranking function** As we have already stated, we are planning to use already pre-crawled data a the proprietary ranking function. We need to measure how this change will effect the results.
- **Testing together with reputation database** Since the system is supposed to work as a complement to the reputation database we should test them together. We think that if our system receives message only from senders which are not in the reputation database the false positive rate will decrease.

9.2 Final implementation

If the the tests which we have mentioned above give satisfactory results, we would like to implement the final solution together with the ESP in its infrastructure. The final system should be able to:

- Automatically retrieve the logs and content from the servers
- Interactively change anomaly detection threshold
- Sort anomaly *Bulk groups* based on multiple criterions (such as total anomaly score, anomaly score of individual features, number of received messages, etc.)

9.3 Other possible features

There are still many other signals which can be used for the anomaly detection. The implementation in the infrastructure will allow us to more easily add and test the new features. The promising signals are:

- Geolocation of sending machine
- Temporal behavior of senders
- Automatic topic analysis of the content
- Outgoing links and their landing pages

Chapter 10

Conclusions

Our task was to propose a system which will be used by a human operator to detect unknown senders of Illegal newsletters. This is a task of detecting unsolicited bulk mails which resembles other usual grey mail and therefore it is not detected by the existing filters and ends up in the users inboxes.

Our proposed system focuses on the masking behavior of the spammers. We have proposed features that should describe the self-presentation habits of the senders. Those features were then extracted from the large set of unlabeled data (tens of millions of emails) and were used to train statistical models of normal behavior. In the testing phase the new instances are compared against the normal behavior model and they receive *Normal behavior score*. If the score is lower than a predefined threshold we consider the email as suspicious and it will be shown to the operator.

After implementing the prototype of the system, we tried to evaluate its performance, but the whole experimentation phase was influenced by the lack of labeled data. Overall, we have managed to label 259 emails which we have used during the evaluation and testing. In the evaluation phase we specified the parameters of the models and then we tested the performance of the classifier on the testing dataset. The results on this small labeled set show that our system should be able to detect approximately 50% of spam while keeping the false positive rate under 5%. Even though such results are very promising, we take them as very tentative and we are planning more tests on larger datasets.

We have also analyzed the distribution of Normal behavior score for labeled emails, and it suggests that approximately half of the spammers do not behave abnormally and therefore we won't be able to detect them by our system. On the other hand there are also ham emails which behave unusually, but since this system is not intended as an automatic spam filter, but rather as a tool which should help human operators, we hope that it can still give them a new and useful perspective on the suspicious emails.

Bibliography

[1] APACHE.ORG. Hadoop project, 2014. Available at: <http://hadoop.apache.org/>. [Online; accessed 31-March-2014].

[2] BLANZIERI, E. – BRYL, A. A survey of learning-based techniques of email spam filtering. *Artificial Intelligence Review*. 2008 2008, 29, 1, s. 63–92.

[3] BOYKIN, P. – ROYCHOWDHURY, V. Personal Email Networks: An Effective Anti-Spam Tool. Technical report, University of California, Los Angeles, February 2004. Available at: <http://arxiv.org/abs/cond-mat/0402143>.

[4] BOYKIN, P. – ROYCHOWDHURY, V. Leveraging social networks to fight spam. *Computer*. April 2005, 38, 4, s. 61–68. ISSN 0018-9162. doi: 10.1109/MC.2005.132.

[5] BRITANNICA. Gary Thuerk, 2014. Available at: <http://www.britannica.com/EBchecked/topic/1668817/Gary-Thuerk>. [Online; accessed 2-February-2014].

[6] CARRERAS, X. – MàRQUEZ, L. – SALGADO, J. G. Boosting Trees for Anti-Spam Email Filtering. In *In Proceedings of RANLP-01, 4th International Conference on Recent Advances in Natural Language Processing, Tzigov Chark, BG*, s. 58–64, 2001.

[7] CHANDOLA, V. – BANERJEE, A. – KUMAR, V. Anomaly Detection: A Survey. *ACM Comput. Surv.* July 2009, 41, 3, s. 15:1–15:58. ISSN 0360-0300. doi: 10.1145/1541880.1541882. Available at: <http://doi.acm.org/10.1145/1541880.1541882>.

[8] CHOWDHURY, A. et al. Collection Statistics for Fast Duplicate Document Detection. *ACM Trans. Inf. Syst.* April 2002, 20, 2, s. 171–191. ISSN 1046-8188. doi: 10.1145/506309.506311. Available at: <http://doi.acm.org/10.1145/506309.506311>.

[9] COOPER, J. – CODEN, A. – BROWN, E. A novel method for detecting similar documents. In *System Sciences, 2002. HICSS. Proceedings of the 35th Annual Hawaii International Conference on*, s. 1153–1159, Jan 2002. doi: 10.1109/HICSS.2002.994037.

[10] CROCKER, D. – HANSEN, T. – KUCHERAWY, M. RFC 6376: DomainKeys Identified Mail (DKIM) Signatures. Technical report, The Internet Engineering Task Force, September 2011. Available at: <tools.ietf.org/html/rfc6376>. Status: INFORMATIONAL.

[11] DREDZE, M. – GEVARYAHU, R. – ELIAS-BACHRACH, A. Learning Fast Classifiers for Image Spam. In *Conference on Email and Anti-Spam*, 2007.

[12] DRUCKER, H. – WU, S. – VAPNIK, V. Support vector machines for spam categorization. *Neural Networks, IEEE Transactions on*. Sep 1999, 10, 5, s. 1048–1054. ISSN 1045-9227. doi: 10.1109/72.788645.

[13] FETTERLY, D. – MANASSE, M. – NAJORK, M. On the Evolution of Clusters of Near-Duplicate Web Pages. In *Proceedings of the First Conference on Latin American Web Congress*, LA-WEB '03, s. 37–, Washington, DC, USA, 2003. IEEE Computer Society. Available at: <http://dl.acm.org/citation.cfm?id=951953.952397>. ISBN 0-7695-2058-8.

[14] GOLBECK, J. – HENDLER, J. Reputation Network Analysis for Email Filtering. In *Proc. Conference on Email and Anti-Spam (CEAS)*, Mountain View, USA, July 2004.

[15] GOODMAN, J. – CORMACK, G. V. – HECKERMAN, D. Spam and the ongoing battle for the inbox. *Commun. ACM*. 2007 2007, 50, 2, s. 24–33.

[16] GOOGLE. Guava project, 2014. Available at: <http://http://code.google.com/p/guava-libraries/>. [Online; accessed 31-March-2014].

[17] GOOGLE. Protocol Buffers library, 2014. Available at: <http://code.google.com/p/protobuf/>. [Online; accessed 31-March-2014].

[18] GROUP, T. R. Email Statistics Report 2013-2017, April 2013. Available at: <http://www.radicati.com/wp/wp-content/uploads/2013/04/Email-Statistics-Report-2013-2017-Executive-Summary.pdf>.

[19] HAJISHIRZI, H. – YIH, W.-t. – KOLCZ, A. Adaptive Near-duplicate Detection via Similarity Learning. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '10, s. 419–426, New York, NY, USA, 2010. ACM. doi: 10.1145/1835449.1835520. Available at: <http://doi.acm.org/10.1145/1835449.1835520>. ISBN 978-1-4503-0153-4.

[20] KASPERSKY-LAB. Spam Statistics Report Q2-2013, 2013. Available at: <http://http://usa.kaspersky.com/internet-security-center/threats/spam-statistics-report-q2-2013>.

[21] KLENSIN, J. Simple Mail Transfer Protocol. Request for Comments: 2821, The Internet Engineering Task Force, April 2001.

[22] KLENSIN, J. Simple Mail Transfer Protocol. Request for Comments: 5321, The Internet Engineering Task Force, October 2008.

[23] LEVINE, J. RFC 5782: DNS Blacklists and Whitelists. Technical report, The Internet Engineering Task Force, February 2010. Available at: <tools.ietf.org/html/rfc5782>. Status: INFORMATIONAL.

[24] PANTEL, P. – LIN, D. SpamCop:A Spam Classification & Organization Program. In *In Learning for Text Categorization: Papers from the 1998 Workshop*, s. 95–98, 1998.

[25] PARZEN, E. On Estimation of a Probability Density Function and Mode. *The Annals of Mathematical Statistics*. 1962, 33, 3, s. pp. 1065–1076. ISSN 00034851. Available at: <http://www.jstor.org/stable/2237880>.

[26] POSTEL, J. B. Simple Mail Transfer Protocol. Request for Comments: 821, The Internet Engineering Task Force, August 1982.

[27] RESNICK, P. RFC 5322: Internet Message Format. Technical report, The Internet Engineering Task Force, October 2008. Available at: <http://tools.ietf.org/html/rfc5322>.

[28] SAHAMI, M. et al. A Bayesian Approach to Filtering Junk E-Mail. In *Learning for Text Categorization: Papers from the 1998 Workshop*, Madison and Wisconsin, 1998. AAAI Technical Report WS-98-05.

[29] SCRAPY. Scrapy crawling framework, 2014. Available at: <http://scrapy.org/>. [Online; accessed 31-March-2014].

[30] SISSON, G. DNS Survey: October 2010. Technical report, The Measurement Factory, November 2010. Available at: <http://dns.measurement-factory.com/surveys/201010/dns_survey_2010.pdf>.

[31] TAYLOR, B. Sender Reputation in a Large Webmail Service. In *CEAS*, 2006.

[32] TELECOMMUNICATIONS, V. M. – METSIS, V. Spam Filtering with Naive Bayes – Which Naive Bayes? In *Third Conference on Email and Anti-Spam (CEAS*, 2006.

[33] WANG, Z. et al. Filtering image spam with near-duplicate detection. In *In Proceedings of the Fourth Conference on Email and AntiSpam, CEAS'2007*, 2007.

[34] WIKIPEDIA. Email — Wikipedia, The Free Encyclopedia, 2014. Available at: <http://en.wikipedia.org/wiki/Email>. [Online; accessed 2-February-2014].

[35] WONG, M. – SCHLITT, W. RFC 4408: Sender Policy Framework (SPF) for Authorizing Use of Domains in E-Mail. Request for Comments: 4408, The Internet Engineering Task Force, April 2006.

[36] WU, C.-T. et al. Using visual features for anti-spam filtering. 3, s. 509–512, 2005. Available at: <http://dx.doi.org/10.1109/icip.2005.1530440>.

Appendix A

Contents of attached CD

Directory structure:

- **Sources**
  - **ProbeDataProcessor**
  - **DomainsCrawler**
  - **PageRankTool**
  - **DataAgregator**
  - **ModelingTool**
- **Thesis**
  - **Sources**
  - Gogar-thesis-2014.pdf

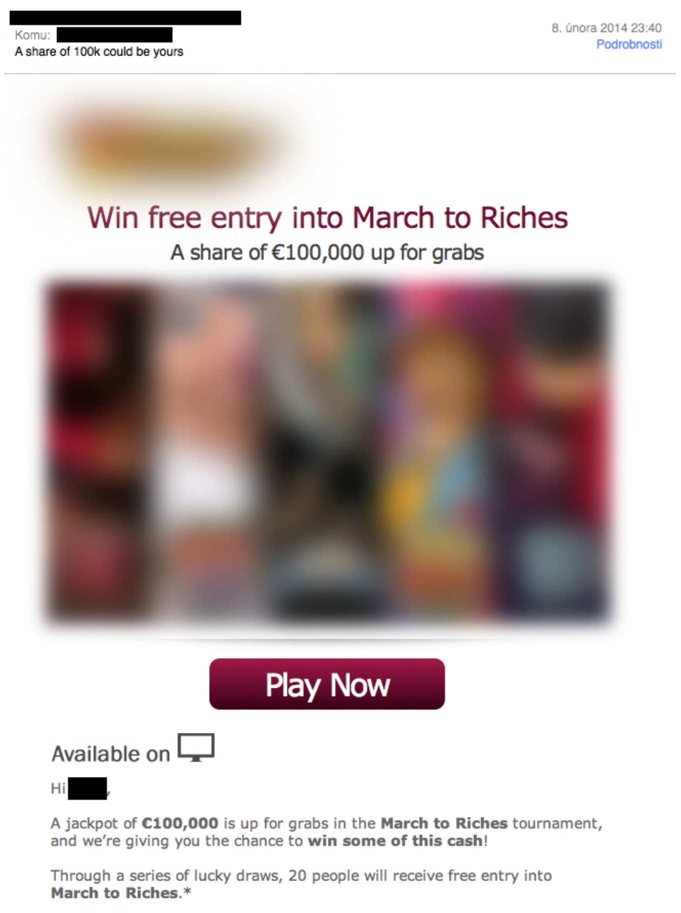
Appendix B

ProbeDataProcessor: config.xml

```
<?xml version="1.0"?>
<proccessConfiguration>

<!-- Folder names -->
<probeDataPath>ProtoFiles/04_15/</probeDataPath>
<dateFolderPath>ProtoOutput/04_15/</dateFolderPath>
<tmpFolder>tmpFolder/</tmpFolder>
<finalFolder>finalOutput/</finalFolder>

<!-- Subfolders names -->
<bulkPepcaSubfolder>bulkPepcas/</bulkPepcaSubfolder>
<statsSubfolder>stats/</statsSubfolder>
<sumphashSubfolder>sumphash/</sumphashSubfolder>
<feedbackSubfolder>feedback/</feedbackSubfolder>

<!-- Constants -->
<bulkMailThreshold>500</bulkMailThreshold>
<relevantPercentageThreshold>0.02</relevantPercentageThreshold>

</proccessConfiguration>
```

Appendix C

Figures



[redacted]

Komu: [redacted]

A share of 100k could be yours

8. února 2014 23:40

Podrobnosti

Win free entry into March to Riches

A share of €100,000 up for grabs

Play Now

Available on

Hi [redacted],

A jackpot of **€100,000** is up for grabs in the **March to Riches** tournament, and we're giving you the chance to **win some of this cash**!

Through a series of lucky draws, 20 people will receive free entry into **March to Riches**.*

Figure C.1: Content of an anomaly email - name, addresses and images were hidden due to privacy concerns