Aalto University
School of Science
Degree Programme in Computer Science and Engineering

Maksim Smirnov

Clustering and classification methods for spam analysis

Master's Thesis
Espoo, June 18, 2018

Supervisor: Professor Tuomas Aura, Aalto University
Advisor: Päivi Tynninen M.Sc. (Tech.), F-Secure

Aalto University
School of Science
Degree Programme in Computer Science and Engineering

Aalto University
School of Science

ABSTRACT OF
MASTER'S THESIS

Author:	Maksim Smirnov	
Title:	Clustering and classification methods for spam analysis	
Date:	June 18, 2018	**Pages:** vii + 73
Major:	Security and Cloud Computing	**Code:** SCI3084
Supervisor:	Professor Tuomas Aura	
Advisor:	Päivi Tynninen M.Sc. (Tech.)	

Spam emails are a major tool for criminals to distribute malware, conduct fraudulent activity, sell counterfeit products, etc. Thus, security companies are interested in researching spam. Unfortunately, due to the spammers' detection-avoidance techniques, most of the existing tools for spam analysis are not able to provide accurate information about spam campaigns. Moreover, they are not able to link together campaigns initiated by the same sender.

F-Secure, a cybersecurity company, collects vast amounts of spam for analysis. The threat intelligence collection from these messages currently involves a lot of manual work. In this thesis we apply state-of-the-art data-analysis techniques to increase the level of automation in the analysis process, thus enabling the human experts to focus on high-level information such as campaigns and actors.

The thesis discusses a novel method of spam analysis in which email messages are clustered by different characteristics and the clusters are presented as a graph. The graph representation allows the analyst to see evolving campaigns and even connections between related messages which themselves have no features in common. This makes our analysis tool more powerful than previous methods that simply cluster emails to sets.

We implemented a proof of concept version of the analysis tool to evaluate the usefulness of the approach. Experiments show that the graph representation and clustering by different features makes it possible to link together large and complex spam campaigns that were previously not detected. The tools also found evidence that different campaigns were likely to be organized by the same spammer. The results indicate that the graph-based approach is able to extract new, useful information about spam campaigns.

Keywords:	spam campaigns, mass email, malware, clustering, graph analysis, threat intelligence
Language:	English

Acknowledgements

I wish to thank my supervisor, Professor Tuomas Aura, for his insightful answers and comments which significantly helped me with this thesis and studies in general. I also want to thank my advisor at F-Secure, Päivi Tynninen, for hours of constructive discussions and helping me to define a strict scope for the research.

I also want to thank fellows at F-Secure for helping me with the research. Most particularly, I am grateful to Karmina, Robin, Alexey and Dmitry Komashinskiy.

Special gratitude to the ITMO University for the opportunity to study at the Aalto University.

Espoo, June 18, 2018

Maksim Smirnov

Abbreviations and Acronyms

ARPANET	The Advanced Research Projects Agency Network
ASCII	American Standard Code for Information Interchange
BCC	Blind Carbon Copy
BEC	Business Email Compromise
CC	Carbon Copy
CCTree	Categorical Clustering Tree
CPU	Central Processing Unit
CSV	Comma-Separated Values
CTPH	Context Triggered Piecewise Hashing
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DCC	Distributed Checksum Clearinghouse
DKIM	Domain Keys Identified Mail
DNS	Domain Name System
DNSBL	Domain Name System-based Blackhole List
DOM	Document Object Model
Email	Electronic mail
FBI	Federal Bureau of Investigation
FPTree	Frequent Pattern Tree
FWS	Folding White Spaces
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
ICANN	Internet Corporation for Assigned Names and Numbers
ID	Identifier
IEEE	Institute of Electrical and Electronics Engineers
IMAP	Internet Message Access Protocol
IP	Internet Protocol
IT	Information Technology
LCS	Longest Common Subsequence
MIME	Multipurpose Internet Mail Extensions
NDA	Non-Disclosure Agreement
NFR	Non-Functional Requirement
POP	Post Office Protocol
RBL	Real-time Blackhole List
RBLDNS	Real-time Block List Domain Name System
RFC	Request For Comments
SHA	Secure Hash Algorithm
SMTP	Simple Mail Transfer Protocol
SPF	Sender Policy Framework
URL	Uniform Resource Locator

Contents

List of Figures

Chapter 1

Introduction and Scope

Electronic mail, also known as email, is a common communication mechanism that is used by many companies and people. Companies use email to keep in touch with their clients, promote their goods and services, receive feedback from clients. People often use email for communication as the default way of reaching other people. Like any widely used technology, email can also be misused. Unsolicited email messages, also known as spam, appeared in the same time as email system appeared. The first spam message was sent on May 1, 1978, by Digital Equipment Corporation to advertise their product to several hundred users on ARPANET[43]. When the Internet opened for commercial and personal use in the 90s, spam became a widespread phenomenon. Thus with the rise of the email system, rose spam. In the first half of 2017, spam accounted for 54% of all email traffic[27].

Most of the spam emails are intended to aggressively advertise goods or websites, and they do not do much harm except wasting people's time and resources. However, besides that, spam serves an important role in cyber attacks and distribution of malicious software. According to Symantec research, it is the most frequently used mechanism to deliver malware. Also, their research states that a user is twice as likely to get infected by malware through email than to get infected through a malicious website.[27] Spam is also used for various types of fraudulent activities targeted at businesses. According to a recent analysis by FBI[38], between 2013 and 2016 years, businesses suffered losses of over $5 billion because of email fraud. These consequences of spam make it an essential attack vector for security service companies to invest their analytic resources.

Information security companies are interested in understanding trends in spam and in predicting what can happen and how to mitigate the threats. Getting a representative overview of spam is a task that requires some preparation. To do that, security companies maintain email honeypots and buy vast amounts of spam from specialized providers. Getting as many malware and phishing samples as possible and updating the detection databases on time can significantly reduce the damage caused by malware and fraud to the security company's clients.

Malicious attachments and URLs is not the only information that can be extracted from spam. Often spam discloses enough information to group different spam campaigns by a common sender. Identifying senders not only allows authorities to take actions against them but also to understand the spammers' intentions and to improve automated detection mechanisms.

One key thing about spam is that it is often not qualitative but quantita-

tive. Even if only 0.1% of users opens a spam email and follows the link, it brings one client for every 1000 messages. Moreover, spam is so cheap that one client can payoff millions of messages. According to Symantec's report, every 9th user had a malicious email sent to him during first half of 2017.[27]

Mass character of spam leads to one of the technical problems of spam analysis: massive amounts of data. F-Secure, the case company in this research, receives tens of thousands of messages every day and also purchases spam from companies which specialize in collecting it. Without special data mining tools, it is practically impossible to get a broad view of the collected data. Currently, such tools are not available on the market and companies that are interested in researching spam are forced to develop their own approaches according to their needs.

This thesis aims at making the analysis of email messages easier, providing tools and approaches to present and correlate email messages in such way that researchers can analyze large sets of email messages.

Two fields of computer science had been used to address this problem: machine learning and graph analysis. Machine learning, specifically clustering, provides ways of grouping objects that are similar according to some selected set of features. Most of the email messages differ insignificantly, therefore, allowing to combine thousands of similar messages into quite well-defined groups. This methodology can frequently provide insights about ongoing spam campaigns.

Another problem addressed in this research is data visualization. Graphs are chosen for the visualization of the clustered emails as they allows analysts to connect objects that have no direct link to each other but are connected through networks of other objects. The graph-based approach might disclose some information about the likeliness of having a common source behind both clusters.

This thesis attempts to answer the following research question:
How clustering techniques can facilitate analysis of unsolicited bulk messages?
The main research question can be divided into the following questions:

- What features can be used for clustering?
- Does clustering spam emails provide better visibility over spam campaigns?
- How can the clustering results be visualized?

The primary focus of the thesis lies on the novel method of correlating bulk email messages and presenting them to the user in such way that representation can provide additional information about the email senders that was not visible by inspecting the individual emails or even individual clusters. Feature selection is an essential part of the research as it determines the quality of the clustering, on which the presentation part relies. The developed tool is not yet

a complete product but helps to answer research questions and to evaluate the novel method of analyzing emails.

Although this thesis is focused on researching spam, the underlying analysis and presentation techniques can be applied to other types of objects.

The following section provides more details about spam, clustering algorithms and discusses relevant researches. After the background section, requirements for the tool are introduced. Next three chapters discuss a selection of the algorithms and their parameters, practical implementation of the software, and the obtained results with use cases. The discussion chapter explains the overall value of the results, how they influence the research area, and what future research directions can be taken. The conclusion chapter sums up the research.

Chapter 2

Background and related work

Email spam has existed for a long time and has a significant impact on humanity. Therefore, spam is a well-explored area with many effective approaches to spam filtering.

This chapter provides background information about the email, spam, and researches on the topic of spam clustering. The first section of the chapter describes electronic mail, explains main concepts, protocols, and formats. The second section discusses spam focusing on spam filtering techniques and techniques to avoid spam filters. The third section focuses on a comparison of commonly utilized clustering algorithms. The last section discusses researches on spam clustering and classification.

2.1 Email

Electronic mail is a method of exchanging text messages between computer users. The core concept of email is that everyone can send messages to everyone. Email protocol was designed in such way that the whole system is decentralized. This section explains how email works.

2.1.1 Email protocol

Email network consists of servers that store, transfer and process email messages. Email messages are transferred by *Simple Mail Transfer Protocol* servers. SMTP allows establishing communication and routing between different email users only using an address from an email and information from DNS servers.[22] This protocol operates within computer networks in such way that interlocutors do not have to be online simultaneously. It also allows relaying messages across different networks.[21]

Although it is possible to send and receive emails using only SMTP servers, common use case includes a mail retrieval service, such as IMAP or POP3. Mail retrieval services provide an easier way of working with incoming messages by storing and maintaining a mailbox. Services are usually run on the same server as SMTP server and allow users to fetch messages from the system. The difference between IMAP and POP3 is that IMAP stores messages on the server allowing to access mailbox from different devices, and POP3 reads messages, deleting them from the server.[26, 4]

Although most of the email clients support POP3 and IMAP, some email clients introduce their own protocols for fetching data from mail servers. Figure 2.1 depicts simplified example of an email message transfer:

1. Email client, for example, MS Outlook, connects to the known SMTP server, running on the host A. Then email client tells the address of the recipient and an email message contents to the connected SMTP server.
2. SMTP server A requests DNS server to get the IP address of the recipient's SMTP server.
3. SMTP server A connects to the SMTP server B to send information received from Outlook email client.
4. SMTP server B receives the message and hands it to the IMAP service.
5. Email client 2 connects to the IMAP service, authenticates, and fetches new messages. Now user can see these messages in his email client.

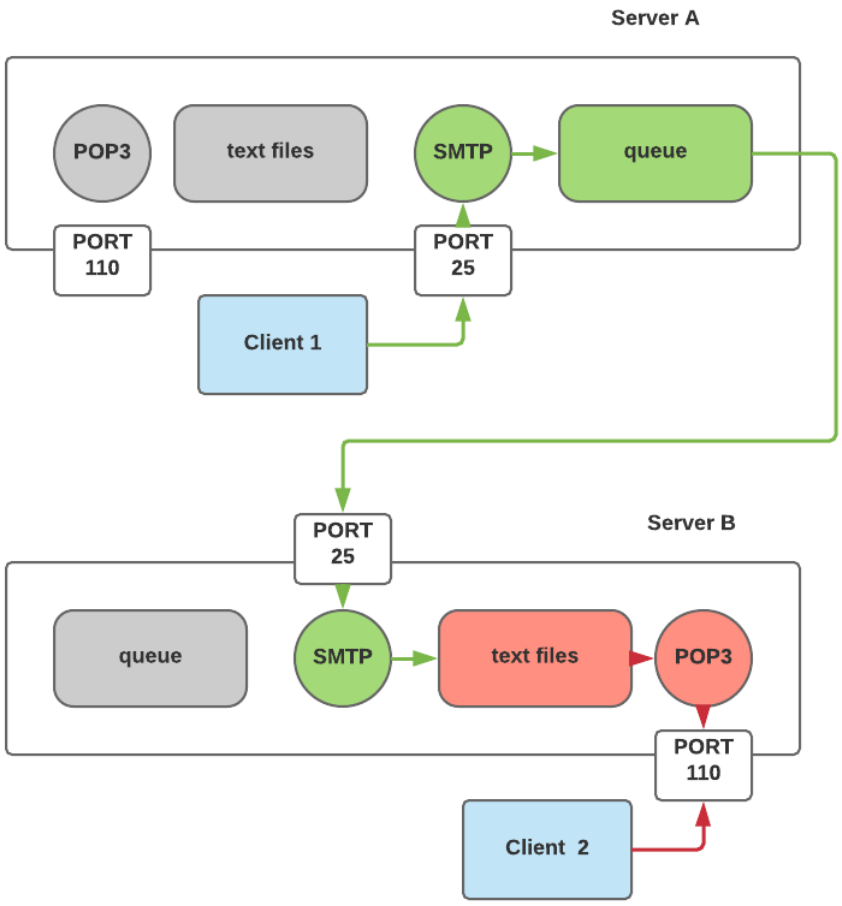

Figure 2.1: Common email transferring process

Email payload does not change during the transfer. However, there is one exception: each SMTP server has to append "Received" header storing information about the current SMTP server to the email. It is done to track the route of the message and find slow servers. Next section elaborates on message format and explains this header more thoroughly.

2.1.2 Email format

An email message is essentially a sequence of characters split into fixed-length lines. Lines are separated with "CRLF" line break.[34] Each email message consists of headers and a body. Body is optional and can be omitted.

A header is essentially a set of lines consisting of printable US-ASCII characters. It is separated from the body with an empty line. Each line of the email *must* not be longer than 998 characters and *should* not be longer than 78 characters. Each header field starts with a field name and is followed by a colon. A line is terminated by "CRLF."

The body of a message is simply lines of US-ASCII characters with several limitations: line cannot be longer than 998 characters, and "CRLF" can be used only for line breaks.[33] RFC 5322 references several other RFCs that greatly extend specification of a body, allowing it to encode non-textual message bodies[12], multi-part message bodies[13, 2], and body/headers in different character sets[25].

Header fields are logically divided into groups by a purpose. RFC 5322 defines eight groups of header fields: Origination Date Fields, Originator Fields, Destination Address Fields, Identification Fields, Informational Fields, Resent Fields, and Optional Fields.[33]

The Origination Date Field
Orig-date. Specifies time and date when the user decided to submit the email to the mailing system.

Originator Fields
From. Specifies the author or authors of the message represented as a comma-separated set of mailboxes.
Sender. Defines sender of the message. This field must be present if "From" field has more than one author. This mailbox is directly responsible for sending the message.
Reply-To. Optional field that contains a comma-separated set of email addresses for the reply.

Destination Address Fields
To. Specifies primary recipients of the message.
Carbon Copy. Specifies addresses that will also receive the email although they are not primary recipients.
Blind Carbon Copy. Specifies recipients of the message whose addresses should

not be revealed to other recipients. It is achieved by sending separate messages to each of the recipients removing the BCC field from the email.

Identification Fields
Message-Id. Contains a unique identifier of the message. Must present in each email.
In-Reply-To & References fields. Fields that are used in replies to carry ID of the original message. *References field* is filled with the content of the parent's references field, *In-Reply-To* is filled with parent's "Message-ID."

Informational Fields
Subject. A common field that describes the topic of the email message. For replies, it is common to inherit the parent's subject adding "Re: " substring at the beginning of the new subject.
Comments. Not mandatory field that may contain comments to the message body.
Keywords. Comma-separated set of important words that might be useful for the recipient.

Resent Fields
Fields *Resent-Date*, *Resent-From*, *Resent-Sender*, *Resent-To*, *Resent-Cc*, *Resent-Bcc*, and *Resent-Message-ID* are used only when original message was reintroduced to the electronic mail system by another user. These fields are purely informational and are not used for routing the message.

Trace Fields
When a message reaches an SMTP server, the server must append some trace information to that message to identify the sender host, receiver host, and the date and time according to the current host.
Received. Mandatory field to help keep track of emails routing through relays. When SMTP server processes the email, it must add a *Received* field with information about itself, for example, its IP address, the IP address of the previous SMTP server, and a current time. It helps to find slow relays and spoofed IP addresses.
Return-Path. An optional field that contains an email address to return email to. A message indicating failed delivery will be returned to this email. Sometimes it is removed by the last SMTP server on the route.

Optional Fields
A client is allowed to add any fields to an email if the field name is made up of the printable US-ASCII characters, followed by a colon, followed by any text that conforms to the unstructured syntax.[33, p. 30]

2.2 Spam

Unsolicited messages sent by email are called email spam. Spam has been a burden to the email system for a long time. According to Symantec report, amount of spam during the first half of 2017-th reached 54%, and one of every nine emails was malicious.[27] Email is still one of the prevalent vectors of malware distribution. Therefore, it is very important for security-related companies to analyze this source of information.

Basically, any unwanted email message is called spam. Thus it can be very hard to distinguish spam message from a real email message.

2.2.1 Collecting spam

Spam is easy to get without putting any effort but getting a complete representation of what is happening in the spam world is a complicated task. Five methods of harvesting spam are commonly used.[31]

Botnet malware observation
Putting a botnet malware into a sandbox and collecting all emails sent from it. This method provides the clearest spam as it comes directly from the source[31].

MX honeypot
Configuring SMTP server to accept all messages that were sent to it. The goal is to receive all emails, even those that were generated by a spammer. To let spammers know that domain has an email server, email addresses hosted on that domain must be collected by a spammer first. One good example of addresses distribution is temporary mailbox services that are used by real people to avoid usage of their real email addresses when they register somewhere. People use these temporary email services to register in other services where they do not need to receive any emails from that service. Accepting all incoming messages allows receiving significant amounts of spam that was intended for the owners of temporary addresses. Nobody will read spam on these accounts thus spammers might filter them for not being active.

Seeded honey accounts
Email addresses are created on third-party email resources and seeded by the owner of the feed. This technique provides a better diversity of domain names and, in many cases, cannot be filtered by domain names from spammers' databases. Such accounts are also not active thus will be quickly filtered by advanced spammers.

Human identified
Usually, it is a real email service with real users who can flag an incoming email

as spam manually. Messages flagged by real users are often high-quality campaigns as they manage to evade automated spam filters and get to the users. The downside is that such feeds are not scalable as to provide larger feeds they require to get more people using the email service.[31] Another downside is that different people have a different definition of spam so false negatives rate will be significant.

Domain blacklists
Domain blacklists are manually maintained feeds that can be driven by different combinations of spam source data based on the organization that maintains them[31].

2.2.2 Spam statistics

Figure 2.2 depicts rise in spam over last several years. Comparing to 2015 and 2016 years, spam rates increased in 2017. Unsolicited email messages composed more than half of the whole email volume during 2017 year.[27]

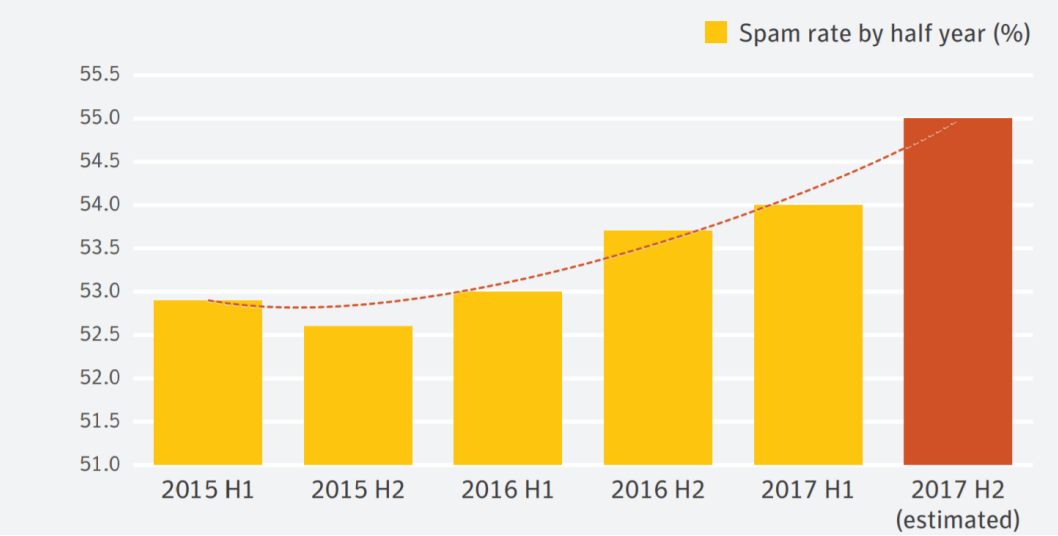

Figure 2.2: Percentage of spam

The increase in spam is not equal for all sectors. Manufacturing, mining sectors, construction, and retail trade targeted spam increased 50% in the first six months of 2017, according to Symantec spam report.[27]

Spam is mostly used to advertise goods or services, distribute malware and deliver phishing messages. Here are some key facts from the Symantec report:

- Infection of email users happens twice more often through emails than through all other infection methods
- 11% of all email users received malware in the 2017 year

- 8000 businesses each month are being targeted by Business Email Compromise (BEC) scam through spam. Targeted organizations receive on average 5.2 BEC emails per month

2.2.3 Spam filtering techniques

In the beginning, spam was very simple. Anyone could send millions of messages to anyone just knowing his or her email address. Then first spam filtering techniques started to appear. Spammers also had to improve their techniques. This section serves the purpose of explaining the situation in the spam market and what difficulties can be met during spam clustering.

SPF and DKIM
Sending an email, a spammer can put any email address into the "From" field and any domain name into the HELO command. SPF and DKIM are two common techniques that allow binding IP address to a domain name.

Sender Policy Framework (SPF) technique allows to check if sending host is allowed to send messages using some domain.[46] To enable SPF check domain owner must add a DNS record that will define which IP addresses are authorized to use the domain name as an identifier.[48]

Domain Keys Identified Mail (DKIM) is another way to perform such verification, but instead of storing authorized hosts in DNS, a public key is stored as DNS record.[5] Then to each email "DKIM-Signature" field is added. This field stores signed information about the domain that is later compared against "From" field's domain. To check that information is valid, a receiver uses the public key stored in DNS record.

Real Time Black Lists
Real-Time Black Lists (RBL) or DNS Black Lists (DNSBL) is a spam mitigation technique described in RFC 5782. It is used to drop messages that came from sources that are suspected of sending massive amounts of spam.[24] These sources are not necessary servers that were created by spammers but also can be misconfigured legit SMTP-servers abused by spammers.

RBL uses DNS infrastructure to store blocked IP addresses, and any SMTP server can access these records from standard DNS servers or specialized servers like RBLDNS.

This method also blocks SMTP servers that allow sending spam through them, for example, open SMTP relays that relay all traffic that comes in without checking the IP address correctness. The Internet is being scanned to detect such SMTP nodes to add them to the RBL before spammers start using them.

Greylisting
For each received email, SMTP server composes three pieces of data called

triplet: sender's IP address, sender's email address, and recipient's email addresses.[47] When the recipient receives a message with the unseen triplet, it rejects the message. It is an effective strategy because spammers often do not have proper infrastructure to process bounced messages and send them again. Moreover, even if they will try to resend the message in some time, they will be already filtered by RBL because, during the delay time, the spammer was sending messages to other clients.

Razor and DCC

Razor is a distributed network for checking spam body contents that allow detecting texts seen in spam without disclosing the real message.[32] Razor uses a fuzzy signature matching algorithm called Nilsimsa. Nilsimsa creates a statistical model of a message or part of a message in such way that small automatic mutations made to the message do not change the output significantly.[6]

Razor depends on user feedback about spam messages. If spam messages are not reported as spam by Razor users, Razor will give many false negatives.

Distributed Checksum Clearinghouse (DCC) is a similar way of content blacklisting, but it does not rely on user's feedback. Instead, it counts similar messages and considers all mass-sent texts as spam. So without white-listing, it will have a high rate of false-positives.[40]

Spamassassin

Spamassassin is a powerful tool for detecting spam. It is configured by adding rules where each rule has a score, and all rules are run for every received email.[11] Scores of these rules are summed up and compared to the threshold value. If the sum is higher than a threshold value, then the message is considered spam. Users can add their own rules which are in the form of regular expression. A rule can be run against all parts of the message, not only body or header. Spamassassin also allows adding more sophisticated rules and plugins for all kinds of spam checks such as statistical analysis tests, RBL tests, Bayesian filtering, and domain name verification.

Bayesian filtering

Bayesian spam filtering is based on a statistical theorem that estimates a probability of an event. It can be configured in different ways, but two most common approaches are to detect words and letter sequences.[1]

Words analysis is rather simple. Bayes filter has to be trained on spam samples and legit messages. Using information from the training, it is possible to see which words are often used in spam emails and which are not. Bayesian filter adapts to new spammers techniques very fast, for example, when spammers started using the word "f-r-e-e" instead of "free" to avoid Spamassassin word filters, Bayesian filter managed to learn that "f-r-e-e" is a good indication for spam emails because it is mostly used in spam.

The Bayesian spam filter is also able to detect randomly generated texts

that spammers often add to their messages to avoid having completely similar messages. It is possible because the probability of having a pair of letters together is not same for all pairs and generated text will not have this distribution of probability. One example of software using the Bayesian filter is DSPAM.[17]

2.2.4 Filters evasion techniques

Spamming techniques are aimed against spam filtering techniques. Filtering was the reason why filtering techniques evolved, and evolution of filtering techniques lead to improvement of spamming techniques.

One interesting thing to keep in mind is that email servers configured differently thus making even obsolete spamming methods to reach some percentage of the recipients. So the existence of anti-spamming techniques that is hard to avoid does not mean that all email servers will apply the new technique and spammers will have to find some new technique immediately. The process of applying new methods takes time.

Email forgery
Changing sender's address or domain in an email message is called email forgery or email spoofing. It is possible because email protocols do not have any authentication mechanisms in them by design.

To send an email using SMTP, a client has to provide two core pieces of information that compose "envelope" part of the email message. These pieces are:

- RCPT TO - specifies email addresses of recipients
- MAIL FROM - specifies the sender that will be presented to the recipient in the "Return-path" field of the email message

If SMTP server accepts the envelope, the client can send the DATA part of the email that includes "From" and "Sender" fields that will be shown to the sender. As a result, a receiver will see the wrong email and may respond to the wrong mailbox. Another consequence is that the spammers can exploit Non-Delivery reports by forging "Reply-to" field in such way, that Non-Delivery report will be sent to the wrong mailbox. Such wrongly routed emails are called *backscatters*".

In terms of this research, it means that often sender fields cannot be trusted if they are not verified with DKIM or SPF.

Hiding text in email body
One way of confusing spam filters is adding fake text to the message body. Many flavors of such technique, but the concept stays the same: make spam filters check a text, that is not part of real payload intended to be shown to a

recipient. The simplest way to do that is to add some generated text to the body after the payload text and to put padding after the payload. In most cases, recipients will not see the appended fake text if they do not scroll the message to its end.[40] Adding text from Wikipedia articles can also confuse Bayesian filters as suspicious part of the text will be smaller compared to the amount of legit text.

Another way is to add generated text as HTML at the beginning of the message, making it the same color as background thus making text invisible to a human. Some spam filters will parse generated text as legit body text, and it can lead to a false negative result.

Generating subject and body text

Message subject line and body text are the way for the spammer to say something to a receiver of a message. Many anti-spam techniques have been developed to track and provide real-time information about spam campaigns based on subject lines and body messages. For example, Razor, discussed above, generates hashes of text parts and checks if they had been seen in other emails submitted before. This section discusses common ways of evading text-based blocking techniques.

Writing millions of similar subjects and texts would be the perfect solution for the spammers, but it requires an enormous amount of resources that will not pay-off. Spam has to be cheap to be profitable. Thus an automated generation of texts is much more preferable for the spammers.

One of the techniques for text generation is called *synonym swapping*. Synonyms swapping is a method allowing to generate sets of sentences that have the same meaning but use different words from a pre-defined set of tokens. It is done by splitting the sentence into several parts and picking a set of possible interchangeable words or collocations.[40] Splitting message on more parts and creating bigger sets of words for each sentence allows to increase the number of possible combinations significantly.

A table is the simplest form of the source structure. Another way of defining the pattern is to create a finite state machine where each node is a set of tokens. Example of such state machine can be seen in Appendix 1. By going from root node through the graph and picking random token from each node, a program can generate different sentences that will have the same meaning. Table 1 shows examples of generated texts.

Generating URLs

URL addresses in emails serve two purposes: track which recipients were interested enough to click the link and to provide some easier way for a user to get to the provided service or get exposed to malicious content. Therefore links cannot be obfuscated in such way that they stop working. It makes parsing and analyzing them an easier task for spam filters compared to the text-based filters.

URL consists of three parts:

1. Scheme - identifies the protocol used to fetch the content.

   Spammers often use HTTP scheme because it does not require to get HTTPS certificate for the server. Many legitimate websites often use HTTP as well.

2. Hostname - points to a machine running the web service. It is also referred as a domain name. To get a domain name, web service owner has to register it through a domain name registrar at the organization called ICANN. The usual cost of owning a domain name for a year varies from $10 to $35.

   Hostnames cost too much to make them unique for each email message, but sub-domains can be generated for free. DNS supports wildcard DNS records that will redirect all requests to sub-domains to a single server thus making the generation of sub-domains an easy option of randomizing the domain name. Comparing domain names between different messages can be tricky due to sub-domain generation.

   Some domain owners provide their sub-domains to other people, so if two emails have the same domain name but different sub-domains it does not necessarily mean that the same person sent them. Fortunately, it is not common in spam for some reason.

3. Path and Query. It includes resource location and query for that resource. It is possible to make it random for each recipient, however, due to the implementation of the software used for displaying the content, the pattern of this part of URL often stays the same.

Apart from generating sub-domains and randomizing the query format, spammers often use URL shortening. Several services are commonly used such as "bit.ly" and "goo.gl." These services allow creating links that will redirect users to any other link. Use of such services can also thwart using URLs inside message bodies as the domain will not be seen without following the shortened link.

2.3 Clustering algorithms

Clustering is an analysis method to discover groupings in a data set.[20] This section provides background information for common clustering algorithms that had been considered for the research and discusses their advantages and disadvantages.

2.3.1 K-Means

K-Means clusters data by randomly putting points, or centroids, across the dataset and moving them each iteration closer to the closest cluster's center. After converging, points are set again. After N iterations, the best result is selected according to the evaluation function. The objective of the algorithm is to find:

$$\underset{s}{argmin} \sum_{i=1}^{k} \sum_{x \in S_i} ||x - \mu_i||^2 = \underset{s}{argmin} \sum_{i=1}^{k} |S_i| Var S_i \tag{2.1}$$

Where x - data points, k - the number of result clusters, S - sets of points forming clusters, μ - mean of a set.

K-Means requires two arguments: number of centroids and number of iterations.

Disadvantages of K-Means:[36]

- Clusters must be convex
- Requires clusters to have a center of mass and for each point to have a set of coordinates. Needs vectorization of the data
- In rare cases, the algorithm can give a not optimal result because it involves random selection of centroids

2.3.2 DBSCAN

Density-based spatial clustering of applications with noise (DBSCAN) is a density-based clustering algorithm that clusters together closely lying points. DBSCAN introduces the concept of core points, composing dense clusters. Result clusters also include points that are not core points but are neighbors of at least one of core points. Outliers do not have any core point neighbors. Figure 2.3 depicts example cluster. **A** point is a core point, **B** and **C** are parts of the cluster but not core points, and N point is an outlier.

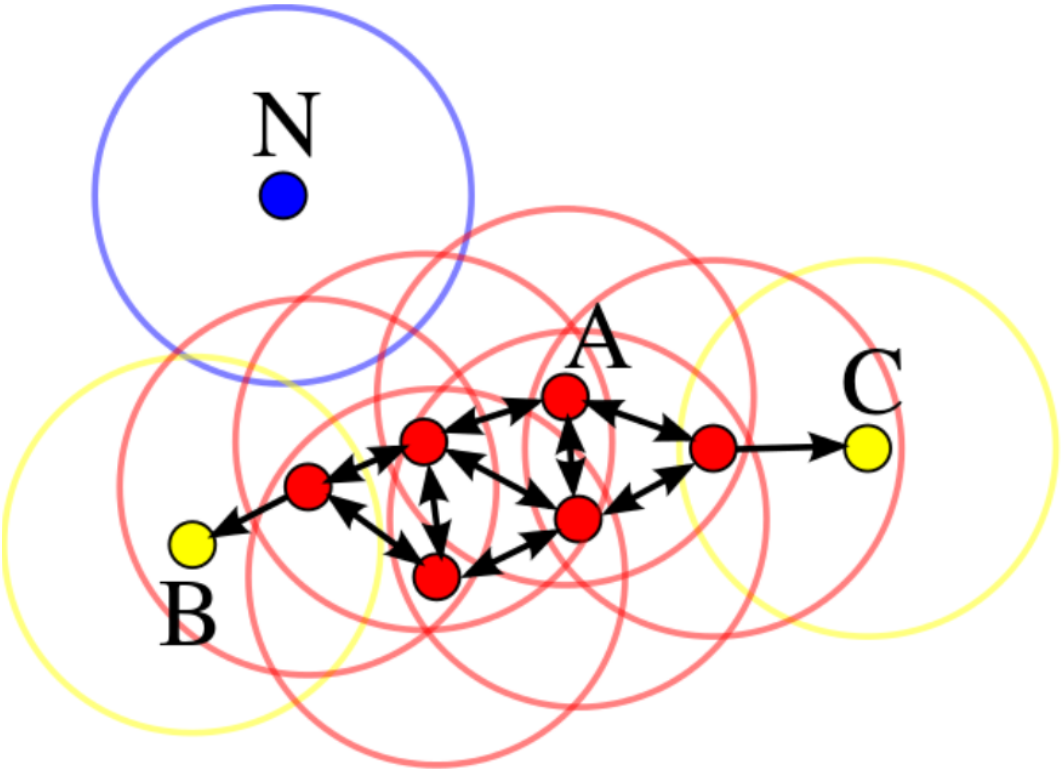

Figure 2.3: DBSCAN example

DBSCAN requires two parameters:

- ε or EPS - maximum distance between neighbours
- MinPts or min_samples - minimum neighbor samples to consider the point to be a core point

Disadvantages:

- Not entirely deterministic
  Non-core points that are reachable from different clusters can be assigned randomly depending on the order of data processing.
- Scalability issues
  Many implementations of DBSCAN are not well optimized. Also, the

algorithm has average runtime complexity of $O(n \log n)$ and worst-case complexity of $O(n^2)$.[19] When working with not vectorized data, implementations often require pre-built distance matrix that requires minimum $O(n^2/2)$ for symmetrical distances.

2.3.3 Hierarchical clustering

Hierarchical clustering is a set of clustering algorithms, the main concept of which is to build new clusters by separating existing ones or delete clusters by merging them. Hierarchy is often represented by a tree where leaves of some node are components of the same cluster. Figure 2.4 depicts an example tree. Agglomerate clustering is a hierarchical clustering that uses a bottom-up approach. In the beginning, all nodes are considered clusters of one node. Each iteration, clusters are being merged into bigger clusters.

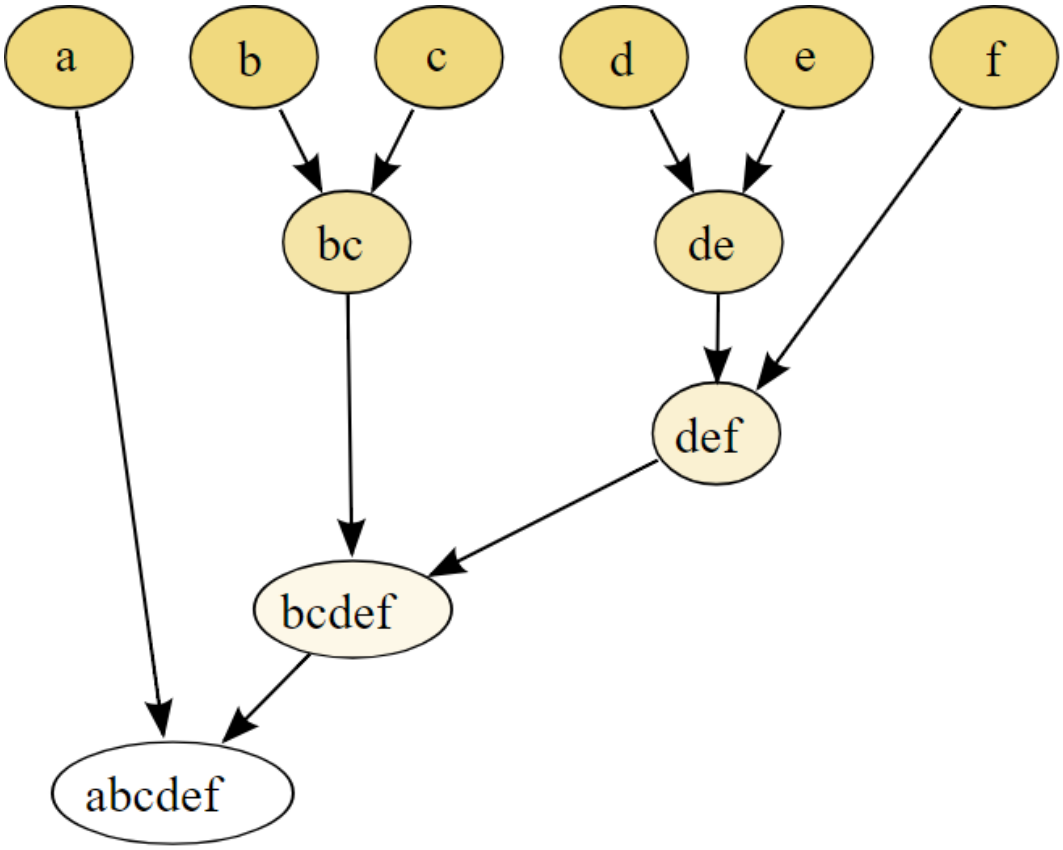

Figure 2.4: Tree representation of a hierarchical clustering

Disadvantages:

- Rich gets richer. Big clusters tend to grow faster

- Computational and memory complex. Has $O(n^3)$ computational complexity and $O(n^2)$ memory complexity [7]

2.4 Related work

This chapter is dedicated to researches on the subject of "Email Clustering." Each section describes one approach to the problem and overviews existing papers, evaluating results of each approach.

2.4.1 Hierarchical approaches

"Spam campaign detection, analysis, and investigation"[9] research proposes an approach based on frequent pattern tree (FPTree) algorithm. Authors consider 6 different features.

Content Type. Common content types are "text/plain", "application/octet-stream", "multipart/mixed", "text/html", etc.

Attachment Names. Names of the email attachments as a string.

Character Set. Roughly represents the spam language. For example, "windows-1251" usually used for Cyrillic alphabets.

Subject. Email subject as a string.

URL Tokens. Each URL is split into features such as hostname, path, and parameters.

Email Layout. Each email body is converted into a sequence of characters such as "T" (Text), "U" (URL), and "N" (newline).

FPTree is a tree where all nodes represent features and their frequency.[15] Each path from a root to a leaf represents an email. All children of the node X are extracted from emails that have all features from root to X.

Figure 2.5 depicts an example of how FPTree is used to work with emails. As can be seen from email subjects, all emails share the common topic.

Figure 2.5: FPTree example

Evaluation of the clustering is also present in the paper. Authors used three methods of evaluation that rely on the text similarities: W-shingling, Content Triggered Piecewise Hashing, and Locality-Sensitive Hashing. Figure 2.6 shows results of the evaluation. The vertical axis represents the score of each metric and horizontal axis represent campaigns sorted by their similarity scores. As we can see from evaluation, about 30% of campaigns have a quite low similarity score.

This method relies on email patterns and does not rely on actual texts thus it will cluster campaigns that have completely different bodies but constructed using the same pattern.

Also, spammers can evade such technique by generating invisible parts of HTML that will change the pattern significantly.

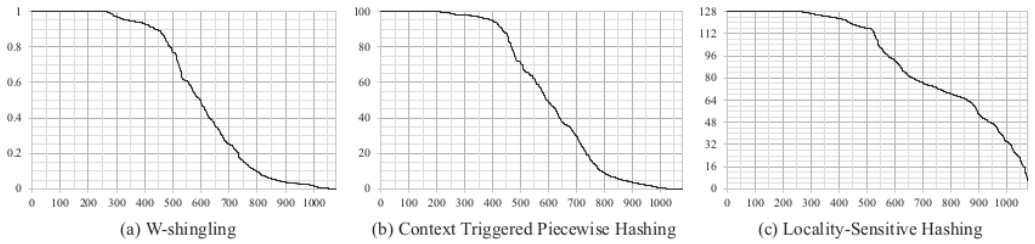

Figure 2.6: FPTree campaigns similarity evaluation

"Fast and Effective Clustering of Spam Emails based on Structural Similarity"[37] yet another tree-related approach for clustering emails. It introduces

Categorical Clustering Tree (CCTree) algorithm that utilizes 21 features extracted from each email message. Features are mostly related to the email contents, such as number of links, number of images, number of attachments, body and subject language, email message size, number of words in a body, types of attachments, and some other similar features. Full set of features is shown in table 2.1.

Attribute	**Description**
RecipientNumber	Number of recipients addresses.
NumberOfLinks	Total links in email text.
NumberOfIPBasedLinks	Links shown as an IP address.
NumberOfMismatchingLinks	Links with a text different from the real link.
NumberOfDomainsInLinks	Number of domains in links.
AvgDotsPerLink	Average number of dots in link in text.
NumberOfLinksWithAt	Number of links containing "@".
NumberOfLinksWithHex	Number of links containing hex chars.
SubjectLanguage	Language of the subject.
NumberOfNonAsciiLinks	Number of links with non-ASCII chars.
IsHtml	True if the mail contains html tags.
EmailSize	The email size, including attachments.
Language	Email language.
AttachmentNumber	Number of attachments.
AttachmentSize	Total size of email attachments.
AttachmentType	File type of the biggest attachment.
WordslnSubject	Number of words in subject.
CharslnSubject	Number of chars in subject.
ReOrFwdlnSubject	True if subject contains "Re" or "Fwd".
NonAsciiCharslnSubject	Number of non ASCII chars in subject.
ImagesNumber	Number of images in the email text.

Table 2.1: Features extracted from each email

The main concept is the same as in FPTree approach, but all features are categorical. To convert numerical values into categories, ChiMerge discretization method is used.

This paper is valuable because of the features that were utilized as they mostly relate to the structure of the document and not to exact resources. Also, the big number of features is highlighted as one of the advantages of this work.

"Mining Spam Email to Identify Common Origins for Forensic Application"[45] is yet another hierarchical approach to work with spam. Research consists of two parts: hierarchical clustering and fixing false positives.

For hierarchical clustering, two features have been used: subject lines and domain names extracted from URLs. The first iteration of hierarchical clus-

tering combines all messages that have the same subject. The second iteration merges clusters that have the same URLs in bodies with a single-linkage clustering algorithm. Single-linkage clustering algorithm clusters two nodes together if they have at least one connection between the elements of different clusters. After the second iteration, researchers noticed that one cluster has 67% of the emails with URLs.

To resolve false positives, researchers applied graph approach. They created a vertex for each domain in the biggest cluster and an edge for each common subject between two domains. This approach allows finding weakly connected subgraphs that are connected only by several edges and separating them to split the cluster into smaller clusters. The main idea is that campaigns that accidentally happen to have the same subject will be weakly connected because the probability of sending messages with the same subjects is low. By applying this technique, researchers managed to reduce the number of clusters and make them more related.

This paper partially inspired approach applied in this research as it also utilizes graphs to correlate different features together. However, the approach suggested in this thesis differs significantly from what was proposed in "Mining Spam Email to Identify Common Origins for Forensic Application".

2.4.2 Text based clustering approach

"Clustering Spam Campaigns with Fuzzy Hashing" [3] uses fuzzy hashing to cluster emails into campaigns. Fuzzy hashing is used because spammers often slightly change email messages to evade spam filters. Thus researchers propose approach depicted in figure 2.7 that allows to cluster new messages into existing clusters and create new clusters if a message cannot be assigned to an existing cluster.

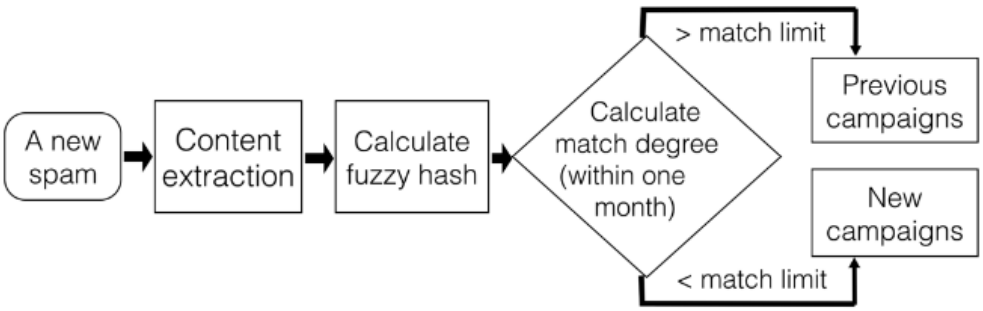

Figure 2.7: Fuzzy hashing clustering process

Context triggered piecewise hashing (CTPH) is used to hash each message. To compare two hashes, researchers used weighted edit distance.

Each new message is compared against all existing clusters using above metric and assigned to a cluster with the highest similarity rate if the simi-

larity is bigger than the constant threshold. If the similarity is less than the threshold, a new cluster is created.

The computation cost is low because the method relies on a simple hash, although it still requires pair-wise comparison among two hashes to obtain a similarity score.[3] This paper does not provide research of how adding random invisible text changes effectiveness of the algorithm.

After empirically selecting the threshold and clustering 550000 samples, researchers got 118760 clusters. Such big number of clusters researchers explain by the diversity of the dataset. It also might be explained by text obfuscation methods applied by spammers.

Chapter 3

Tool requirements

The purpose of the developed software is to show how the suggested approach can be utilized in a real environment on real data. Although the tool is just a proof of concept, usability is an important part of it. If researchers have problems using the tool, it can be hard to evaluate its advantages for them. Thus, before creating a user interface, few questions had been asked, and few core requirements developed. This chapter describes requirements suggested by the researchers.

3.1 Functional requirements

Functional requirements determine what system's behaviour: what system supposed to do, what functionality supposed to implement.

FR1: Clustering
Emails sharing common features must be grouped together.

FR2: Searching
A user must be able to work with the results of clustering and be able to search the data by different attributes.

3.2 Non-functional requirements

Non-functional requirements describe properties of the system. Usability and performance are often part of non-functional requirements[39].

NFR1: Performance
The system must be able to provide browsing functionality in real time.

NFR2: Visualization
Data must be presented through a graphical interface.

NFR3: Learning curve
The learning curve for the instrument must be reasonable. A researcher should be able to work with the data with less than one-day training.

Chapter 4

Clustering algorithms and visualization

In unprocessed raw format, email messages are text documents stored in a database or a file system. To work with the data, researchers often parse emails and put valuable features into some databases or search engines like Elastic Search. Search engines allow to easily search through the data and build simple clusters using parsed features. However, loading all the data into the database costs a significant amount of money and loading more than a year worth of data, even omitting attachments and message bodies, might be too expensive for a company, especially because it still lacks methods for presenting the data in a way that researcher will not be overwhelmed with the it.

Researchers' main aim is to get information about running campaigns. Spammers often share features inside a single campaign and sometimes even between different campaigns.[3] Such features can be meaningful parts of email bodies, message templates, used software, IP addresses, email subjects, domain names within the message, and other characteristics of an email message that is hard to change randomly. These features can be used for clustering.

Some campaigns last for quite a long time[3] and undergo different changes to their features to avoid spam filters.[29] Spammers often try to obfuscate some features in such way that it is practically impossible to cluster them by an obfuscated feature. Therefore clustering should be done in such way that email messages will be connected even if one of the features is obfuscated or changed.

One of the solutions might be clustering by different features and searching for overlapping clusters. Thus if features change during the campaign not simultaneously, messages will still be connected with other clusters.

This chapter discusses the selection of features and clustering algorithms. Visualization of the clusters is being discussed at the end of the chapter.

4.1 Feature selection for clustering

This section enumerates features extracted from the messages. Each feature's significance is discussed. For some of the valuable features, clustering algorithms are implemented and tested on the dataset to select optimal parameters. These parameters are used later in the implementation chapter. Figure 4.1 depicts all considered approaches as a mind map.

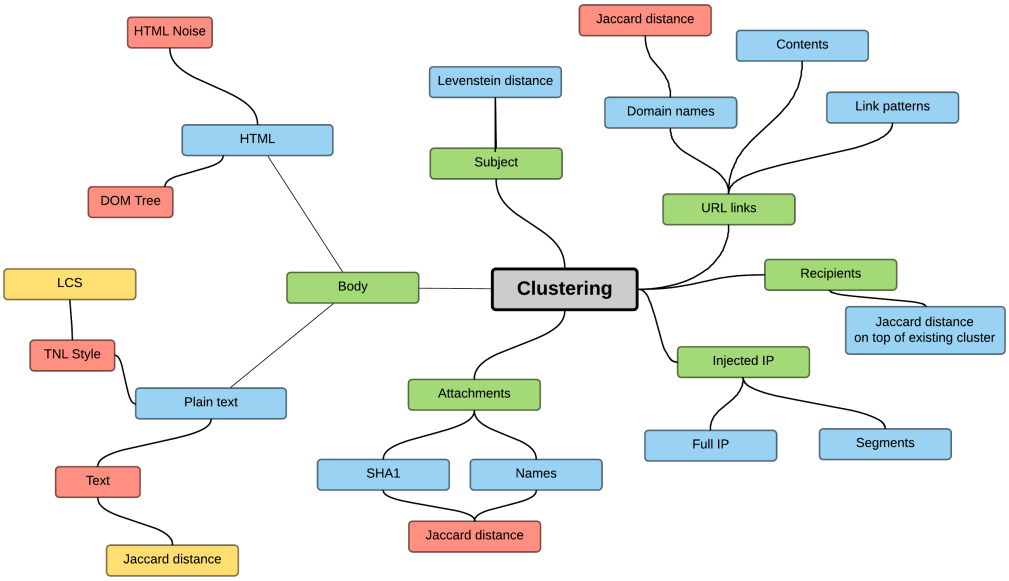

Figure 4.1: Considered clustering features and approaches

4.1.1 Subjects

A subject line is often present in spam messages. It is the first thing that receiver sees in the inbox, so it has to be human-readable and enchanting for a recipient to open the email. A spammer can change a subject line during a campaign, but often meaning stays the same. The technique called synonym swapping, described in 2.2.4 section, allows to generate meaningful subjects and bodies with a low similarity between different email messages. However, it is rarely used because it demands a human effort to compose such table. On practice, spam subjects from one campaign look the same, and most of the messages can be clustered only according to the similarity of subject lines.
Clustering. Clustering by subjects is done in 4 steps:

1. Parse subjects from email messages. This includes decoding the subject and converting it into readable Unicode text.

2. Cut subject line to 78 characters. Cutting subject lines not only improves performance but also is related to RFC standard. RFC states that fields should not be longer than 78 characters[33], therefore, longer subjects are often cut by RFC-compliant email clients.

3. Create n^2 distance matrix with all distances between all messages.

4. Run DBSCAN clustering on calculated distance matrix with learned parameters.

Levenshtein distance is used as similarity metric for subject lines because changes made to campaigns are often insignificant and limited to adding or changing words or punctuation.

DBSCAN is a clustering algorithm that works by separating low-density areas from high-density areas. It had been selected for clustering because of the next properties:

- No vectorization needed

  Selected distance metric does not allow to generate vectors for the messages, but it can be used to generate distance matrices for the whole set.

- No cluster number estimation needed

  The algorithm must work with any amount of clusters and data without the need to revise data each time to determine parameters for the algorithm. DBSCAN allows setting constant density which is related to the similarity of subject lines.

- Shape of clusters can be non-convex

  If synonym swapping, discussed in 2.2.4, is used DBSCAN will be able to cluster them by finding subjects that have only a few different words building a bridge between completely different subjects but created using same synonyms table.

Estimation of Clustering Parameters. DBSCAN requires two parameters to define density of clusters: *EPS* - maximum distance between neighbours and *MinPts* - minimum neighbours for sample to be considered a core sample.

Estimation of the parameters is a simple optimization task. Iterating parameters through possible values is one of the existing approaches to determine parameters of clustering algorithms.[10] Both parameters were iterated separately, and the optimal value was determined. With MinPts value equal to 2, EPS value was iterated from 0 to 1 with 0.01 step. After determining the optimal EPS, MinPts was iterated from 1 to 10, keeping EPS optimal.

Test set had to be created to evaluate the clustering. 600 samples were labeled manually into 56 clusters. Manual labeling was performed according to many different features, not only subject lines. In some cases, subject lines were completely different for samples advertising same goods, and in some cases, emails with same subject lines were clearly from different campaigns.

For each step next six characteristics were collected: number of false positives (FP), number of false negatives (FN), number of true positives (TP), number of false negatives (FN), number of clusters, and number of outliers. For clustering, terms "positive" and "negative" are applied to each possible pair of samples. For example, if two samples that are supposed to be in the same cluster are put in different clusters, it can be called false negative. If they are put in the same cluster - true positive.

Figure 4.2 depicts results of clustering and evaluation according to the labeled set. As can be seen from the figure, TP and TN rise rapidly until EPS equal to 0.37. FP rate starts to rise from EPS 0.35. Interval of EPS from 0.26 to 0.34 gives a good amount of TP and TN, meanwhile having 0 FP and comparably low FN. Table 4.1 depicts confusion matrix for EPS equal 0.34.

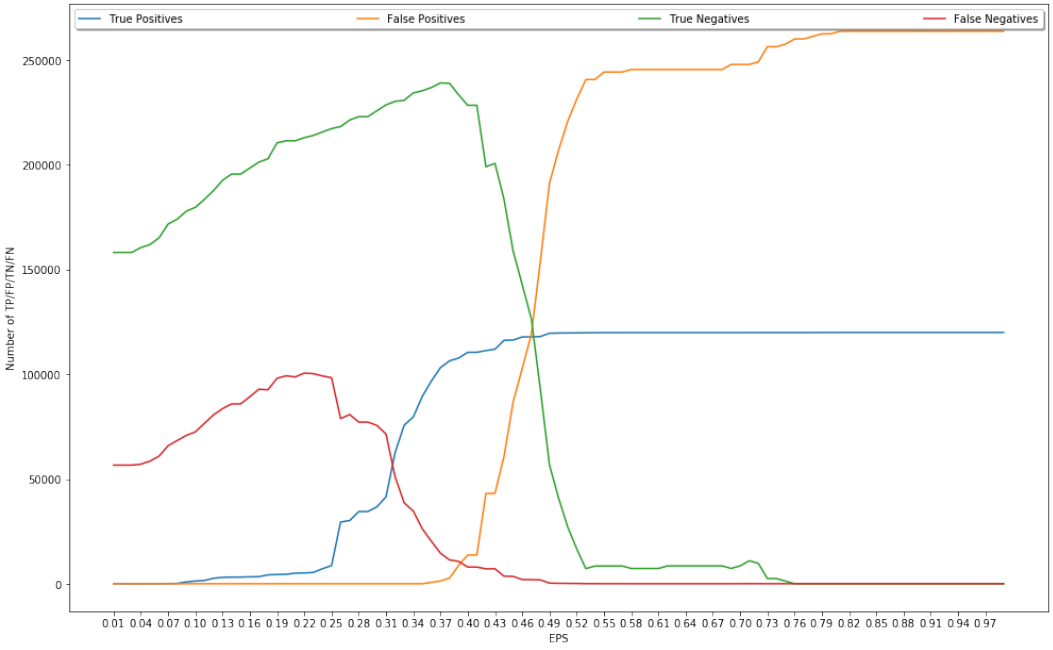

Figure 4.2: TP-FP EPS evaluation for subjects clustering. MinPts = 2

	Predicted: No	Predicted: Yes
Actual: No	TN=263792	FP=0
Actual: Yes	FN=40372	TP=79616

Table 4.1: Confusion matrix for EPS=0.34. Subjects clustering

It is common to separate labeled set on a training set and a test set to prevent overfitting of machine learning algorithms. In this research, parameters for clustering are being selected manually, so correctness of the clustering parameters rely only on how representative labeled dataset is.

To show that labeled dataset is sufficient to select parameters, all graphs are also created for a dataset having random 50% of the full labeled dataset. Figure 4.3 depicts evaluation of EPS for 50% of the whole labeled set. It can be seen that results are the same as for 100% set. Similarity of FP and TP rates for different sizes of labeled datasets shows that increasing size of labeled dataset will not provide more precise results for threshold selection.

Different sizes of test datasets were tested for threshold selection for other features too, but it did not change the selection of the threshold. Therefore graphs for 50% datasets are omitted in future parameters estimations.

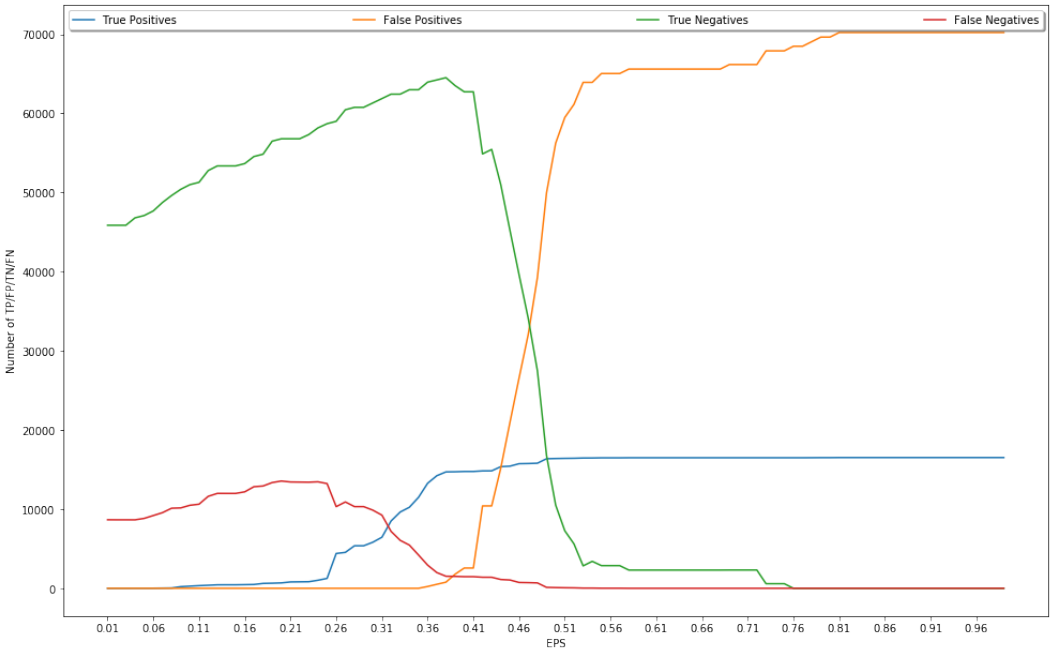

Figure 4.3: TP-FP EPS evaluation for subjects clustering. MinPts = 2. 50% of the labeled set

Figure 4.4 shows how the number of outliers and clusters change with the change of MinPts. It can be seen that with MinPts = 1 many more outliers were clustered while keeping rates of TP and FP, depicted in figure 4.5, almost the same. However, further inspection showed that it happened because with such low MinPts clusters were created with only one email in them. After removing 650 clusters that consisted only from one email, number of clusters happened to be same as with MinPts = 2.

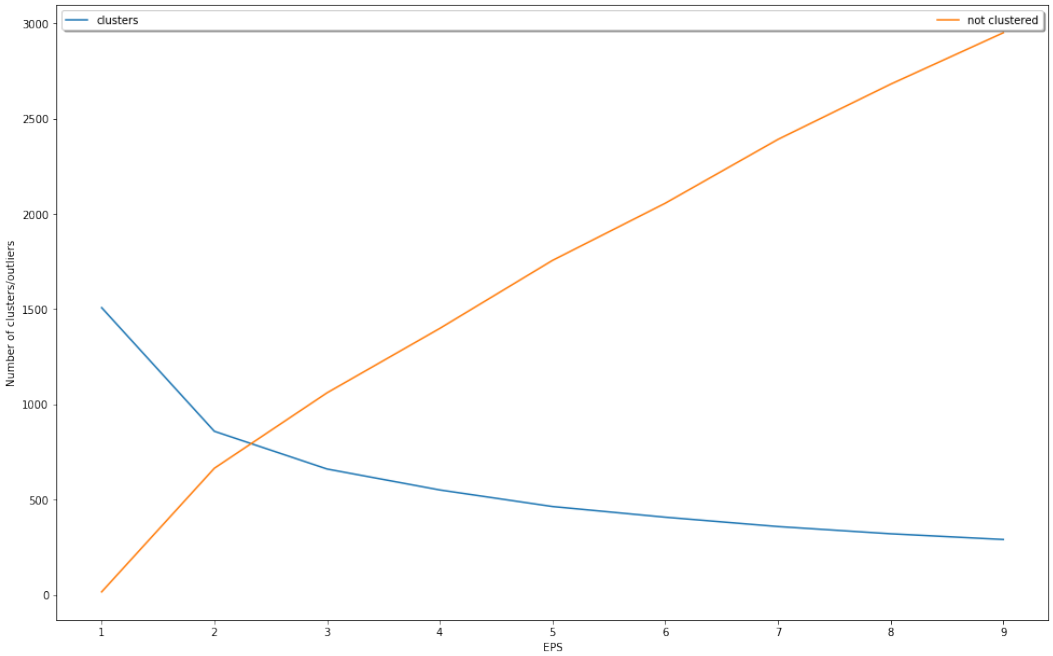

Figure 4.4: Clusters and outliers for changing MinPts for subjects clustering. EPS = 0.34

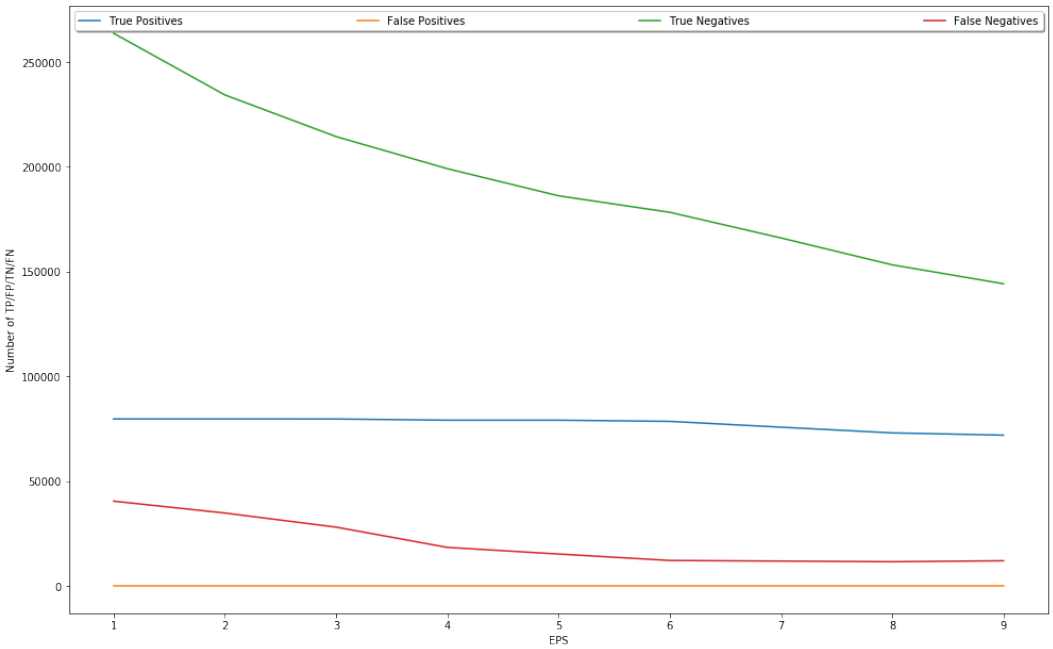

Figure 4.5: TP-FP for changing MinPts for subjects clustering. EPS = 0.34

MinPts was checked for all features, and it behaved the same as here. Thus graphs are omitted. MinPts for future clustering parameters evaluation is set to two.

4.1.2 Bodies

As discussed in 2.1.2 section, body is optional. However, most of the spam samples have it. Body is viewed by people who got interested in the email by reading the subject of the email. Thus it has to be simple and engaging enough to make the user perform some simple actions such as clicking on the link, downloading and running the attachment or responding to the email.

Bodies are much easier to obfuscate due to the variety of formats that modern email clients support. Common methods to obfuscate body are discussed in 2.2.4 section. Body is often present either in HTML or plaintext format.

HTML itself can also be considered as a feature for clustering. Two features related to the HTML can be retrieved: HTML noise and DOM tree. Rendered text can also be a feature for clustering.

HTML noise

HTML noise had been already utilized to track and cluster web spam in "Tracking Web Spam with Hidden Style Similarity"[44], but it had never been applied to spam.

Noise retrieval allows recovering details of messages, such as spaces and symbols, that are produced by the software used to generate the HTML template for the campaign. If two different campaigns have same templates noise, it might indicate that the same spammer is responsible for different campaigns. However, if different actors use the same software or standards-compliant software, the approach will give many false positives as many samples will have similar HTML noise.

Although it still can be used to find some valuable insights in some cases, it had been considered to be not strong enough and had been postponed for the future.

DOM Tree

Email messages quite often have some unique structure, although it is possible to obfuscate DOM structure for each email, often template is not changed or changed only slightly. Thus different tags and their positions might compose a feature that stays persistent during the campaign.

One of the advantages of this feature is that it might disclose campaigns with completely different content but that have the same HTML template to generate messages. Thus it can show campaigns created by the same sender. However, this feature also has two significant disadvantages:

- Spammers can easily change the template or start using common templates to mimic legitimate mailing software
- Not all spam is HTML, so it is not a universal feature

This feature is quite promising, and it might be used in future implementations, but now it had not been selected for the implementation.

Text
The text is present in almost every email message, although sometimes it comes in HTML layout. Spammers are interested in a client being able to read the text and having text to look not suspicious. Therefore, text can often be extracted from an HTML using deterministic approaches similar to those used in HTML rendering. Three properties of the body text has been considered:

Text meaning
What is advertised or discussed in the text.

Text layout
Representing text as a complex of text fragments, URLs, and line breaks. The text itself is excluded. This approach had been implemented and tested in "Spam campaign detection, analysis, and investigation" research.[9]

Characters or words sequence
Perceiving body text as a sequence of characters or words that can be changed during the campaign only slightly by replacing words or changing symbols in them.

Body text represented as a set of identifiers has been selected for the implementation as it is the most obvious and straightforward way to cluster similar generated text.

Many distance functions exist to compare texts[14], here some of the commonly used ones:

- Jaccard index based distance

  Measuring distance between sets of words using Jaccard index. This approach does not take into account words order. Complexity is $O(n^2)$ for n size sets [8].

- Levenshtein distance

  It is a memory and time intensive distance measurement. It has memory and time complexity of $O(m * n)$ [23].

- LCS (Longest Common Substring) length

  LCS will find common substrings, but it also should take into account length of the overall message. It has no tolerance for small changes in the text.

Jaccard index was chosen because as it is working with words and size of sets will be much less than when working with characters in Levenshtein distance.

Clustering is done in next steps:

1. Parse HTML and extract first 20 words from the parsed text.

   Manual inspection of spam messages showed that all, or almost all, spam messages try to provide the main idea and important content in the first 20 words. It might be because spammers have to keep a recipient interested from the first moment he opens the email. Otherwise, the message will be sent to spam.

2. Filter everything except words and create sets from the sentences.

3. Calculate distance matrix from every set to every set using Jaccard index based distance.

   Jaccard index is a similarity measure. To convert it to distance metric it must be subtracted from 1.

4. Cluster emails using DBSCAN with distance matrix.

Estimation of Clustering Parameters. The process is very similar to Levenshtein-DBSCAN clustering showed before. Figure 4.6 depicts evaluation of TP and FP for changing EPS. False positives go shallowly and start going up very steep from value 0.71. True negatives reach the peak at EPS 0.67 and True positives going up very fast from 0.48 to 0.72. According to the graph, the optimal value is 0.70 as it provides highest true positives rates with almost no false negatives.

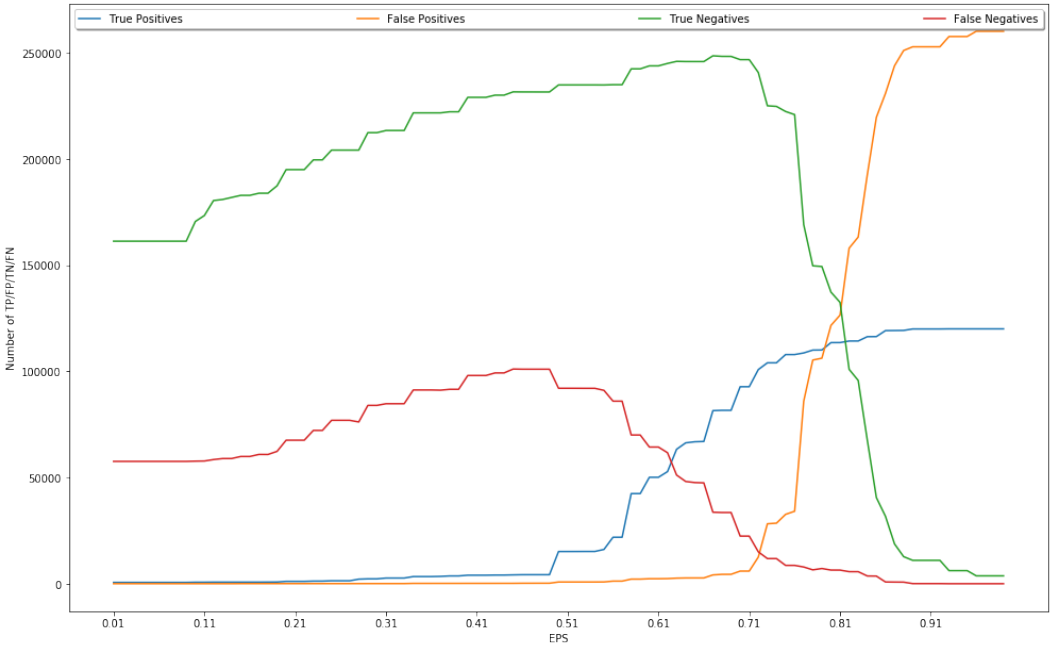

Figure 4.6: TP-FP EPS evaluation for bodies clustering. MinPts = 2

Figure 4.7 depicts how the number of clusters and outliers changed with the change of EPS. It can be noticed that starting from EPS 0.74 clusters start to decrease faster than outliers and eventually intersect outliers graph. It means that algorithm starts to merge clusters instead of clustering outliers.

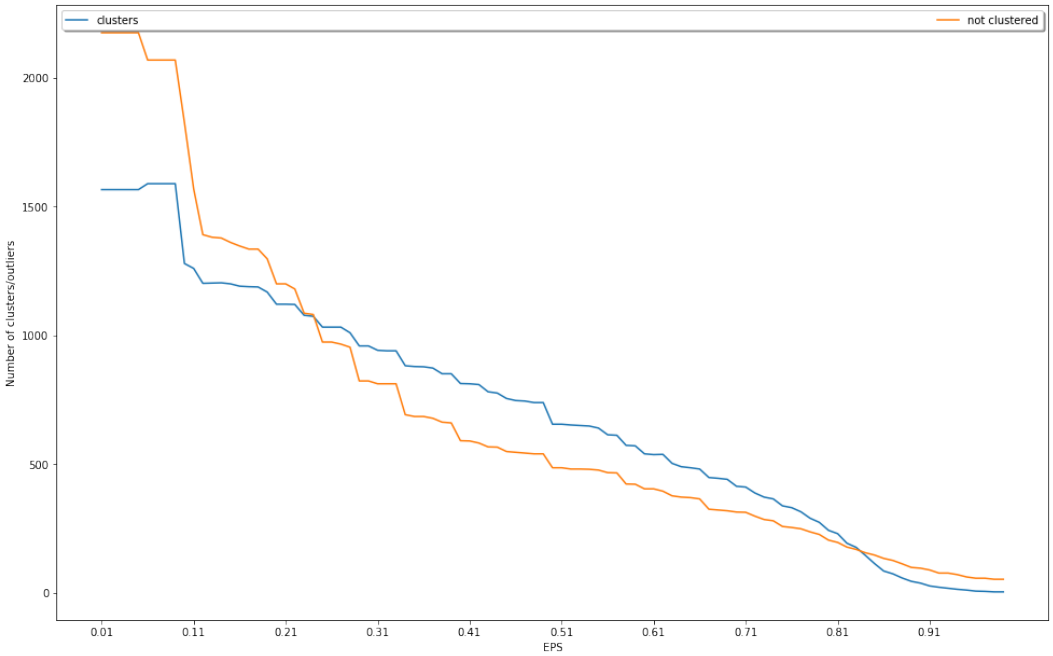

Figure 4.7: Clusters and outliers for changing EPS for bodies clustering. MinPts = 2

4.1.3 URLs

URLs often present in SPAM. Figure 4.8 depicts histogram of number of URLs in the emails found in F-Secure's spam database (10000 emails were sampled). Emails without any URLs compose only 0.01% of the dataset.

URLs are part of spammers' resources. Domains cost money, and it is quite expensive to use unique domains for each email in big campaigns. Although domain names can stay the same for different campaigns, unlike topic or similar template, it is much less likely that different spammers share control over same domain names. Finding campaigns that share domain names can help to identify likeliness of campaigns to be sent by the same spammer.

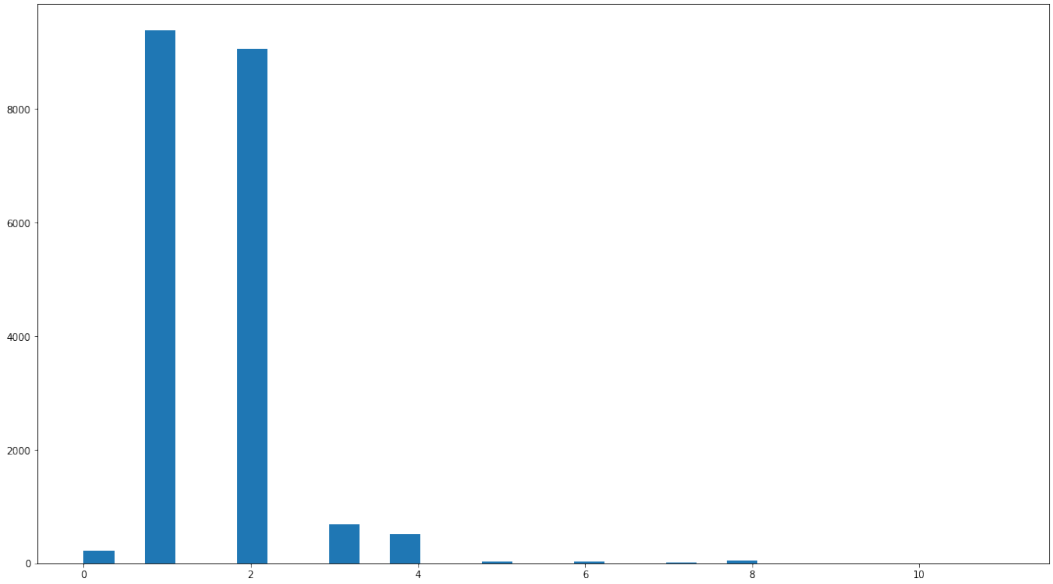

Figure 4.8: Histogram of URLs count in used dataset

Clustering

Clustering by URLs had been proposed[37][44][45][29] before and was discussed in section 2.2.4. URLs have several features that that were considered:

- Domain names

  Domain names cost money, although it is possible to create sub-domains almost for free, it is still possible to extract domain name cutting off sub-domains. One of the drawbacks is that some services might sell their sub-domains or provide a service to redirect requests, for example, URL shortener discussed in section 2.2.4. However, it will always lead to the real server controlled by the spammer and can be extracted from the email by following the shortened link.

- URL patterns

  URL patterns are easy to change in theory, but practical experience shows that it is often not done by spammers.

Full domains had been chosen as a feature for clustering as it is easy to implement and validate the approach.

Clustering is done in next four steps:

1. Parsing the whole body and extracting all used domain names.
2. Creating a set of used domain names.
3. Creating a distance matrix using Jaccard index based distance.
4. Running DBSCAN clustering on the calculated distance matrix.

This approach can perform worse if spammers append fake URLs to the end of the email or make them invisible. In that case, Jaccard index will output significantly lower value if sets will have many different domain names. However, it is not highly likely as it will not help email message to get through the spam filter and manual inspection showed that it does not happen in practice.
Estimation of Clustering Parameters. DBSCAN was run with different EPS values to evaluate the results. Figure 4.9 shows overall results. As can be seen from the picture, TP and FP values are on a significantly lower level than true and false negatives. False negatives number is much bigger because spammers often use different domains during one campaign.

Figure 4.10 depicts only false positives and true positives. Level of true positives is higher from the start, and at EPS 0.31 it jumps while the number of false positives stays the same. The second jump appears at EPS 0.5, but now both values increased. On value 0.66 both false positives and true positives jump, but false positives go up significantly faster than true positives.

EPS = 0.46 was chosen as the optimal value. It will provide strict information about the similarity of used domains and will tend to provide as least false positives as possible.

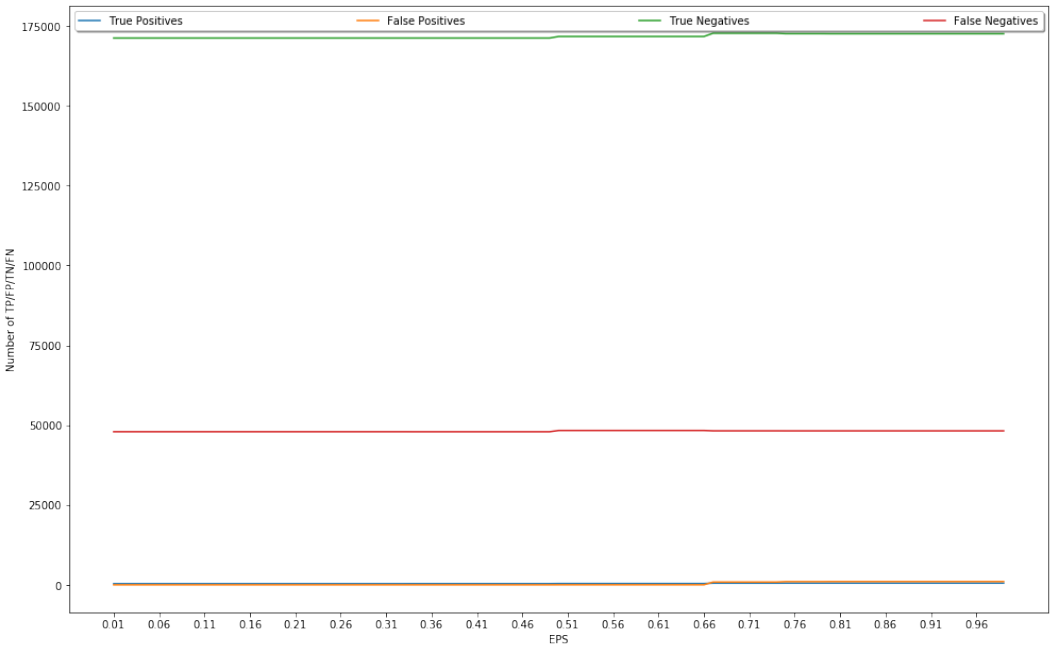

Figure 4.9: TP-FP-TN-FN EPS evaluation for URLs clustering. MinPts = 2

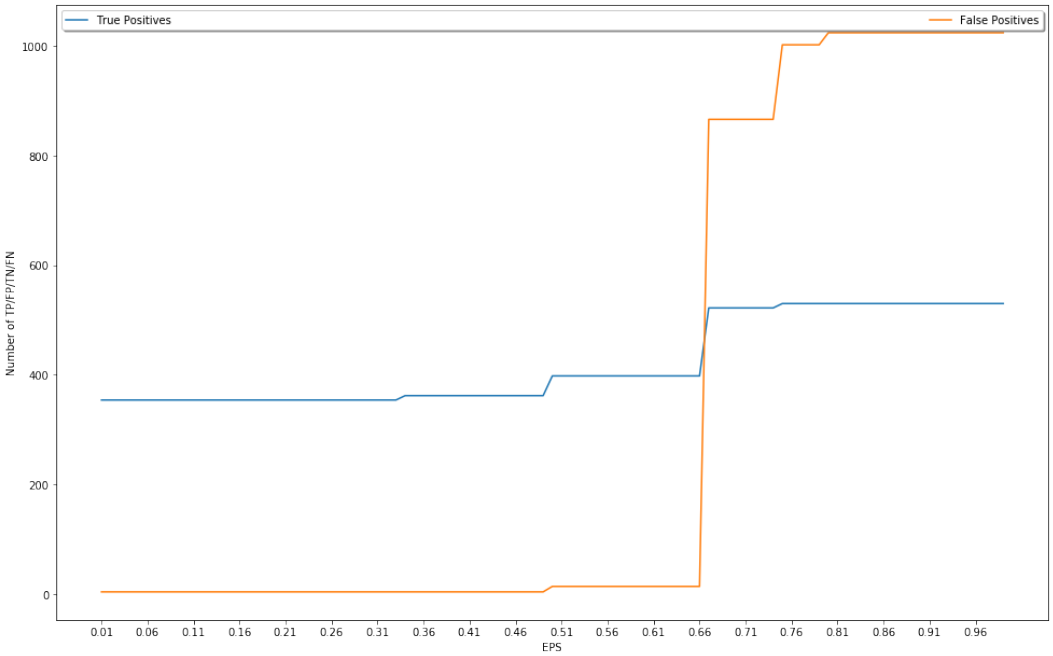

Figure 4.10: TP-FP EPS evaluation for URLs clustering. MinPts = 2

4.1.4 Source IP addresses

IP address used to send an email is a very important feature that can be used to identify the real sender. Unfortunately, based solely on a received email message, it can be hard or even impossible to determine the real IP address that was used to send the message. As discussed in section 2.2.4, IP address can be forged as well as "Received" headers. Meaning that even if IP address should always be in "received" headers list, it can be impossible to say which one exactly is the real source IP address.

Because this feature is not reliable, it was decided to postpone it to future researches. Some of the possible solutions might be taking several possible addresses from the Received headers and compare them as sets.

4.1.5 Attachments

Attachments can serve different purposes that can be divided into two groups: attachments providing additional content to the message and attachments serving as a separate payload that has to be opened by the recipient explicitly.

Each attachment has a file name and a content. The name often contains extension and the name of the attachment. Thereby three features from each attachment can be extracted: file name, file extension, and content.

Attachments related features are valuable and often can show that two emails are from the same campaign.

4.1.6 Recipients

Apart from infrastructure for sending messages, a spammer has to have recipients to send spam to. Lists of email addresses usually bought or generated by the spammer and these lists evolve through the spammer's career. Not active addresses are dropped, and new addresses are being added to improve profits from sending spam. Each experienced spammer has his own sources of email addresses thus recipients lists can be quite unique.[40] Therefore such resource as "list of recipients" can help to identify the actors standing behind the spam campaign.

Each email has at least one recipient and does not provide much information about the whole spammer's mailing list. However, if spam dataset contains many email messages that can be definitely grouped by some other features, for example, all of the messages have the same subject line, same body text and same URLs inside, all recipient email addresses can be collected into one set for that campaign. The bigger set is - the better it represents the original list of emails that spammer used for the campaign. By counting similarity between mailing lists of different campaigns, it might be possible to correlate different campaigns.

4.2 Visualization

Visualization of the data is one of the key aspects of the developing tool. This section discusses selected approach for the visualization, leaving elaboration of specific technologies to the 5.1.3 section, which discusses implementation-specific details.

Clustering does not reduce the amount of data. It produces more data - cluster labels. However, produced data can be used to structure existing email messages as sets of similar objects.

Preserving only important information helps to reduce time finding it. Thus only important information is left for each email message:

- Subject line - allows to get quick estimation of what email is about. A subject is always the first thing that recipient sees, so it is often a quite good representation of the email message.

- Attachments names and SHA1s - attachments often can define email message purpose. Many file reputation systems use SHA1s as keys for getting more info about the file so it is often quite handy to have SHA1s at hand. File names often allow to understand the purpose of the file as it also contains extension in it and short name that is showed to the recipient. In some cases the purpose of the file is obvious only from its name.

- URLs - seeing URLs found in email allows to investigate each link by

hands if it is needed. Also it is often obvious if it is a URL shortener, picture, or something else.

- Way to open original email - one of the important features is to have the ability to get all info about the email in the original form. Showing email file path allows opening the email fast in any application, for example, Thunderbird.

Yet one problem to solve is displaying clusters and relation between them. One possible solution is to utilize graph concept. Each cluster can be a node, and all emails that are in that cluster can be nodes connected to the *cluster node*. Thus if one email is in two clusters, relations between these clusters will be visible to the user. Figure 4.11 demonstrates how it might look like. All email related information is hidden intentionally to let the reader focus on cluster visualization and not emails themselves.

Figure 4.11 depicts 3 clusters: *31_bodies*, *254_subjects*, and *1635_bodies*. These clusters represent clustering by different features, according to their names: bodies and subjects.

Yellow nodes represent emails that belong to these clusters. For example, email *11717* belongs to 2 clusters: *254_subjects* and *31_bodies*.

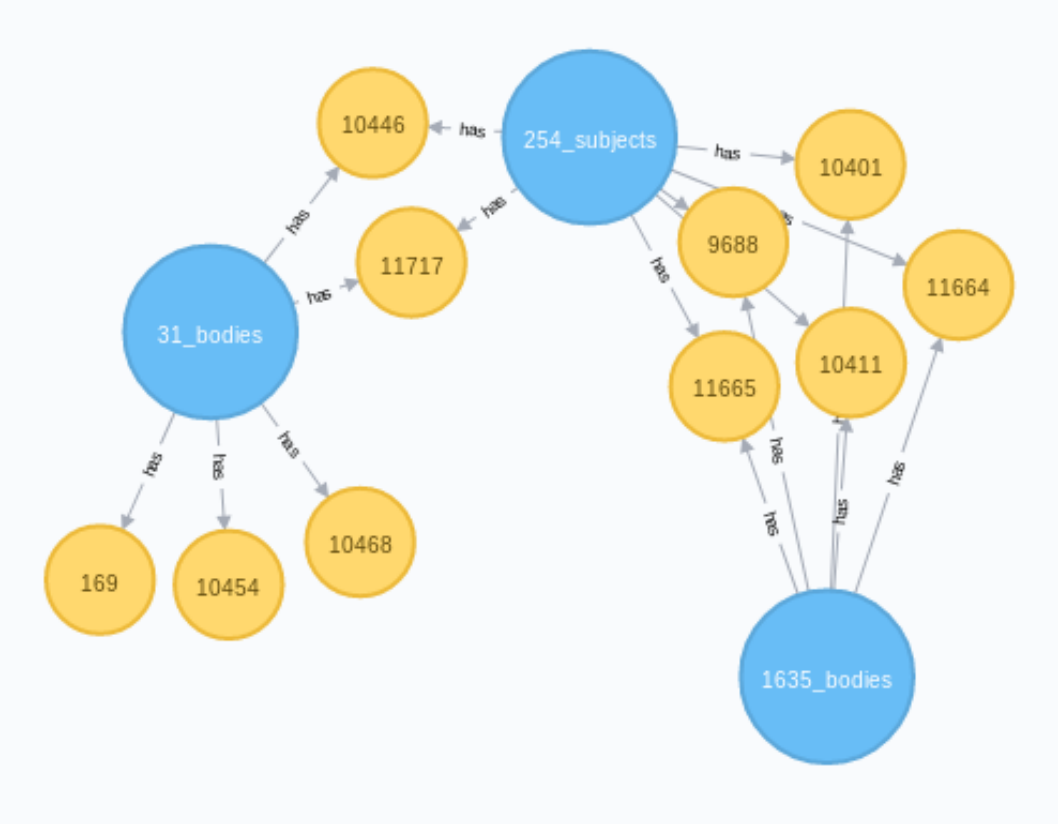

Figure 4.11: Clusters visualization example

The benefit of such visualization is that it is easy to see not only what is similar and different between emails, but also how emails with all different features still relate to each other through intermediate clusters. For example, it is easy to see that emails 169 and 10717 have same bodies' contents because they are in the same cluster *31_bodies*. However, 11717 is also in the cluster *254_subjects* that has other five nodes that do not belong to the *31_bodies* cluster. But all these five nodes have the same bodies. Intercluster connections allow correlating different features together only by examining the graph model.

Chapter 5

Technical implementation

Internal and external feeds are alike but still have some diversity in sizes of spam campaigns represented in them. Also, because external feeds are bigger, they have a better representation of small campaigns than the internal feeds. Extracting data from the feeds is automated and can be implemented in several lines of infrastructure-specific commands.

Working with such big amount of data requires quite thought through set of technologies. Even simplest operation, done without thorough inspection, can take days to run. Complexity of infrastructure and used software often adds up to a unforgiving piece of code that will work impermissibly long even if logic seems right. It happens because each operation is multiplied on number of processing objects and when number of objects is huge, all imperfections of the process pops out as spikes in memory consumption, network traffic and processing time.

Working with the whole dataset is impractical thus sampling had been applied to the dataset. Sampling is a way of data selection that allows to work with lesser data keeping general properties of the whole set. Sampling for the spam data set can be done in few ways: selecting all samples from one day or taking samples randomly from the whole dataset. For this research, two databases had been created with sampling.

Dataset 1 consists of 10000 emails over 2017-th year that have attachments. It mostly contains malware but some emails are just spam with not malicious. Spam usually has either attachment or URL in it. It is rare to see those together as it increases probability of message being put into spam because of SpamAssassin rules summing up. Thus it is rare possible to see URL clusters working with this dataset.

Dataset 2 consists of 20000 emails over 2 days in January 2017-th. Purpose of this dataset is to test the tool on continuous dataframe of real-life data. Most of the spam is not malware and does not have attachments. Usually spammers advertise goods and provide URLs that lead recipients to the website where they can spend their money.

5.1 Design

At the moment of writing the thesis, the system is in a proof of concept stage, so the user has to control the whole workflow himself. However, the process can be easily automated, and it is strictly determined. Figure 5.1 depicts the clustering process.

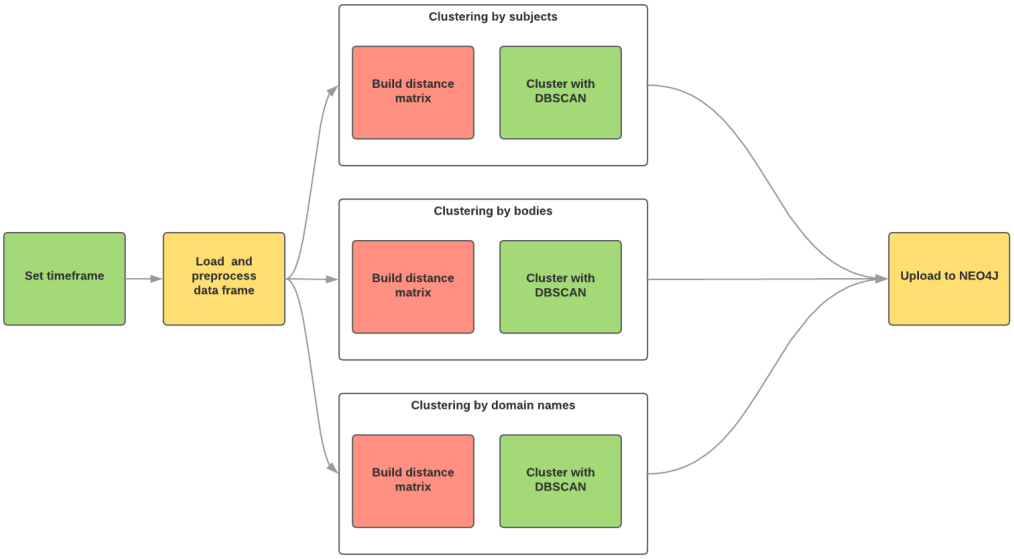

Figure 5.1: Clustering process

The program takes email messages, stored as files in a raw format, from a given directory path, splits them on bulks of 20000 messages, and runs clustering algorithms for each bulk. After running the dataset through the clustering algorithms, the program creates three files, one for each clustering algorithm. These files have to be uploaded into Neo4j database using the functionality of Neo4j. After uploading the data, researchers can access Neo4j database from any browser and make requests to the graph.

5.1.1 Loading and parsing emails

Data is loaded and pre-processed in an IPython notebook. Notebook loads all email addresses from given path, parses them using "mail-parser"[41] library, extracts useful features and saves the result as a pickle file - python serialized object written to a file. Extracting features are:

- File path

  Used to identify the message and provide additional functionality, like displaying the original message, later.

- Subject

- Message body

  Some email messages that are intended to be shown as HTML have two bodies: first has a note that message has to be viewed in HTML-supporting email client and second has the real message. Thus parser extracts the last available body.

- List of recipients
- List of attachments

  For each attachment, only SHA1 and filenames are extracted. SHA1 is counted using "hashlib" - standard python library for hash digests.

- Parsed text

  HTML payloads have to be parsed, or it will interfere with body clustering as each HTML tag will be considered a separate word.

- List of email body URLs

Loading each 10000 email messages takes on average 12 seconds. One day spam from internal spam feed can be parsed in less than a minute.

5.1.2 Clustering

All clustering algorithms, including distance matrix, are implemented as IPython notebooks. As result of calculations, they provide a one-dimensional array of numbers. Each *i-th* number is a cluster label for the *i-th* email sample in the input pickle object. For each clustering, a directory is created, and result arrays are put there as pickle files.

For the proof of concept implementation, only 3 clustering methods had been utilized: bodies, subjects, and links. It is sufficient to evaluate the approach, but it can be improved by implementing other clustering methods.

Bodies clustering module extracts set of words from each email, calculates distance matrix and uses DBSCAN to cluster these sets of words. The distance matrix is calculated using distance function based on the Jaccard index.

The next algorithm processes each body text:

1. BeautifulSoup[35] library with "html5lib" parser is used to parse the HTML body.
2. Head tag is removed including the contents.
3. Plain text is extracted using "get_text" method from BeautifulSoup.
4. URLs are being removed using the next regular expression:

   ```
   (https|http)?:\/\/(\w|\.|\/|\?|\=|\&|\%)*\b
   ```

5. Plain text is tokenized with "nltk.word_tokenize", all words are made "lowercase," and all digits and punctuation are being filtered out.

6. Dictionary of all words is created, and integer ID is assigned to each word. Comparing words is much more computationally demanding than comparing numbers.

7. All words in bodies are replaced with integer IDs, and bodies are converted into integer arrays with the fixed length of 20, cutting the rest of the body. Selection of the length is discussed in the algorithms section 4.1.2. If body length is 0, then the message is excluded from clustering.

8. Similarity matrix is created by comparing all messages to all other messages using Jaccard Index. Results are stored in "ndarray" type from "numpy" library. It allows to preserve memory [28]. Jaccard Index is a similarity measure from 0 to 1 and not a distance. To get distance measure, it has to be subtracted from 1.

9. DBSCAN algorithm is run with next parameters:

```
db = DBSCAN(
            eps=0.2,
            min_samples=2,
            metric="precomputed",
            n_jobs=-1)
labels=db.fit_predict(similarity_matrix)
```

   metric="precomputed" tells that input is a precomputed distance matrix
   n_jobs=-1 allows to use all found cores of all CPUs
   eps and *min_samples* are DBSCAN's parameters explained in 2.3.2 section.
   DBSCAN algorithm is implemented in "sklearn"[30] library.

10. Results are saved as a pickle file.

Domain names clustering extracts all domain names from email's body, combines them in sets, calculates Jaccard index based distance matrix on these sets and clusters sets with DBSCAN. The algorithm is similar to the "bodies clustering," but instead of words works with domain names.

1. Extract plaintext from the body.

2. Extract all URL links and keep only domain names, combine them into a set. If set is empty, object is excluded from clustering.

3. Calculate distance matrix using Jaccard index based distance.

4. Perform DBSCAN with next parameters:

```
db = DBSCAN(
            eps=0.51,
            min_samples=2,
            metric="precomputed",
            n_jobs=-1)
labels= db.fit_predict(similarity_matrix)
```

5. Results are saved as a pickle file.

Subjects clustering builds distance matrix from all subjects using Levenshtein distance function and performs clustering with DBSCAN algorithm.

1. Extract all subjects.
2. Calculate distance matrix using Levenshtein distance.
3. Perform DBSCAN with next parameters:

```
db = DBSCAN(
            eps=0.34,
            min_samples=2,
            metric="precomputed"
            ,n_jobs=-1)
labels= db.fit_predict(similarity_matrix)
```

4. Results are saved as a pickle file.

5.1.3 Visualization

Neo4j database is utilized to visualize, store and query the results of the clustering. Neo4j is a NoSQL graph database that outperforms all other graph databases in common tasks[18] and has big developing community. It uses *Cypher* query language that allows writing complex queries for the database. Neo4j is distributed as a docker image that includes web-based graph visualization and Neo4j engine.

To visualize and work with the data in Neo4j, data has to be loaded there first. Neo4j allows loading data from CSV documents row by row, executing *CREATE* queries for each row. However, firstly, CSV documents have to be generated.

To generate CSV documents, the python3 script loads pickles with clustering results and original emails to the memory. Then it generates next two files: *clusters.csv* and *nodes.csv*.

clusters.csv has 3 columns: cluster_type, cluster_id, email_id. Cluster_type stores one of 3 types: bodies, subjects or links. It describes the type of cluster. An ID of a cluster is a type of a cluster with a unique number that identifies the cluster appended to the type. Email ID is a unique ID of email that belongs to the cluster. The file contains all relationships between emails and clusters that they are in.

nodes.csv has 5 columns: id, subject, sha1s, path, URLs. Columns do not influence the internal structure and are needed to show information about each email. Email ID is identifier that is used in *clusters.csv*.
Files are put into "/data" directory in the docker container and loaded into the database using Cypher queries:
Load all emails as nodes

```
load csv with headers from "file:///emails.csv" as row
fieldterminator '|'
create (:Email {
subject: row.subject,
id: toInteger(row.id),
path: row.path,
sha1s: row.sha1s,
urls: row.urls}
)
```

Load all clusters as nodes

```
load csv with headers from "file:///clusters.csv" as row
fieldterminator '|'
merge (c:Cluster{
type: row.cluster_type,
id: row.cluster_id
})
```

Connect all emails to according clusters

```
load csv with headers from "file:///clusters.csv" as row
fieldterminator '|'
match (c:Cluster{
id: row.cluster_id
})
match (n:Email{id:toInt(row.email_id)})
create (c)-[r:has]->(n)
```

To make Neo4j perform faster, nodes and relations must be indexed:

```
CREATE INDEX ON :Email(id)
CREATE INDEX ON :Cluster(id)
CREATE INDEX ON :Cluster(type)
```

After the indexing, the database is ready to be used.

Chapter 6

Evaluation

This chapter presents some results obtained by working with the tool. The main intention of the chapter is to evaluate introduced approach and tool implementation by providing some examples of how the tool can be used and explain the results.

6.1 Clustering performance evaluation

The most computationally complex task is to create distance matrices for the data. Figure 6.1 shows how the time for building the distance matrix increases with the increase in the number of emails. It is evident that clustering by subjects is more time complex than other methods. The reason is that Levenshtein distance, used in subjects clustering, has higher time complexity than Jaccard index, used in other clustering methods. Time rises in a quadratic manner.

After building distance matrix, DBSCAN algorithm is executed. DBSCAN has an average complexity of $O(n \log n)$ and worst of $O(n^2)$. Figure 6.2 depicts comparison of DBSCAN computation times. Graphs are quite similar because in all methods DBSCAN takes already precomputed distance matrices. DBSCAN computation time is neglectable compared to distance matrix computation time. However, it rises rapidly too.

The complexity of the algorithms adds some limitations to the tool use cases. The main limitation is that sets of data cannot be too big. However, even with this limitation, the tool provides significant value. It can be used to provide statistics for relatively short periods of time, or it can be used for longer periods with the use of sampling. Next section discusses practical use cases and what can be achieved with the developed software.

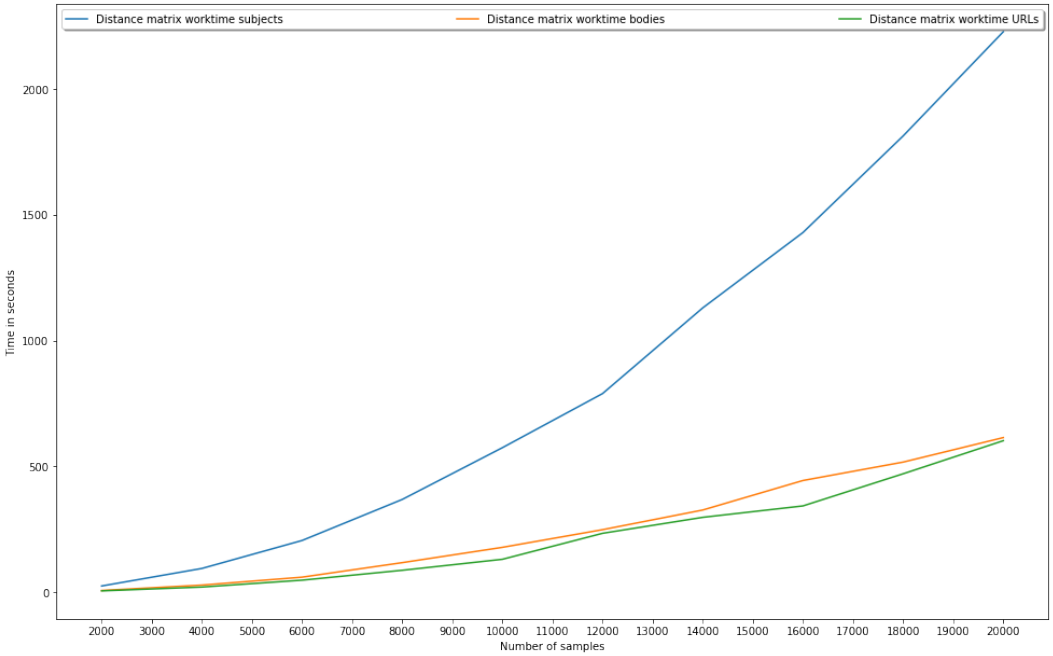

Figure 6.1: Time complexity comparison for distance matrix creation

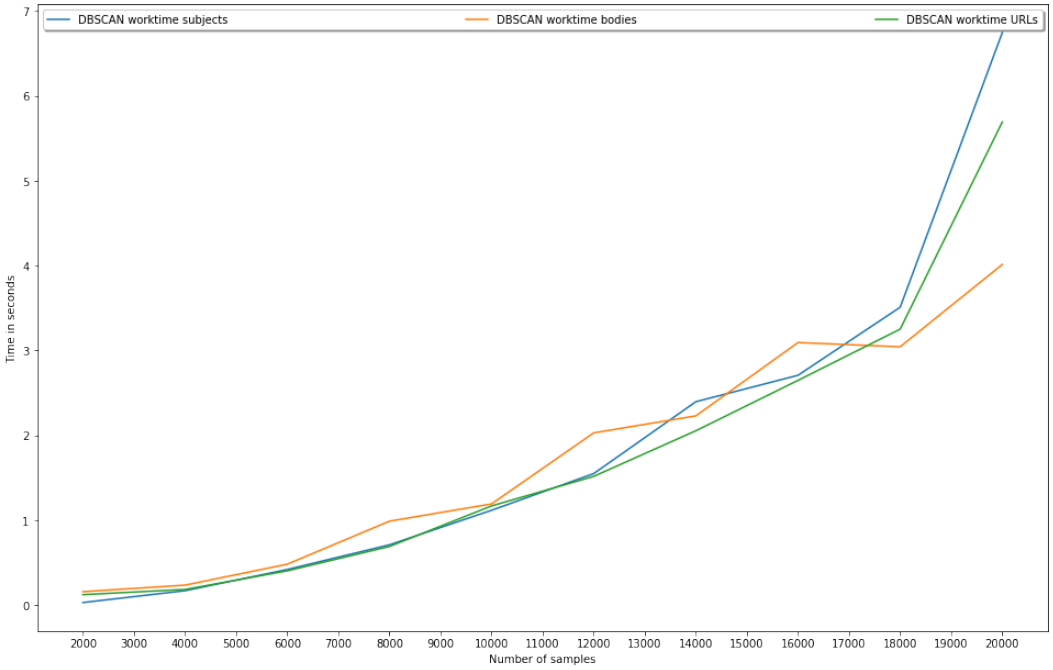

Figure 6.2: Time complexity comparison for DBSCAN clustering

6.2 Practical results

The developed tool provides quite valuable output and can be used to easily detect very complex campaigns that cannot be easily extracted with any other tool that is used in the company. In this section, several use cases are being discussed to show what kind of insights can be gained with the developed tool on real data.

6.2.1 Synonym swapping

Synonym swapping method, as discussed in 2.2.4 section, allows to generate unique sentences using sets of interchangeable words and collocations that are composed together randomly. This approach allows generating big amounts of different email subjects and bodies where most of the randomly taken pairs of samples will have only a few words in common. Such campaigns are not easy to disclose even to humans. Fortunately, the developed tool can handle such campaigns.

DBSCAN, utilized for subjects and bodies clustering, can follow the cluster, combining all nearby objects. Thus if spam feed contains enough samples, DBSCAN will combine close ones to the point where all or most of the generated texts will be in one cluster.

Dataset 2 contains a few campaigns with generated subject lines. After running clusterization for the dataset, 400 messages were clustered into the same cluster. All messages cannot be disclosed due to the NDA restrictions. However, *synonyms state machine* had been recreated from the messages, and it is depicted in figure 1. Although synonyms tables have been known for a long time, this example shows capabilities of the tool to find such generated campaigns. Knowing the table used for generation, such spam campaign can be mitigated by blacklisting all possible combinations.

6.2.2 Tracking by features

Spammers often change some parameters of the campaigns to improve the effectiveness of the campaign. However, some features stay the same. One campaign can undergo many changes during a long time and often set of features changes completely, so it is no longer possible to cluster two messages without intermediate ones.[29]

Correlating completely different sets of features by tracking the transitions is a novelty that this research introduces. It is achieved due to clustering by different methods and connecting intersecting clusters together, visualizing results as a graph.

Here are some examples of visualized campaigns, showing how complicated feature transitions can be and how developed tool makes understanding the data easier.

Parcels campaigns.
Figure 6.3 demonstrates part of one campaign as a graph. Clusters are blue nodes, emails are yellow nodes. Picture depicts several overlapping clusters: *27_subjects* overlaps with *16_bodies*, and *16_bodies* overlaps with *37_subjects*. An overlap means that some emails got into two clusters in the same time. Table 6.1 shows contents of the highlighted messages.

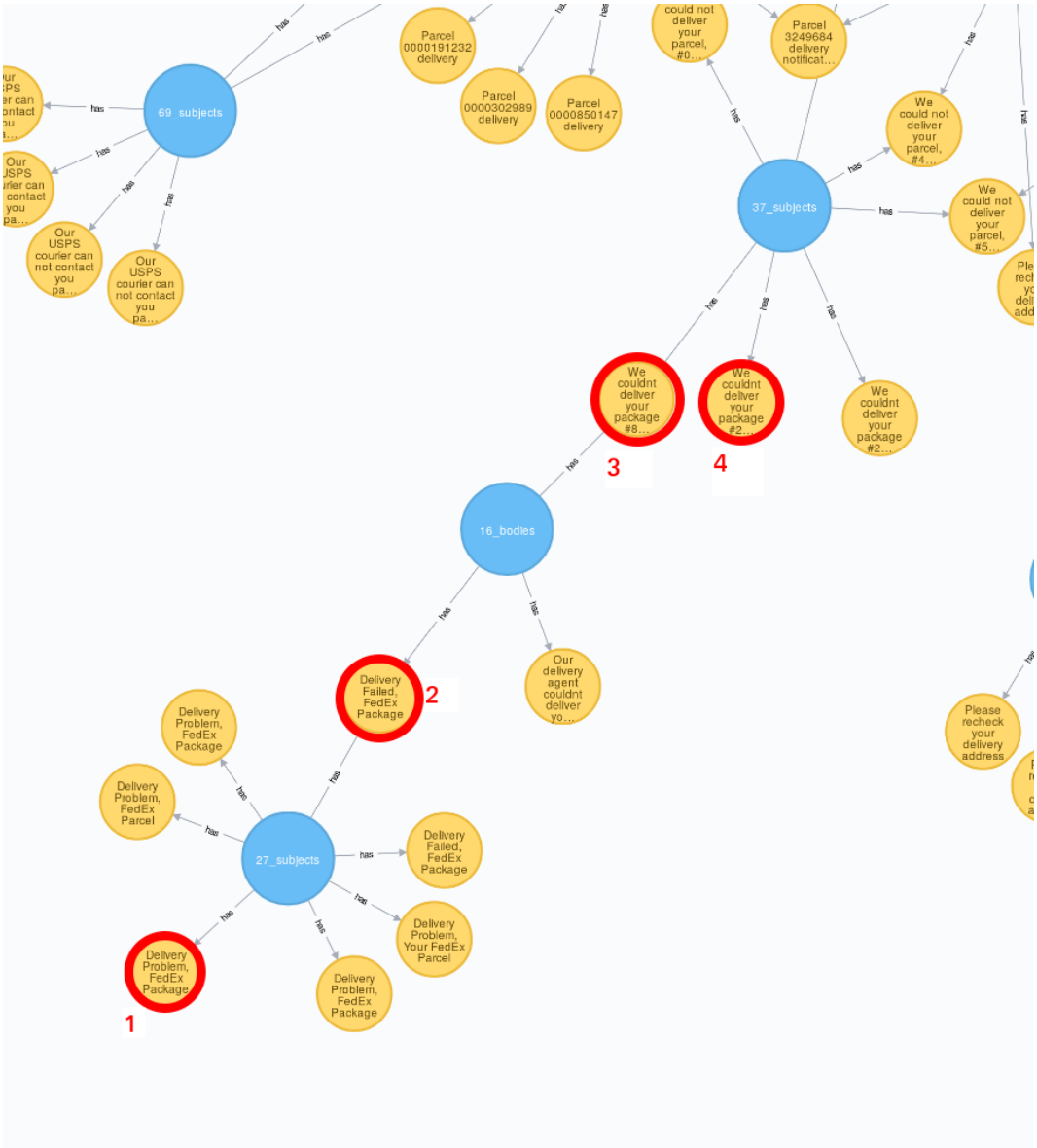

Figure 6.3: Clustering by tracking changes of features

ID	Subject	Body
1	Delivery Problem, FedEx Package #xxxxxxxxx	Hello, Your order is here, but our delivery representative was not able to deliver the package. Important details can be found in the confirmation document attached. Thanks for your consideration. Sibby Glander - FedEx Expedited Supervisor
2	Delivery Failed, FedEx Package #xxxxxxxxx	Hello, By this letter, we'd like to let you know that we've tried to deliver your package today. Additional information about your parcel is in the shipment notice enclosed. We're glad to help you any time. Letta Czap - FedEx Delivery
3	We couldn't deliver your package #xxxxxxxxx	Hello, By this letter we'd like to inform you that we've tried to deliver your package #xxxxxxxxx today. Additional information about your parcel is in the shipment notice enclosed. Thanks for using FedEx. Adaline Casper - FedEx Expedited Delivery
4	We couldn't deliver your package #xxxxxxxxx	Hello, Delivery status notification: our delivery representative was not able to deliver your package today. We kindly ask you to check the delivery confirmation attached for more details. Thank you for using FedEx. Ysabel Shilleh - FedEx Delivery Agent

Table 6.1: Features used to cluster parcels campaign

As can be seen from the table 6.1, messages 1 and 4 do not have anything in common. Messages 1 and 2 have almost identical subject lines but have different bodies thus they are in the same cluster - *27_subjects*. Messages 2 and 3 have different subject lines but identical body messages that were easily clustered into *16_bodies* cluster. However, message 3 and message 1 do not have anything in common: subjects are different as well as bodies. Message 3 and message 4, on the other hand, have identical subjects. Although message 1 and message 4 do not share any features, by tracking changes it can be assumed that they were sent by the same actor as part of one campaign.

Malware correlation.
For this example we have a hash of malicious file that came from some spam campaign and we want to find all attachments that came from the same campaign. To do that we can query all nodes that are closer than 10 nodes away. Here is the query that does that, sorting attachments by distance (hash is randomized):

```
match (e:Email)-[r*1..10]-(s) , p = shortestPath((e)-[*]-(s))
where e.att_sha1s contains "317296*3ca4"
and s.att_names contains ".zip"
return distinct s.path,s.att_names,s.att_sha1s,length(p)
order by length(p)
```

The queried subgraph is depicted in figure 6.4. Table 2 contains all extracted attachments. As can be seen from the table, most of the attachments pretend to be from a delivery company. Thorough checking using internal file reputation systems also show that all of them contain malware from one family.

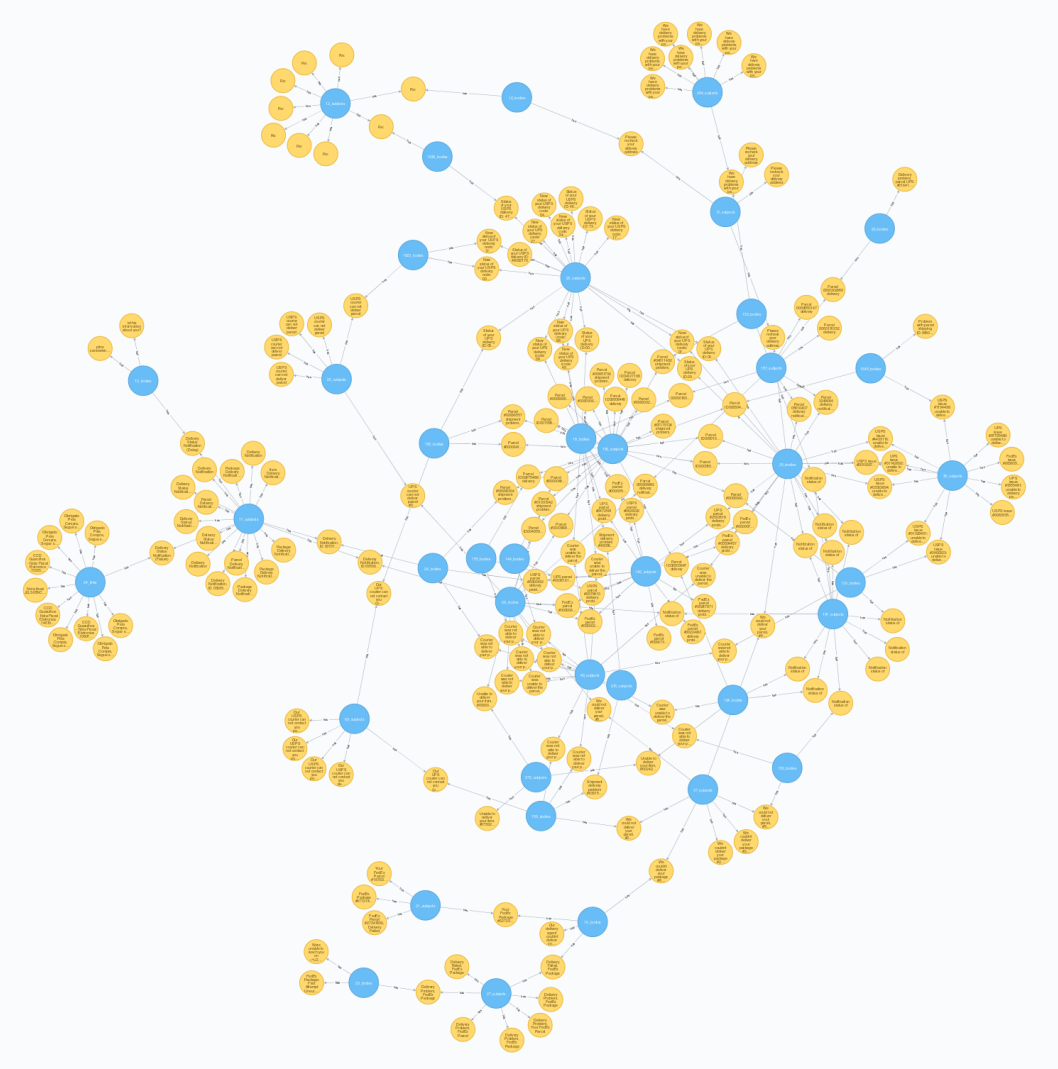

Figure 6.4: Subgraph of campaign distributing malware pretending to be a delivery service

However, suggested approach can sometimes provide false positives in cases where different campaigns accidentally appeared to have same features. For

example, at the end of the table 2, few attachments look differently. "warez_-tear.zip" does not look like part of the delivery campaign. Figure 6.5 is part of subgraph explaining the connection. The problem is in node 1. It accidentally has the same name as node 3 and same body as node 2. Node 2 has attachment "warez_tear.zip" that is not part of the delivery campaign.

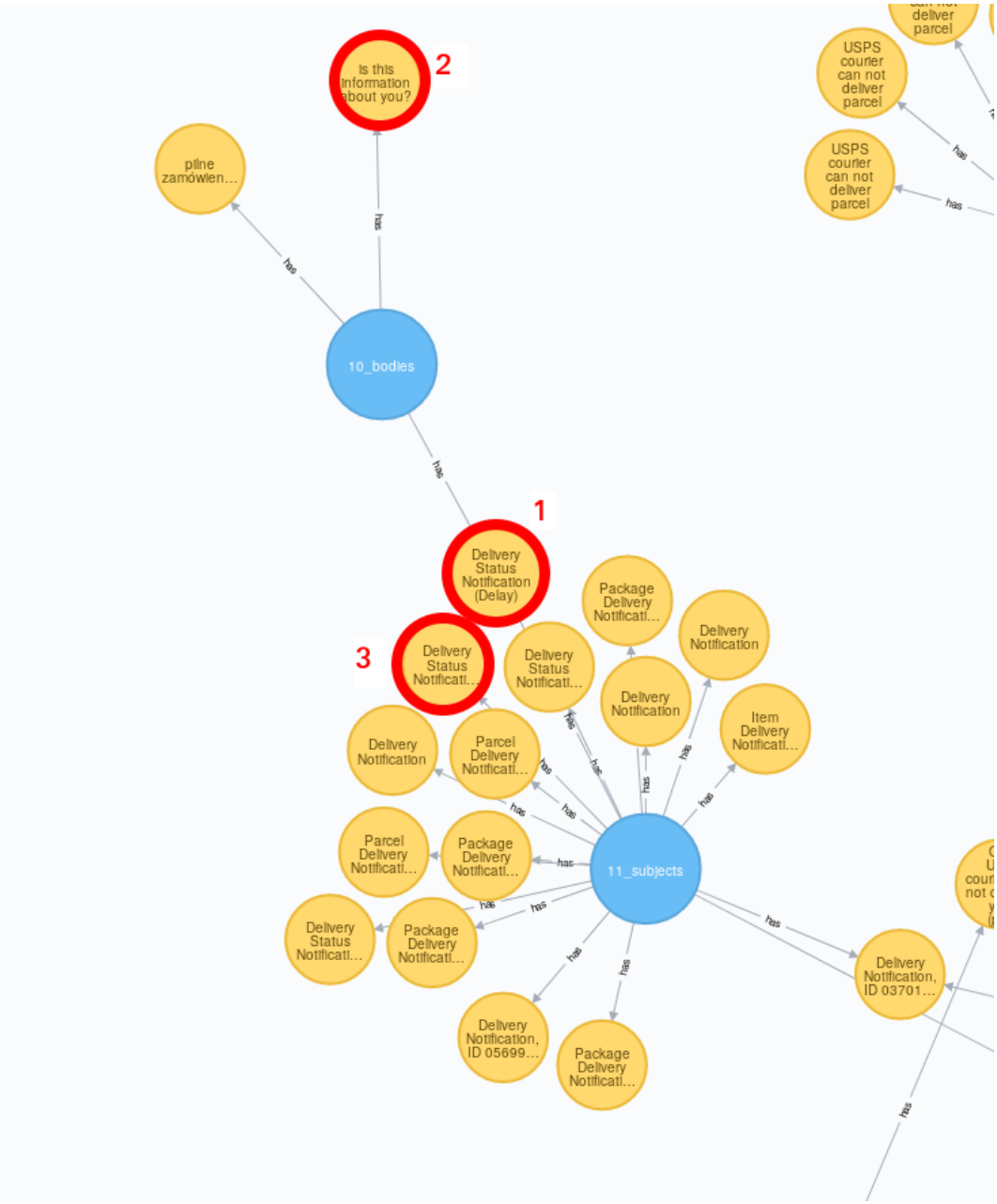

Figure 6.5: Wrong connection between clusters

6.2.3 Same identity clustering

Sometimes seemingly different campaigns get clustered together. However, it does not always mean that clustering is incorrect. Sometimes spammers leave same features in different campaigns, disclosing that the same spammer had sent those campaigns.

Amazon and Flashlights. In this example, spammers sent a few messages of the new campaign, using domain names from the old one. It might be due to a mistake, or it might be that domain from the last campaign did not get banned and it was still profitable to send messages storing content on it.

Figure 6.6 depicts these two campaigns. One campaign is a phishing campaign that tricks people into thinking that they received a message from Amazon. Second is advertising flashlights. Nodes 1 and 3 do not have any common features. Even topic is different. However, messages 1 and 2 have same domain names. Messages 2 and 3 share common subjects. Exact feature values can be seen in the table 6.2.

Id	Subjects	Domains
1	Free Military Grade Tactical Flashlight!! Limited Time Only	[imagizer.imageshack.us, leadecompany.cricket]
2	Your amazon prime points are about to expire	[imagizer.imageshack.us, leadecompany.cricket]
3	WARNING your amazon points are about to expire	[www.home8they.cricket]

Table 6.2: Features used to connect Amazon and Flashlights campaigns

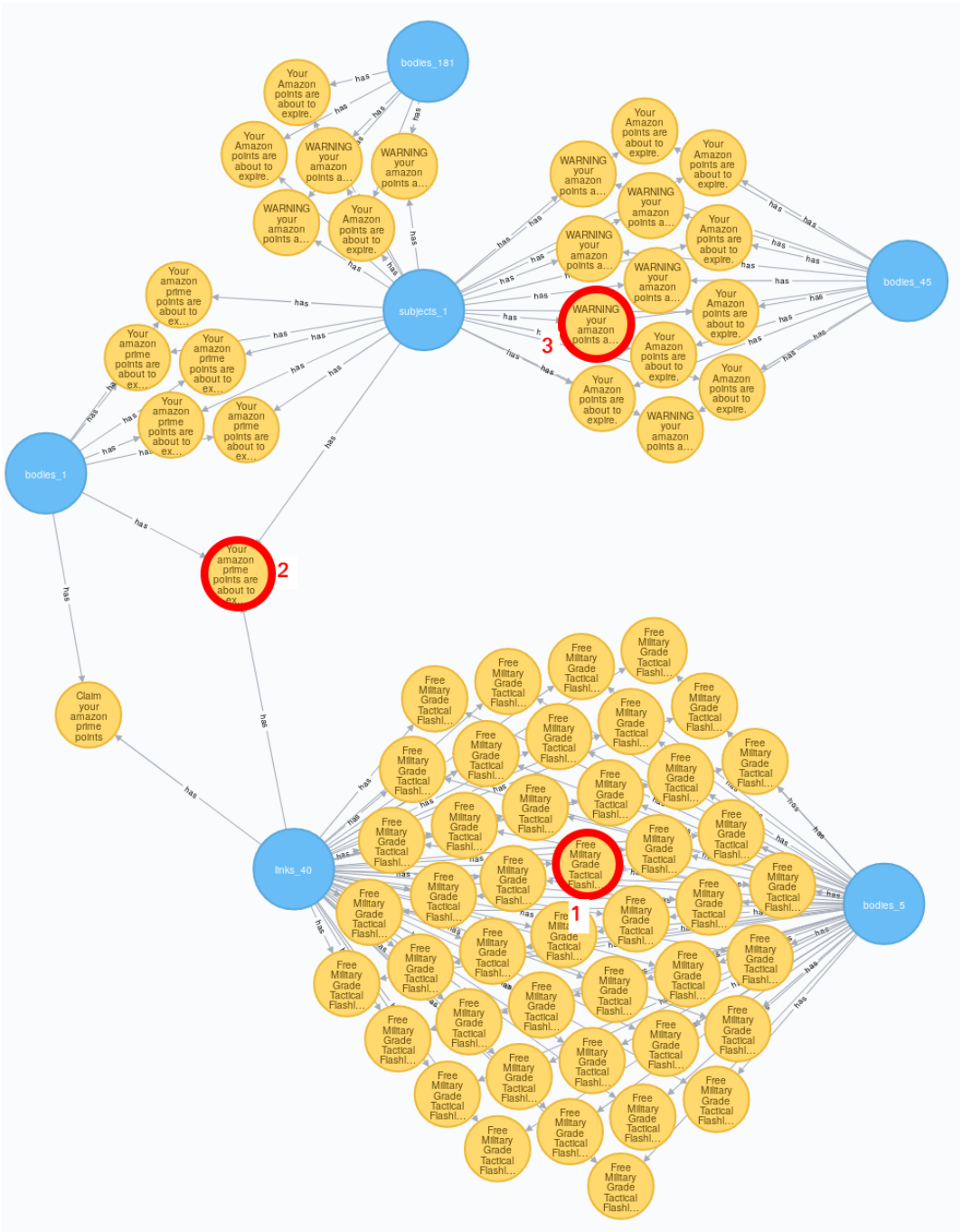

Figure 6.6: Different campaigns clustered because of common resource

6.3 Researchers' comments

Two spam researchers from F-Secure have been asked to try the tool, tell if they would like to see the tool as part of their workflow and provide comments in a free form.

Both researchers found the tool capable of improving the workflow but had

a bit different opinions on how to integrate it. One suggestion was to use it as an independent tool for research. Another was to integrate just results of clustering into the existing database with spam as campaign labels. Here are some other comments:

Can be part of the spam workflow. Both researchers said that they want to see the tool as part of their workflow. In one case as a complete tool for research, in another - as part of the existing system.

Learning curve is shallow. It took less than an hour to figure out the concepts and start performing rather complex queries. The Neo4j basic user interface allows using a mouse to navigate a graph and expand nodes.

Tool does not provide sufficient statistics mechanisms that are available in current implementation. Neo4j allows to use third-party visualization modules, but it still needs further research. One way is to transfer results of clustering, supervised by a researcher, to another database that can provide some dashboard. Another option might be to write a tool for visualizing the results. This part still needs some research to find the optimal way of providing a statistical overview of campaigns.

6.4 Comparison with previous work

Approach in this paper was influenced by the "Mining Spam Email to Identify Common Origins for Forensic Application"[45] research. Main similarity is that both researches use a set of features simultaneously to correlate spam. However, some key differences allow to achieve better results.

"Mining Spam Email to Identify Common Origins for Forensic Application" used graphs only to reduce false positives, created by the agglomerative clustering. Clustering implementation combines existing clusters by a new feature each iteration. Each iteration is very simple, features are compared directly, without machine learning techniques. So, for example, if two messages differ by a few characters, they will not be clustered.

After getting too big cluster, researchers decided to split it using graph approach. Graph approach used two features: subject line and domain name. Domain names were presented as nodes and subject lines that had been sent from two different domains were depicted as edges connecting these domains.

In this thesis, machine learning techniques are used to detect patterns and close features in the first place. It is done several times by different features, and each feature clustering returns already some results. For example, subject lines are clustered by Levenshtein distance and DBSCAN. It already allows to detect campaigns with subjects, generated using synonym swapping technique, as shown in section 6.2.1.

Graph theory is used to correlate intersections of these results. It allows to mitigate false negatives by combining small campaigns into bigger ones and allows to do deeper analysis of the existing clusters, like reducing false positives as it is done in "Mining Spam".

Results are hard to compare as datasets are different. However, in each research use of graphs helped improving the base approach. In this research it had been shown as examples of usage.

6.5 Requirement fulfillment

Implemented tool meets all the requirements stated in 3 chapter. Table 6.3 depicts all the requirements and how they were fulfilled. It allows to evaluate relevance of developed software to what researchers expected in the beginning of the research.

Requirement	Fulfilled
FR1: Clustering	Yes
FR2: Searching	Yes
NFR1: Performance	Partially
NFR2: Visualization	Yes
NFR3: Learning curve	Yes

Table 6.3: Evaluation of requirements fulfillment

FR1: Clustering.
Emails are being clustered on two levels. The first level is machine learning clustering. The second is clustering by correlating emails through existing clusters.

FR2: Searching.
To search through the data, Neo4j uses Cypher.[16] Cypher is a graph query language, providing a way to match patterns of nodes and relations in a graph. It had been originally released in 2007 and still evolving.

NFR1: Performance.
Performance was achieved partially. The requirement was regarding being able to analyze data in real time, and it is fulfilled because searching and working with the data is handled by Neo4j in real time. The slow side is clustering, and it depends on the size of the dataset. Clustering of 1-day data will not take more than one hour for companies' feeds. For longer periods of time, it can be run overnight or even several days.

NFR2: Visualization.
Visualization satisfied researchers. However, Neo4j allows using third-party

frameworks for visualization.

NFR3: Learning curve.
Researchers were happy with the learning curve. They managed to start working with the data in the first hour.

Chapter 7

Discussion

Clustering email messages by several features can combine loosely related messages into spam campaigns and find campaigns that likely to have a common sender. Possible suitable features to perform such clustering are: subjects, bodies, embedded links. Clustering by several features can be visualized as a graph, where each cluster of each type and each email are presented as nodes and email belonging to a cluster represented as an edge. Results show that this approach allows to track changes applied to campaigns and get better visibility over the spam landscape.

Developed tool allows to improve analysis of spam campaigns and get more thorough information about spam actors. In practical sense it means that in specific cases researcher will spend less time and effort to achieve same results. Also it allows to perform more complex tasks on the campaigns, resulting in better understanding of the biggest malware distribution mechanism.

Because of spam being so popular among threat actors, many related researches has been held. "Spam campaign detection, analysis, and investigation" [9], "Clustering Spam Campaigns with Fuzzy Hashing" [3], "Mining Spam Email to Identify Common Origins for Forensic Application"[45], and "Fast and Effective Clustering of Spam Emails based on Structural Similarity" [37] laid a background for this work and inspired to combine approaches together.

This thesis takes vague idea using graphs to correlate spam, introduced in "Mining Spam Email to Identify Common Origins for Forensic Application"[45], and, with use of other features tested in other researches, introduces a way of correlating messages. This research makes the next step from clustering messages by a single feature to clustering messages by several feature and using intersection of clusters as additional information. Although some related works implement more sophisticated and advanced methods for clustering, the main novelty of this work is utilizing different existing approaches, combining them with a graph theory, and presenting as a graph for future use by a human.

In the beginning of the research, expectations were that it can improve some existing methods by covering "blind spots" through adding features. Evaluation of the results shows that intermediate messages play an important role in spam. Sometimes it reveals mistakes or negligent use of resources that allows to assume common actors, sometimes it allows to track what features campaigns change through time, sometimes it can reveal big campaigns that underwent changes during the dispatch and would not be clustered by single feature.

Unfortunately, connecting loosely relating campaigns might lead to connec-

tion between unrelated campaigns that just use common features, for example, short subject like "Re:" or "hi". Frequency of such wrong connections depends on the dataset. However, for tested datasets it was clear in all cases and researchers could easily resolve the problem by removing few nodes from the graph.

Scalability is another issue that had been left out of the focus of the research. One of the possible solutions is to merge clusters created from smaller batches of data. However, the most computationally complex parts are clustering algorithms and they can be replaced without changing the results of the research.

Nevertheless, current proof of concept implementation already has some value for the researchers. It received very good comments and working with real data proved prior knowledge about some campaigns.

This research shows a way to combine clustering algorithms together to get more valuable information from spam. However, many things can be improved. Here are a few possible future research directions:

Improving clustering algorithms
Complexity of utilized algorithms can be reduced by changing way of processing features and by changing clustering algorithms themselves.

Trying other features
Only 3 features had been implemented for the research. Adding features can give more information about relation between campaigns but it also can produce connectivity between unrelated campaigns. Thus features should be added cautiously.

Recipients clustering
Recipients clustering has been introduced in section 4.1.6. It uses similarities between lists of recipients for already found spam campaigns to detect relations between different campaigns. Researching this method might give more valuable information about spam actors.

Researching the graph
This research introduces the graph of email clusters. Next step would be to find optimal and automated ways of working with such graph to extract valuable information without the researcher.

Removing connections between unrelated campaigns can also be another way of improving the results. It might be possible to achieve by calculating the flow between two nodes to determine their connectivity and cutting weakest connections when flow is less than the threshold.

Visualization
Visualization for this research is done by default Neo4j visualization frame-

work. Neo4j has many other visualization frameworks that might suit better, such as KeyLines [42].

Chapter 8

Conclusions

This thesis describes the theory and implementation of a new approach for spam clustering and analysis. Proof of concept software was implemented and tested as part of research for F-Secure, a large Finnish cybersecurity company.

Spam constitutes more than half of all email traffic and it is the main vector of infecting users with malware.[27] Therefore, spam is one of the most important sources of information for security companies.

Spammers use many different techniques to avoid spam filters, which complicates the email analysis. Email subjects and message bodies are dynamically generated, headers with source IP addresses are forged, and the domain names are frequently replaced with new ones.

Existing email analysis tools allow the analyst to group messages together for further analysis. Unfortunately, most of them cluster email by one feature or one set of features, requiring all messages in cluster to match on these features. Therefore, they miss information about how the spam campaign changed through time and how likely it is for different campaigns to have a common sender. Although one source [45] presents an approach for correlating different campaigns together, it utilizes only two features and does not take into account slightly changed feature values. For example, if two subject lines have one different word, they will be processed as unrelated subjects.

To improve spam analysis, we have developed new clustering techniques. The clustering happens in two stages. In the first stage, messages are clustered several times by different features in such way that an independent set of clusters is generated for each feature. In second stage, emails clusters are presented as a graph where each email and each cluster are vertices and the relation of an email to a cluster is expressed as an edge. Such representation allows us to easily see intersections of clusters and to track changes made to the campaign. These relations are visualized with a graph database and visualization tool, Neo4j, that provides graphical and textual query capabilities to browse and search the graph.

To evaluate the approach, it was tested on real data provided by F-Secure. The developed tool managed to prove its usefulness by helping to find and visualize complex spam campaigns. One of the demonstrated abilities is being able to find spam campaigns that changed all their features over time. Such messages would be impossible to cluster without following the graph through intermediate messages that connect them. However, after the connection is found, a human can easily verify that these messages are part of the same campaign. The developed tool allows analysts to find such campaigns automatically.

To evaluate usability of the tool, we interviewed security analysts, who agreed that suggested tool gives valuable insight and should be integrated into the internal analysis systems of F-Secure.

One interesting problem to solve was the feature selection, i.e. which email features can be used to find connections between spam campaigns. Besides finding textually similar campaigns, our approach is able to find campaigns that shared some resources at some point, for example, domain name in URL. This can be used to combine campaigns by common sender. The work on feature selection could still be continued.

The most important remaining open question is how the search for valuable information can be reliably automated. One of the use cases for our techniques is to provide daily statistics about the incoming threat-intelligence data. Unfortunately, accidentally shared features sometimes cause false clustering and we currently rely on the human expect to confirm the tool findings. New graph algorithms could possibly mitigate false clustering.

In summary, the tool developed in this thesis project can also provide valuable information for governmental organizations to identify major spammers and for security companies to get a better understanding of the spam landscape. Based on state-of-the-art data mining techniques, it finds non-obvious connections between messages and reduces the amount of data that needs to be shown to the human analyst. With a rich search engine and usable visualization, it provides an easy way to work with spam campaigns instead of single messages.

Appendix A

Attachment name	Graph path
Delivery-Details.zip	2
UPS-Package-XXXXX.zip	2
Item-Delivery-Details-XXXXX.zip	2
Ground-Label-XXXXX.zip	2
UPS-Label-XXXXX.zip	2
UPS-Receipt-XXXXX.zip	2
Delivery-Details.zip	2
Delivery-Details.zip	2
Delivery-Details.zip	2
Delivery-Details-XXXXX.zip	4
Delivery-Details-XXXXX.zip	4
UPS-Delivery-Details-XXXXX.zip	4
UPS-Delivery-Details-XXXXX.zip	4
Item-Delivery-Details-XXXXX.zip	4
UPS-Label-XXXXX.zip	4
UPS-Receipt-XXXXX.zip	4
Delivery-Receipt-XXXXX.zip	4
Undelivered-Package-XXXXX.zip	4
Item-Delivery-Details-XXXXX.zip	4
Ground-Label-XXXXX.zip	4
Delivery-Details-XXXXX.zip	4
UPS-Receipt-XXXXX.zip	4
Ground-Label-XXXXX.zip	4
Undelivered-Package-XXXXX.zip	4
UPS-Delivery-XXXXX.zip	4
UPS-Receipt-XXXXX.zip	4
Delivery-Details.zip	4
UPS-Delivery-XXXXX.zip	4
Delivery-Details.zip	4
Delivery-Details.zip	4
UPS-Package-XXXXX.zip	4
UPS-Parcel-ID-XXXXX.zip	4
Delivery-Receipt-XXXXX.zip	6
Delivery-Details-XXXXX.zip	6
Delivery-Details-XXXXX.zip	6
Delivery-Details-XXXXX.zip	6
Delivery-Receipt-XXXXX.zip	6
Undelivered-Package-XXXXX.zip	6

UPS-Label-XXXXX.zip	6
Undelivered-Package-XXXXX.zip	6
Item-Delivery-Details-XXXXX.zip	6
Delivery-Receipt-XXXXX.zip	6
Undelivered-Parcel-ID-XXXXX.zip	6
Ground-Label-XXXXX.zip	6
Undelivered-Package-XXXXX.zip	6
Undelivered-Parcel-ID-XXXXX.zip	6
Ground-Label-XXXXX.zip	6
Ground-Label-XXXXX.zip	6
Undelivered-Parcel-ID-XXXXX.zip	6
Item-Delivery-Details-XXXXX.zip	6
UPS-Delivery-Details-XXXXX.zip	6
Item-Delivery-Details-XXXXX.zip	6
UPS-Label-XXXXX.zip	6
UPS-Delivery-XXXXX.zip	6
Ground-Label-XXXXX.zip	6
Undelivered-Parcel-ID-XXXXX.zip	6
Undelivered-Package-XXXXX.zip	6
Delivery-Receipt-XXXXX.zip	6
Delivery-Receipt-XXXXX.zip	6
Delivery-Details-XXXXX.zip	6
Undelivered-Parcel-ID-XXXXX.zip	6
Delivery-Receipt-XXXXX.zip	6
Delivery-Details-XXXXX.zip	6
Ground-Label-XXXXX.zip	6
Ground-Label-XXXXX.zip	6
Undelivered-Package-XXXXX.zip	6
Delivery-Receipt-XXXXX.zip	6
UPS-Parcel-ID-XXXXX.zip	6
UPS-Parcel-ID-XXXXX.zip	6
Ground-Label-XXXXX.zip	6
Ground-Label-XXXXX.zip	6
UPS-Delivery-XXXXX.zip	6
Delivery-Details-XXXXX.zip	6
Delivery-Details-XXXXX.zip	6
Undelivered-Parcel-ID-XXXXX.zip	6
Undelivered-Parcel-ID-XXXXX.zip	6
Ground-Label-XXXXX.zip	6
Undelivered-Package-XXXXX.zip	6
Ground-Label-XXXXX.zip	6
Delivery-Details-XXXXX.zip	6

Ground-Label-XXXXX.zip	6
Item-Delivery-Details-XXXXX.zip	6
UPS-Delivery-XXXXX.zip	6
Delivery-Details.zip	6
Delivery-Details.zip	6
UPS-Receipt-XXXXX.zip	6
UPS-Delivery-XXXXX.zip	6
Delivery-Details.zip	6
Delivery-Details.zip	6
UPS-Receipt-XXXXX.zip	6
UPS-Package-XXXXX.zip	6
Delivery-Details.zip	6
Delivery-Details.zip	6
UPS-Package-XXXXX.zip	6
Delivery-Details.zip	6
Delivery-Details.zip	6
Delivery-Details.zip	6
UPS-Delivery-XXXXX.zip	6
UPS-Delivery-Details-XXXXX.zip	6
UPS-Parcel-ID-XXXXX.zip	6
UPS-Label-XXXXX.zip	6
UPS-Delivery-XXXXX.zip	6
UPS-Label-XXXXX.zip	6
UPS-Parcel-ID-XXXXX.zip	6
UPS-Parcel-ID-XXXXX.zip	6
Delivery-Details.zip	6
Delivery-Details.zip	6
Delivery-Details.zip	6
Delivery-Receipt-XXXXX.zip	8
Item-Delivery-Details-XXXXX.zip	8
UPS-Label-XXXXX.zip	8
UPS-Delivery-Details-XXXXX.zip	8
Item-Delivery-Details-XXXXX.zip	8
Undelivered-Package-XXXXX.zip	8
Delivery-Details-XXXXX.zip	8
Undelivered-Package-XXXXX.zip	8
Item-Delivery-Details-XXXXX.zip	8
Undelivered-Parcel-ID-XXXXX.zip	8
Item-Delivery-Details-XXXXX.zip	8
Undelivered-Package-XXXXX.zip	8
Undelivered-Package-XXXXX.zip	8
Delivery-Details-XXXXX.zip	8

Delivery-Details.zip	8
Delivery-Details.zip	8
Delivery-Details.zip	8
UPS-Label-XXXXX.zip	8
UPS-Delivery-Details-XXXXX.zip	8
UPS-Package-XXXXX.zip	8
Delivery-Details.zip	8
Delivery-Details.zip	8
Delivery-Details.zip	8
Delivery-Details.zip	8
Delivery-Details.zip	8
FedEx-Parcel-ID-XXXXXXX.zip	8
FedEx-Label-ID-XXXXXXX.zip	8
FedEx-Parcel-ID-XXXXXXX.zip	8
FedEx-Delivery-ID-XXXXXXX.zip	8
FedEx-Package-ID-XXXXXXX.zip	8
FedEx-Label-ID-XXXXXXX.zip	8
FedEx-Label-ID-XXXXXXX.zip	8
FedEx-Label-ID-XXXXXXX.zip	8
UPS-Delivery-XXXXX.zip	8
FedEx-Delivery-ID-XXXXXXX.zip	8
Delivery-Details.zip	8
Delivery-Details.zip	8
Delivery-Details.zip	8
Delivery-Details.zip	8
Delivery-Details.zip	8
Delivery-Details.zip	8
Delivery-Details.zip	8
Delivery-Details.zip	8
Delivery-Details.zip	8
Delivery-Details.zip	10
Delivery-Details.zip	10
Delivery-Details.zip	10
Delivery-Details.zip	10
Delivery-Details.zip	10
Delivery-Details.zip	10
Delivery-Details.zip	10
Delivery-Details.zip	10
Delivery-Details.zip	10
Delivery-Details.zip	10
Delivery-Details.zip	10

Delivery-Details.zip	10
Delivery-Details.zip	10
Delivery-Details.zip	10
Delivery-Details.zip	10
s.erhart_XXXXX.zip	10
warez_tear.zip	10
NFE-Compra-Realizada-com-sucesso.zip	10
NFE-Compra-Realizada-com-sucesso.zip	10
NFE-Compra-Realizada-com-sucesso.zip	10
NFE-Compra-Realizada-com-sucesso.zip	10
NFE-Compra-Realizada-com-sucesso.zip	10
NotaFisca-XXXXX.zip	10
NotaFisca-XXXXX.zip	10
NotaFisca-XXXXX.zip	10
NF-Fiscal-XXXXX.zip	10
nota fical do pagamento_XXXXX (2).zip	10
UPS-Parcel-ID-XXXXX.zip	10
Delivery-Details.zip	10

Table 2: Extracted attachments from campaign distributing malware pretending to be a delivery service (identificators were replaced with "X")

Generated text
Are you ready to amaze your gf today?
Do you want to gratify your woman at night?
Do you want to impress your lover this night?
Are you ready to please your female partner at night?
Do you wish to please your lover at night?
Do you desire to please your lady this night?
Do you wish to impress your lover every night?
Do you desire to amaze your girlfriend this night?
Do you want to see her satisfied tonight?
Do you want to impress your wife today?
Are you ready to surprise your loved one this night?
Do you desire to see her pleased tonight?
Are you ready to amaze your girl at night?
Do you want to impress your loved one this night?
Do you desire to amaze your lady today?
Do you want to surprise your female partner at night?
Are you ready to satisfy your wife tonight?
Do you desire to amaze your babe tonight?
Do you want to please your lover tonight?
Do you wish to impress your woman this night?

Table 1: Example of synonym swapping generated subjects

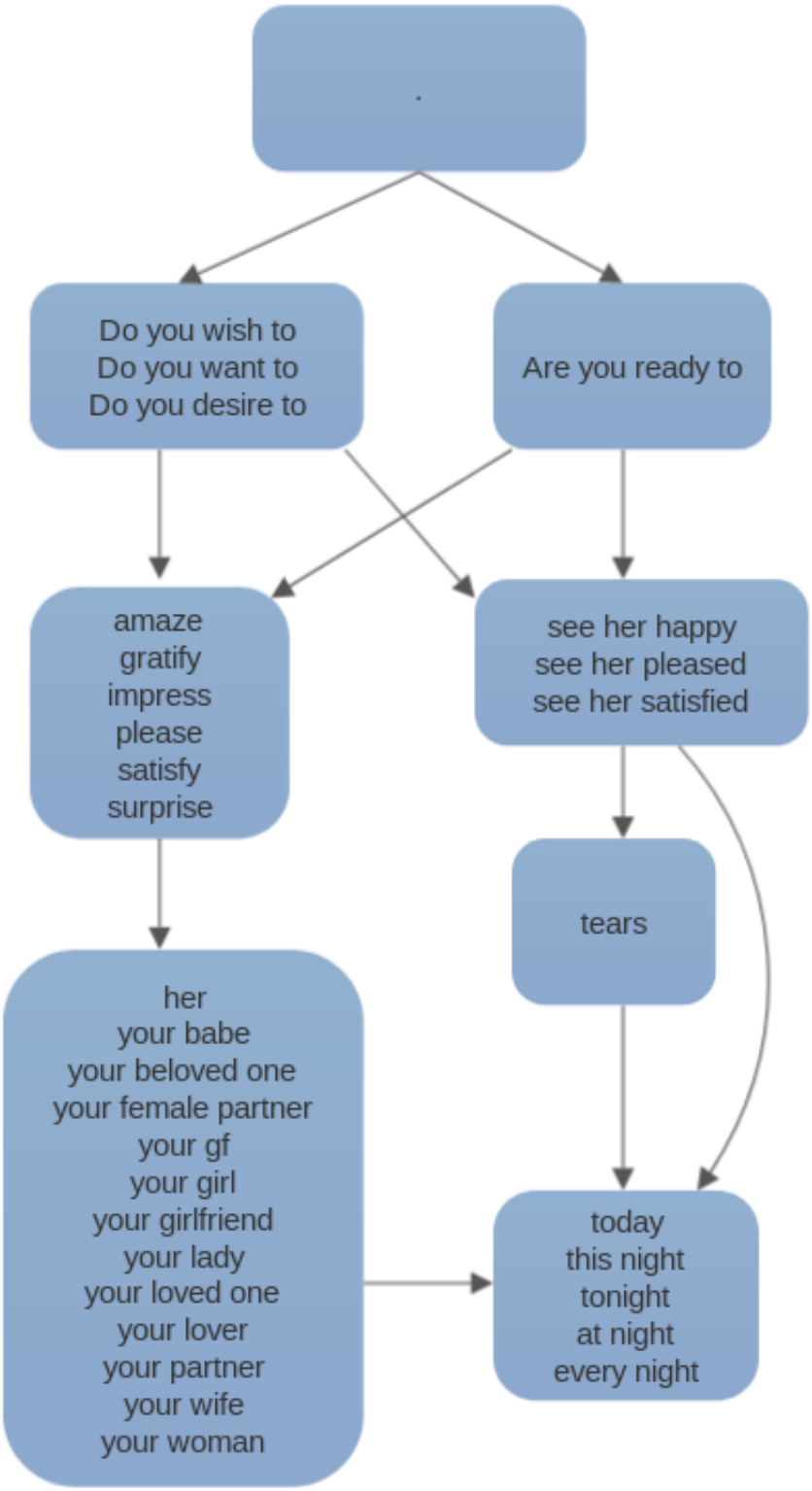

Figure 1: Graph representation of the synonym table extracted from a real-world dataset

Bibliography

[1] Ion Androutsopoulos et al. "An experimental comparison of naive Bayesian and keyword-based anti-spam filtering with personal e-mail messages". In: *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*. ACM. 2000, pp. 160–167.

[2] Nathaniel S Borenstein and Ned Freed. "Multipurpose internet mail extensions (MIME) part two: Media types". In: (1996).

[3] Jianxing Chen et al. "Clustering spam campaigns with fuzzy hashing". In: *Proceedings of the AINTEC 2014 on Asian Internet Engineering Conference*. ACM. 2014, p. 66.

[4] M Crispin. *RFC 3501: Internet Message Access Protocol–Version 4rev1 (2003)*.

[5] Dave Crocker, Tony Hansen, and Murray Kucherawy. *DomainKeys Identified Mail (DKIM) Signatures*. Tech. rep. 2011.

[6] Ernesto Damiani et al. "An Open Digest-based Technique for Spam Detection." In: *ISCA PDCS* 2004 (2004), pp. 559–564.

[7] William HE Day and Herbert Edelsbrunner. "Efficient algorithms for agglomerative hierarchical clustering methods". In: *Journal of classification* 1.1 (1984), pp. 7–24.

[8] Fan Deng, Stefan Siersdorfer, and Sergej Zerr. "Efficient jaccard-based diversity analysis of large document collections". In: *Proceedings of the 21st ACM international conference on Information and knowledge management*. ACM. 2012, pp. 1402–1411.

[9] Son Dinh et al. "Spam campaign detection, analysis, and investigation". In: *Digital Investigation* 12 (2015), S12–S21.

[10] Martin Ester et al. "A density-based algorithm for discovering clusters in large spatial databases with noise." In: *Kdd*. Vol. 96. 34. 1996, pp. 226–231.

[11] Taras Fetsyuk. *Spamassassin Wiki*. `https://wiki.apache.org/spamassassin/`. (Accessed on 14/04/2018). May 2018.

[12] N Freed and J Klensin. *RFC-4288: Media Type Specifications and Registration Procedures*. 2005.

[13] Ned Freed and Nathaniel Borenstein. *Multipurpose internet mail extensions (MIME) part one: Format of internet message bodies*. Tech. rep. 1996.

[14] Wael H Gomaa and Aly A Fahmy. "A survey of text similarity approaches". In: *International Journal of Computer Applications* 68.13 (2013).

[15] Jiawei Han et al. "Mining frequent patterns without candidate generation: A frequent-pattern tree approach". In: *Data mining and knowledge discovery* 8.1 (2004), pp. 53–87.

[16] Florian Holzschuher and René Peinl. "Performance of graph query languages: comparison of cypher, gremlin and native access in Neo4j". In: *Proceedings of the Joint EDBT/ICDT 2013 Workshops*. ACM. 2013, pp. 195–204.

[17] Mick Johnson. *DSPAM Project Homepage*. 2011. URL: `http://dspam.sourceforge.net/` (visited on 03/11/2018).

[18] Salim Jouili and Valentin Vansteenberghe. "An empirical comparison of graph databases". In: *Social Computing (SocialCom), 2013 International Conference on*. IEEE. 2013, pp. 708–715.

[19] Kamran Khan et al. "DBSCAN: Past, present and future". In: *Applications of Digital Information and Web Technologies (ICADIWT), 2014 Fifth International Conference on the*. IEEE. 2014, pp. 232–238.

[20] Frank Klawonn and Frank Höppner. "What is fuzzy about fuzzy clustering? Understanding and improving the concept of the fuzzifier". In: *International symposium on intelligent data analysis*. Springer. 2003, pp. 254–264.

[21] J. Klensin. *Simple Mail Transfer Protocol*. RFC 5321. IETF, Oct. 2008, pp. 1–95. URL: `https://tools.ietf.org/html/rfc5321`.

[22] John Klensin. "RFC2821: Simple mail transfer protocol, 2001". In: *AT&T Laboratories* ().

[23] Gad M Landau and Uzi Vishkin. "Fast parallel and serial approximate string matching". In: *Journal of algorithms* 10.2 (1989), pp. 157–169.

[24] John Levine. *DNS blacklists and whitelists*. Tech. rep. 2010. URL: `https://tools.ietf.org/html/rfc5782`.

[25] Keith Moore. "RFC 2047: MIME (Multipurpose Internet Mail Extensions) part three: Message header extensions for non-ASCII text". In: *ISI, Marina Del Ray, CA (November 1996)* (1996).

[26] John Myers and Marshal Rose. *Post office protocol-version 3*. Tech. rep. 1996.

[27] Ben Nahorney. *Email Threats 2017. An ISTR Special Report*. Oct. 2017. URL: `https://www.symantec.com/content/dam/symantec/docs/security-center/white-papers/istr-email-threats-2017-en.pdf`.

[28] *numpy.ndarray — NumPy v1.14 Manual*. Jan. 2018. URL: `https://docs.scipy.org/doc/numpy-1.14.0/reference/generated/numpy.ndarray.html` (visited on 04/23/2018).

[29] Abhinav Pathak et al. "Botnet spam campaigns can be long lasting: evidence, implications, and analysis". In: *ACM SIGMETRICS Performance Evaluation Review*. Vol. 37. 1. ACM. 2009, pp. 13–24.

[30] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[31] Andreas Pitsillidis et al. “Taster’s choice: a comparative analysis of spam feeds”. In: *Proceedings of the 2012 Internet Measurement Conference*. ACM. 2012, pp. 427–440.

[32] Vipul Ved Prakash. *Vipul’s Razor*. 2018. URL: `http://razor.sourceforge.net/` (visited on 06/06/2018).

[33] P. Resnick. *Internet Message Format*. RFC 5322. IETF, Oct. 2008, pp. 1–57. URL: `https://tools.ietf.org/html/rfc5322`.

[34] P Resnick. “RFC 2822: Internet Message Format, 2001”. In: *URL http://www.ietf. org/rfc/rfc2822. txt* (2008).

[35] Leonard Richardson. *Beautiful soup documentation*. 2007.

[36] Andrew Rosenberg and Julia Hirschberg. “V-measure: A conditional entropy-based external cluster evaluation measure”. In: *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning*. 2007.

[37] Mina Sheikhalishahi et al. “Fast and effective clustering of spam emails based on structural similarity”. In: *International Symposium on Foundations and Practice of Security*. Springer. 2015, pp. 195–211.

[38] Scott S. Smith. *Internet Crime Report*. 2016. URL: `https://pdf.ic3.gov/2016_IC3Report.pdf`.

[39] Ian Sommerville. “Software requirements”. In: *Software Engineering* (2004), pp. 117–141.

[40] Spammer-X Spammer-X. *Inside the SPAM Cartel: By Spammer-X*. Elsevier, 2004.

[41] *SpamScope/mail-parser: Tokenizer for raw mails*. Mar. 2018. URL: `https://github.com/SpamScope/mail-parser` (visited on 04/05/2018).

[42] KeyLines team. *KeyLines White Paper*. Mar. 2018. URL: `https://cambridge-intelligence.com/wp-content/uploads/2018/01/KeyLines-White-Paper.pdf` (visited on 04/12/2018).

[43] Brad Templeton. “Reaction to the DEC Spam of 1978”. In: *availabe at: http://www. templetons. com/brad/spamreact. html(accesed December 2010)* (2003).

[44] Tanguy Urvoy, Thomas Lavergne, and Pascal Filoche. “Tracking Web Spam with Hidden Style Similarity.” In: *AIRWeb*. 2006, pp. 25–31.

[45] Chun Wei et al. “Mining spam email to identify common origins for forensic application”. In: *Proceedings of the 2008 ACM symposium on Applied computing*. ACM. 2008, pp. 1433–1437.

[46] Anders Wiehes. “Comparing anti spam methods”. MA thesis. 2005.

[47] Wikipedia contributors. *Anti-spam techniques*. (Accessed on 12/04/2018). 2018. URL: `https://en.wikipedia.org/wiki/Anti-spam_techniques`.

[48] Meng Wong and Wayne Schlitt. *Sender policy framework (SPF) for authorizing use of domains in e-mail, version 1*. Tech. rep. 2006.