Semi-Automatic Analysis of Large Textfile Datasets for Forensic Investigation

Sæmundur Óskar Haraldsson

Faculty of Industrial Engineering, Mechanical Engineering and Computer Science
University of Iceland
2012

SEMI-AUTOMATIC ANALYSIS OF LARGE TEXTFILE DATASETS FOR FORENSIC INVESTIGATION

Sæmundur Óskar Haraldsson

60 ECTS thesis submitted in partial fulfillment of a
Magister Scientiarum degree in Industrial Engineering

Advisors
Tómas Philip Rúnarsson
Steinn Guðmundsson

Faculty Representative
Sven Þórarinn Sigurðsson

Faculty of Industrial Engineering, Mechanical Engineering and Computer Science
School of Engineering and Natural Sciences
University of Iceland
Reykjavik, 06 2012

Semi-Automatic Analysis of Large Textfile Datasets for Forensic Investigation
Semi-Automatic Analysis of Large Textfile Datasets
60 ECTS thesis submitted in partial fulfillment of a M.Sc. degree in Industrial Engineering

Faculty of Industrial Engineering, Mechanical Engineering and Computer Science
School of Engineering and Natural Sciences
University of Iceland
Hjarðarhagi 2-6
107, Reykjavik, Reykjavik
Iceland

Telephone: 525 4000

Bibliographic information:
Sæmundur Óskar Haraldsson, 2012, Semi-Automatic Analysis of Large Textfile Datasets for Forensic Investigation, M.Sc. thesis, Faculty of Industrial Engineering, Mechanical Engineering and Computer Science, University of Iceland.

Printing: Háskólaprent, Fálkagata 2, 107 Reykjavík
Reykjavik, Iceland, 06 2012

To my family

Abstract

This thesis establishes a framework for investigating digital communication between individuals in a semi-automatic manner. The purpose of this framework is to give forensics investigators a better oversight over the vast amount of communication data commonly confiscated in a crime investigation. The motivation for this work is the vast amount of data collected in Iceland with the fall of their banks. The framework established, termed SALTeD, is composed of 3 main components; data pre-processing and feature selection, cluster analysis, and finally dynamic social network analysis. These components are integrated in a unique manner to give the investigator a semi-automated way of investigating communication networks and discover new and novel patterns of communication. The framework is tested on collected real world data, including the Enron Corp. emails, confiscated by detectives investigating the Enron fraud case.

Útdráttur

Þessi lokaritgerð lýsir aðferðafræði sem beita má við rannsóknir á stafrænum samskiptagögnum milli einstaklinga. Tilgangur þessa ramma er að gefa aðilum réttarrannsókna betra yfirlit yfir það gríðarmikla magn samskiptagagna sem jafnan er safnað fyrir réttarrannsóknir. Kveikjan að þessari vinnu er það magn gagna sem safnað hefur verið á Íslandi frá falli bankanna. Ramminn kallast SALTeD og er samsettur af þremur hlutum, forvinnu gagna og einkennavali,klasagreiningu og að lokum virkri samfélags netgreiningu. Þessum hlutum er fléttað saman á einstakan máta til að aðstoða við rannsókn á samskiptanetum og uppgötva nýstárleg samskiptamynstur. Aðferðafræðin er prófuð á raunverulegum gögnum, þar á meðal vefpóstasafni Enron Corp. sem gert var upptækt við rannsókn á fjármálamisferli stjórnenda þess.

Contents

Acknowledgments

In the time it took to finish this thesis I received help and support from numerous sources and I will possibly forget to mention some. Those I do remember are

> First and foremost my wife, Heiða, for endless support and patience which without I could not have succeeded.
>
> My advisor, prof. Tómas Philip Rúnarsson for excellent guidance and help.
>
> My co-advisor, Steinn Guðmundsson for invaluable counsel on everything, not only my thesis.
>
> The Faculty of Industrial Engineering, Mechanical Engineering and Computer Sciences for financial support in the form of a grant and employment.
>
> My parents and in-laws mainly for financial support but also for taking care of the kids when the writing and research demanded too much time of me.
>
> Everyone with whom I shared the office at Tæknigarður for the companionship through our joint suffering of seemingly endless projects and teaching.

and everybody else for everything else.

Thank you all and sorry if I forgot to mention you specifically.

1. Introduction

Ever since the Internet became publicly available around 1990, digital traffic has grown immensely and it has been estimated that the growth rate through the late 1990's was 100% per year [12]. It can be safely assumed that the rate of communication through the Internet, with e.g. email, instant messaging, and social media sites, has surpassed other forms of communication such as telephone and/or written mail, humorously nicknamed snail mail, especially with the introduction of smart devices such as smart phones, tablet computers etc. Even voice communication with added video through the Internet is now getting more and more popular with Skype and other similar programs. A consequence of this development is that much more detailed information of the interaction between two or more individuals is available and in larger quantities than before. This helps law enforcement agencies tremendously when investigating crimes, since now they can collect vast amount of data already available from servers, hard drives and smart devices, including personal computers. The data can be in various forms, not only text or communication patterns, but also location information from cell cites, global positioning systems (GPS) integrated into many smart devices and the Internet protocol (IP) number, although that last one can be masked by various means. All this comes as an addition to the proven traditional methods like wire taps and manual surveillance.

Many methods, techniques and even programs are already available for dealing with the data that is collected for forensic purposes. This thesis presents a digital framework that can aid forensic detectives in analyzing emails, memos, short message service (sms) and other text communication data on digital form. The framework is called “Semi-automatic Analysis of Large Textfile Datasets“ or SALTeD for short. It has the potential to divide datasets into categories according to their topic and reveal any odd or suspicious communication within that topic by presenting an animation of its development over time. SALTeD is a promising tool to assist future forensic investigations where vast amount of digital communication text data is accessible.

1.1. Motivation

In the wake of September 2008, when the three major commercial banks in Iceland collapsed, and through the ongoing economical recession, multiple criminal investigations have been launched centering on the financial industry. The majority of the confiscated data in those investigations are in digital form, including large collections of emails.[1] The investigative method utilized by the Icelandic investigators initially involved reading through each digital document. Obviously those kind of methods are thorough but take up a good amount of time and manpower. What is not as obvious is that they can also overlook great supportive evidence such as dynamic communication patterns and connections between the parties involved.

It was this problem that spurred the development of SALTeD, a framework or set of methods that would give the investigator better oversight of the data and would speed up the investigation considerably by minimizing the set of emails that he had to read manually.

Since the Internet has been publicly open for over two decades it is not surprising that this field of research is in no way unexplored. For the most part it has been aimed at protecting computers and networks from spam and malicious email by filtering the good from bad. Most often this is accomplished with clustering by subject matter [53] or by using headers and the content type [83]. With the latter it is also possible to distinguish between two different spam campaigns, not only to find out how many campaigns are active but also to trace multiple campaigns to the same source. Some studies have been focused on distinguishing between authors of different documents for the purpose of detecting plagiarism [20] while others have ignored the content of the emails while mapping out the social network from the header information [74].

The aforementioned methods and studies have been collected in at least two packages for forensic purposes; the Email Mining Toolkit (EMT) [9, 73, 75, 76] and the Integrated E-mail Forensic Analysis Framework (IEFAF) [32]. They compliment each other quite nicely, what one lacks the other makes up for, on most levels. When automatically analyzing a large quantity of objects for categorization or some kind of sorting, a set of characterizing details called features must be selected to represent them. This process is usually called feature extraction. For text documents the obvious features are individual words, combinations of words and linguistic features. The two packages are both designed with the English language in mind to make the feature extraction as simple as possible and thus speed up the analysis.

[1] Quantities measured in terabytes

1.2. Contribution

The main contribution of this thesis is a framework which integrates three main components: feature selection, clustering and dynamic network visualization. From the input data SALTeD selects relevant information in the form of features. The features are then used as an input to a clustering algorithm. The data is then presented as a visual simulation of a network with the open source software Gephi [3]. In particular the framework SALTeD combines:

- a simple yet effective document threshold method (DTM) and some intuitive rules to select and manipulate features.
- the K-means clustering algorithm and a version of CURE [31] to partition the datasets into topics.
- and the open source program Gephi [3] to visualize and analyze the communication network presented by the datasets.

The Newsgroup data set as well as the Enron email data set are used for testing the framework. The Enron data is a good example of real world data collected for the purpose of investigating financial crime.

1.3. Overview

The organization of this thesis is as follows: Chapter 2 is a short literature review of the underlying methods used in SALTeD, namely feature extraction and feature processing, clustering, and visualization. Chapter 3 will then explain how those methods are interwoven to make SALTeD, give some instructions on how to use it and interpret the output. A case study and experiments that were conducted on the Enron corpus[2] [47] and the 20 Newsgroup Data[3] are presented in chapter 4. Finally, in chapter 5 the results of the experiments are discussed and in chapter 6 conclusions are drawn and further development of SALTeD or its integration with other frameworks and toolkits will be suggested.

[2]available at `http://www-2.cs.cmu.edu/~enron/`
[3]available at `http://people.csail.mit.edu/jrennie/20Newsgroups/`

2. Background

This chapter will describe clustering, feature selection and network visualization in sufficient detail for the reader to be able to understand better the framework SALTeD described in the next chapter.

Feature selection, where the representing characteristics of objects are chosen, and clustering are both optimization problems. Feature selection minimizes the number of features while maximizing information and clustering minimizes or maximizes various distance attributes defined by the algorithm. These two optimization problems are highly dependent on each other. Firstly, to cluster efficiently and accurately it is of major importance to select appropriate features. Secondly, to evaluate the selected features it is usually best to compare them with the applied clustering method [54] or some pre-classification of training data [56]. Methods have also been developed to select features independently for unsupervised learning [23].

2.1. Clustering

Clustering is one of the primary tools for analyzing data sets in their entirety as opposed to analyzing each object in the set individually. It can reveal engaging distributions in the data and uncover hidden or unknown groups. It is the most commonly applied unsupervised learning method [59].

The task is to divide a set of n points in some space into k clusters, otherwise known as subsets. This is accomplished with minimal prior knowledge of which point belongs to which cluster or what kind of attributes will characterize each cluster, hence the term unsupervised learning. These points can represent objects of any type such as emails, medical reports, pictures, genome patterns or communication data to name a few. Therefore, various scientific disciplines such as image processing [67], pattern recognition [41], and biology [6] apply clustering on a regular basis.

A simple clustering problem is shown in figure 2.1. The nine points in two dimensional space can be clustered into two or even three clusters by visual inspection. Real world problems are rarely this simple but this figure serves well to explain the

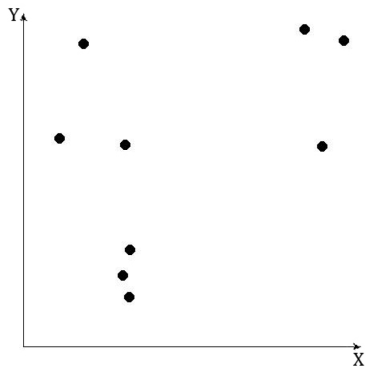


Figure 2.1: Randomly displaced data points.

essentials of clustering. If each point represents an object of some sort then the individual clusters produced should contain objects that are similar in structure and dissimilar to objects in other clusters.

The similarity is calculated with various methods but usually by some kind of a distance measure where the location of each object in the space is defined by numerical values of the selected features. The dimensionality of the space corresponds to the number of features. Here it is assumed that the features are real valued m dimensional vectors, i.e. for any object i its features are represented by vector $v_i \in \mathbf{R}^m$.

A straightforward approach for distance measurement, and also the most commonly used is the Euclidean distance [39] where the distance between objects i and j is defined as

$$D(v_i, v_j) = \sqrt{\sum_{k=1}^{m} (v_{i,k} - v_{j,k})^2} \tag{2.1}$$

A drawback of the Euclidean method is that if one or more features are dominant in size they can distort the outcome. To counteract that, one should normalize the feature vectors or use some other methods of weighing each component.

There are a number of different techniques available for clustering. They can be categorized by subtle or not so subtle differences in handling the given data. Every technique produces different clusters, with minuscule to substantial variations, since the outcome of every clustering problem is highly dependent on several variables such as the distance measurement, feature selection, and of course the choice of algorithm. Those variables are in turn often correlated, making the choice of the right technique that much more complicated. For the simplification, this thesis uses the categorization of clustering approaches from Jain and Dubes [40] and divides them into two main groups as seen in figure 2.2,

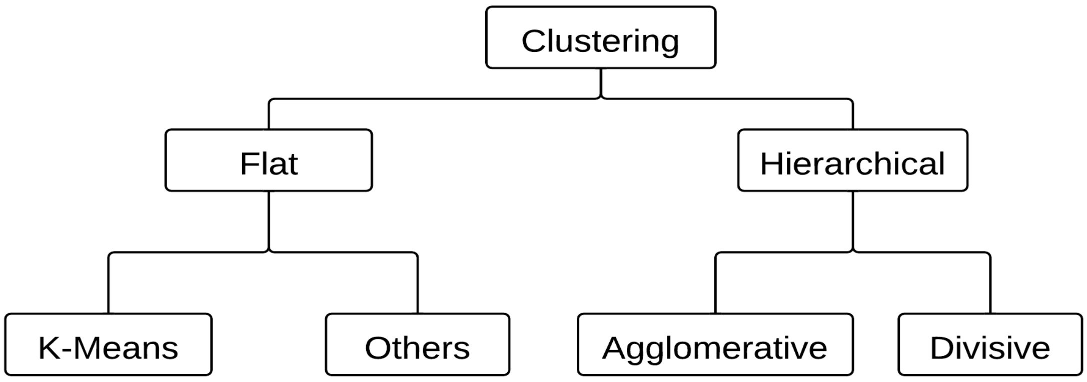


Figure 2.2: A taxonomy of clustering techniques.

- Flat clustering algorithms with *K-means* being the most popular among many [58].
- Hierarchical clustering algorithms which are further split into agglomerative and divisive.

Flat clustering algorithms are often referred to as partitional algorithms because their output is a single partition of the data where each cluster is usually independent of another and all clusters are on the same level. Figure 2.3 demonstrates an example of how the points in figure 2.1 could be clustered into three clusters by a flat clustering algorithm. Most flat clustering algorithms are simple to implement and have short running times due to their simplicity but a disadvantage is that the number of clusters k must usually be specified beforehand. This drawback demands a priori knowledge of the data or the algorithm must be executed multiple times to find the optimal number of clusters under some conditions. These conditions involve some kind of a quality measure of the clustering outcome. A common approach for determining this is by using the Akaike Information Criterion (AIC) [59], which takes the form

$$K = \underset{K}{\operatorname{argmin}}[\mathrm{RSS}(K) + \lambda K] \tag{2.2}$$

for K-means with $\lambda = 2m$, double the number of features, commonly used. K is the number of clusters and RSS is the total squared distance of every point from a specified reference point in its assigned cluster. The reference point is the arithmetic mean of all points in the cluster and is referred to as the cluster centroid. For centroid μ_i in cluster i the RSS is:

$$\mathrm{RSS}_i = \sum_{j \in cluster_i} D^2(v_j, \mu_i) \tag{2.3}$$

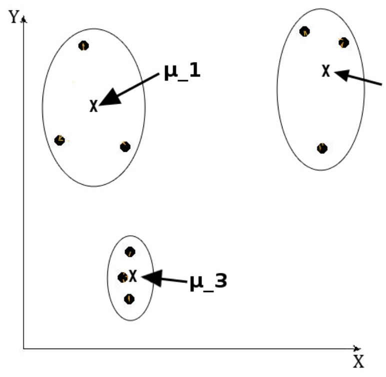


Figure 2.3: Flat clustering. The cluster centroids are represented by μ_1, μ_2 and μ_3

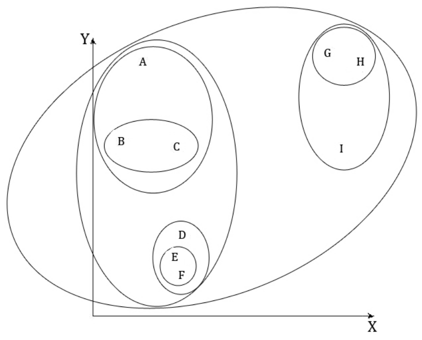
and then for the entire set:

$$\text{RSS} = \sum_{i=1}^{K} \text{RSS}_i \tag{2.4}$$

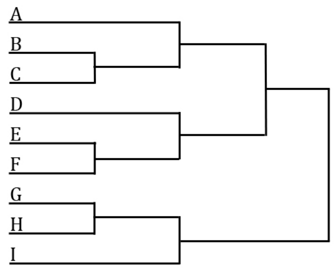
Datasets for real world applications are almost always very large and determining K from (2.2) can therefore result in considerably long running time. One way of determining the number of clusters beforehand without having to cluster the entire dataset more than once is by bootstrapping [63]. In short, it is a statistical method of estimating parameters of an underlying distribution in the data by randomly resampling it repeatedly and estimating the parameters for each sample set. Since the computational complexity is usually more than $O(n)$, by only taking e.g. half the set and computing twice is going to execute faster than with the whole set once. Repeatedly estimating parameters with a randomly selected subset of the data and cumulatively calculating their means should eventually result in convergence towards their actual value, given that the sampling size and number of iterations are large enough. In the case of clustering having the samples size in the right range and calculating K from (2.2) for each sample could give the right estimate of K for the whole dataset without exhausting the computational capacity.

As mentioned above *K-means* clustering is the most commonly used method and a simple version of it will be described in section 2.1.1. Other partitional algorithms apply graph theory [85], mixture resolving and mode seeking methods [21, 40, 60] to name a few. Those algorithms were not utilized during the development of SALTeD and hence will not be discussed further.

Hierarchical Algorithms are named so after the structures they produce, which are hierarchical trees of clusters and sub-clusters. These methods can provide much more insightful and detailed analysis of the data as figure 2.4 depicts where the hierarchy can be represented as a tree of connections. A scale could be placed along the horizontal axis of figure 2.4(b) with added information such as the value of a fitness function of the clustering to decide how many and what clusters are optimal. Due to these additional information the hierarchical algorithms can provide, they require more computational power than the flat clustering algorithms, both regarding speed and memory. The *K-means* algorithm as an example has a time complexity of $O(n)$ for one iteration [39] while the most commonly used Hierarchical algorithms have at least $0(n^2)$ complexity[59].



(a) Visual clustering



(b) A tree showing the connections

Figure 2.4: Hierarchical clustering.

Hierarchical agglomerative clustering algorithms (HAC) are often referred to as bottom-up algorithms as they start of with every data point as its own cluster and then, by various rules, merge pairs of clusters until either some stop criteria is met or all the points belong to the same cluster. The other type of hierarchical clustering algorithms is divisive, also known as top-down algorithms and work in the opposite direction of the HAC algorithms. No divisive clustering algorithm was considered for SALTeD since they have even more complexity than HAC and the outcome is highly dependent on the rules of splitting [36].

2.1.1. K-means

The *K-means* algorithm dates back to the nineteen sixties [2, 58] and has been applied in and adapted for various problems. All *K-means* variations have the common objective of dividing n points into K clusters. Most of them also define the space as $\mathbf{R}^m$ [45], the distant measure as Euclidean and the mean squared distance for each point to the nearest cluster center as the fitness function.

Algorithm 2.1 is the *K-means* as used in this thesis. It starts by choosing, with some

Algorithm 2.1 The K-means algorithm

```
1: choose $k$ $\mu$
2: repeat
3:     for $i = 1$ to $n$ do
4:         for $j = 1$ to $k$ do
5:             if $D(v_i, \mu_j) < D(v_i,\text{current } \mu)$ then
6:                 assign $v_i$ to $\mu_j$
7:                 adjust $\mu_j$ and (now) former $\mu$
8:             else
9:                 continue loop
10:            end if
11:        end for
12:    end for
13: until largest adjustment $< \alpha$
```

heuristic, K cluster centers and then for each pair of center μ_j and data point v_i it compares the distance between them with the distance between the data point and the center to which it is currently assigned. If it is smaller, the assignment of the data point is changed and centers are adjusted to account for the switch. This is usually repeated until the adjustment is less than some small value α. This produces K groups represented by the mean of each group and is thus called *K-means.*

With many benefits such as simplicity in implementation and computational efficiency *K-means* also has its limits. They include the uncertainty of convergence to a global minimum, the shape of the clusters and the need for a priori knowledge of the number of clusters. Convergence to a local minimum is guaranteed [41] but only rare circumstances provide the global minimum. With Euclidean distance, *K-means* produces hyper spherically shaped clusters [77] which usually do not fit well with the data or deal effectively with outliers and variably sized clusters. Counteracting the drawbacks involves a trade-off such as giving up simplicity and/or demanding more computational capacity. One must therefore weigh and measure each individual task against a few alternative clustering techniques for an example various *K-means* methods and *CURE.*

2.1.2. CURE

Since traditional hierarchical clustering algorithms are ill-suited to deal with large databases without immense computational capacity, a number of variations have been developed and proposed. Among those is the CURE algorithm, proposed in 1998 by Guha et. al [31].

What mainly sets CURE apart from traditional HAC algorithms is the way it represents the clusters and how it deals with large databases. Instead of a centroid it employs a constant number of strategically chosen points from each cluster which tighten towards the center by some predetermined fraction α. That way variably sized and shaped clusters can be detected and not just of one shape or minimal size. For large databases the sample space is partitioned into subspaces which is each partially clustered independently to a predetermined threshold before the whole space is merged to form a complete hierarchy.

CURE employs two specialized data structures which are not detailed in this thesis, a priority queue called a heap [27, 51, 82] which stores an entry for each cluster and a binary search tree called k-d tree [8, 70].

Algorithm 2.2 CUREs clustering algorithm

```
1: $Q \leftarrow$ make_Heap
2: $T \leftarrow$ make_Tree
3: **repeat**
4:     $u \leftarrow$ extract_min($Q$)
5:     $v \leftarrow u.closest$
6:     delete($Q, v$)
7:     $w \leftarrow$ merge($u, v$)
8:     delete_rep($T, [u, v]$); insert_rep($T, w$)
9:     $w.closest \leftarrow x$ $\{x$ is an arbitrary cluster from $Q\}$
10:    **for all** $x \in Q$ **do**
11:        **if** $dist(w, x) < dist(w, w.closest)$ **then**
12:            $w.closest \leftarrow x$
13:        **end if**
14:        **if** $x.closest$ is $u$ or $v$ **then**
15:            **if** $dist(x, x.closest) < dist(x, w)$ **then**
16:                $x.closest \leftarrow$ closest_cluster($T, x, dist(x, w)$)
17:            **else**
18:                $x.closest \leftarrow w$
19:            **end if**
20:            relocate($Q, x$)
21:        **else if** $dist(x, x.closest) > dist(x, w)$ **then**
22:            $x.closest \leftarrow w$
23:            relocate($Q, x$)
24:        **end if**
25:    **end for**
26:    insert($Q, w$)
27: **until** $size(Q) < k$
```

Algorithm 2.2 is a rough outline of CUREs clustering algorithm. The first line is the initialization of the priority heap (Q) where the point with the smallest distance to its closest neighbor has the highest priority. The second line is the initialization of the k-d search tree (T) where each point is treated as a separate cluster. CURE needs a predetermined number, k, of desired clusters, contrary to what was stated in the introduction to HAC algorithms above, but this is a trade-off to reduce space complexity to $O(n)$. In lines 4 through 26, the algorithm iterates until the desired number of clusters is reached, always merging the cluster with highest priority from Q with its closest cluster and updating Q and T. The time complexity of the algorithm is at most $O(n^2 \log n)$.

2.2. Features

A *feature* of an object is an attribute of some kind that characterizes it. Objects that belong to the same class will therefore share similar features. As an example a person has multiple features such as height, weight and the number of toes. Collect all imaginable features of a person and you have the means to describe anyone uniquely in detail. The features can be discrete like the number of toes or have continuous values like weight. Normally, a feature pattern for a single object is represented by an array or a vector where each element is a feature [22]. Multiple patterns are combined to form a data matrix which is then used as an input to a grouping algorithm of some kind.

When clustering, it may be tempting to select the whole set of features the dataset has to offer and to think that more detailed description equals better clusters. Two things suggest the opposite, firstly overfitting, a well known term from statistical analysis occurring when a model describes noise instead of the underlying connections; Secondly the curse of dimensionality [5], as the dimensionality of the dataset increases, the volume of the search space expands so much that the number of different distances between objects become astronomical [37]. Although the first problem can be contributed to the second one in many situations, overfitting can also happen in lower dimensions when features irrelevant to the analysis at hand distort the outcome. It is therefore very important to carefully select features from the dataset that can best describe the underlying information. If an objective of some research is to verify a link between a certain age group of people with their occupation, the hair color of the population would tell, by all reasonable assumptions, absolutely nothing of interest.

For any given clustering problem there likely exists an optimal feature set that gives the best results for the given problem. Finding this optimal set would theoretically require an exhaustive search through all possible subsets and is not practical unless the number of features is relatively small [49]. Features can be eliminated by reasonable assumptions such as in the example above, if enough information about the problem is known a priori. Any kind of information can be utilized to eliminate features but doing it manually would make the whole process slow. There is an alternative, to approximate the best set or find the best set given a certain criteria. The means to that alternative depend on the method that will use the features. Feature selection is widely investigated [56] in such areas as machine learning [10, 43, 48], data mining [17, 18, 46] and statistical pattern recognition [42, 61]. Feature selection is also successfully applied in image retrieval [78], genomic analysis [84] and text categorization [52, 65], to name a few applications.

There are many ways to categorize feature selection methods, one is based on the difference in the evaluation criteria [56]. *The filter* model which utilizes the general characteristics of the data without any mining algorithm [18, 33, 55]. *The wrapper* model employs a predetermined mining algorithm to evaluate the selected features [11, 24, 46] and the hybrid model, takes advantage of both methods at different levels of the search [16, 64]. Methods from each of these three categories can be used for most data mining tasks [56] and each has its merits over the others. The wrapper model for example tends to find features better suited to the specific mining method while also being more computationally expensive than the filter model [23]. The hybrid method can either be computationally expensive or find less suited features, depending on the implementation.

Choosing a good set of features is in most cases non-trivial [56]. For clustering, the classes and their definitions are not known so one can not choose features that differentiate between classes. If, on the other hand, it would be known that e.g. a set of news documents should be grouped into four categories, sports, weather, domestic and international, keyword features such as “winner”, “loser”, “first”, “wind”, “country”, should be easy to identify. Selecting features for clustering, however, is much more complicated because it is an unsupervised learning problem (see section 2.1). Then one has to consider the trade-off, computational cost versus better features.

2.3. Network Visualization

“A picture is worth a thousand words" is a popular adage and quite true since a single image can convey a large amount of data. Network visualization is that in practice where an overview of an interconnected community with lines for connections and nodes for each individual or object gives more information than any text describing said community. Network visualization with pictures is static and could be considered as a long exposure photograph of the network. Static network analysis has been around for a long time and the visualization part of it was first drawn by hand [14] where the outcome was simple and highly dependent on the author’s skill and analytical eye. Since then visualization has come a long way through research in multiple fields such as social network analysis [28, 29, 72], statistics [50], biology [57], computer graphics [25] and more.

“A video is worth many pictures" should be added to the original saying to make it more adequate to modern times. *Dynamic network visualization* is an emerging discipline accounting for the changes that the networks undergo with respect to time. In research articles and such, networks still have to be represented on paper but now they are snapshots rather than long exposures and can convey some understanding of the dynamics of the network [62]. With the world wide web open to most people there is now also an efficient way to present to fellow researchers network dynamics with video. There are various options for visualization tools available, both free [3, 7] and commercial [81], that can produce videos and snapshots of networks using multiple algorithms, such as Multi-Component Kamada-Kawai (MCKK) [44], many of which are ad-hoc methods for specific problems. To avoid reinventing the wheel the user can find a suitable algorithm to represent whatever data he wants to draw. SALTeD integrates Gephi which in turn uses various algorithms, all of them adaptations of static algorithms.

Since the mid nineteenth century, work on network layouts has been mostly focused on static graphs. “Force-directed” layouts [30, 44], lower dimensionality projection [35], and layout criteria optimization [19] have been used with good results. But the need for more than just static images for analysis has arisen in later times. Being able to visualize how the network evolves and identify catalysts to certain events is as much or even more important than the connections themselves. What better way to do so than with a virtual surveillance video?

For that purpose many of the algorithms for static graphs can be recycled and used for dynamic visualization as well. They are being implemented into software packages specific to dynamic visualization [7] where they are used for calculating each slice or frame of the videos. What these software packages then add on, is the means to choose the length of each slice and transition from one to the other.

Quite a few software packages [4, 26, 38] have been implemented that extended visualization research by introducing three dimensional viewing, time steps, descriptive statistics, and more. The most promising software to date is Gephi, as it is open source and being developed by a large community of researchers and professionals.

2.3.1. Gephi

Gephi [3] is an open source software collaboration project of The Gephi Consortium. The program supports various file formats for importing data, most of them only for static analysis and only two for dynamic analysis, *graphic exchange XML format* (GEXF), specifically developed with Gephi in mind, and spreadsheet format. It has an easily adaptable graphical user interface (GUI) for convenience.

Dynamic data of the form that network visualization program accept is usually either in a script file that tells them exactly where and when some event should take place or in a comma separated value (csv) files. The script files like GEXF can convey much more and detailed information making it only a formality to load them in and producing an aesthetically pleasing graph. Csv files on the other hand convey as little information as possible, only nodes, arcs and lifetime. It is then up to the user to manipulate the graph once it is loaded to make it pretty. The advantage of the csv file is, as with clustering, that no a priori knowledge is required so any data can be fed to the program and then manipulated to discover relevant information hidden from view.

Gephi, with its GUI, makes all of this relatively easy for any computer savvy individual while still providing transparency and all analytical tools needed for a professional.

2.4. Summary

For most purposes clustering is not possible without features since they define the space in which the objects reside. Feature selection and clustering are thus tightly intertwined so SALTeD has to implement both if one is going to be used. Clustering is the act of grouping similar things together and in order to know if some things are similar one has to know their distinctive features. If those features can be quantized then it is possible to calculate a difference between two objects and thus group them automatically with a clustering algorithm.

If those objects are signals of some sort, like messages, it would be useful to be able to track their paths and find those individuals responsible for them. Network visualization has been implemented in various programs but the one chosen for SALTeD is called Gephi, a simple and user-friendly software.

The next chapter will explain the implementation of aforementioned components into SALTeD.

3. SALTeD

SALTeD is a framework for analysing large text-file datasets and is aimed at aiding forensic investigations. The datasets might be emails, short message service (sms), memos, or any kind of communication data where the information of author, recipients, and date of delivery are readily available.

The implementation described in this section assumes that each document is initially contained in its own file and begins by iterating through the files to collect the information into two files, a central datafile and a dictionary. In the datafile each document gets a formatted row of its own. The dictionary contains a single line entry for each word recorded with the word itself and its frequency in the whole dataset. From there on SALTeD uses those files in the analysis. The process is outlined in figure 3.1. It starts by pre-clustering the documents to determine the number of clusters or topics, followed by feature selection to reduce the dimensionality of the data. Then with the information from the first two parts the data is divided into clusters that are manually examined to discover interesting topics relating to the investigation at hand. Those clusters containing interesting topics are thereafter visualized as a dynamic network, hopefully revealing some communication pattern that could aid any forensic investigation regarding the group or topic.

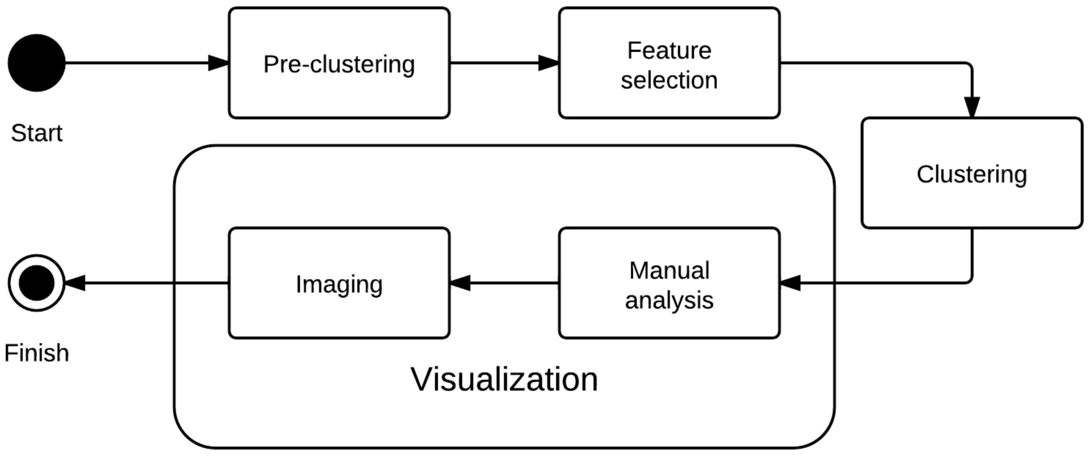


Figure 3.1: Main steps in the process from start to finish.

3.1. Pre-Clustering

The problem is to divide a set of text documents into K unknown groups and with the partitional methods K is determined by clustering iteratively with the number of chosen clusters as explained in chapter 2.1. As already mentioned that can be very time consuming for large datasets. Thus to circumvent the problem, randomly chosen subsets of the data are repeatedly clustered to approximate the number of clusters required for the whole set. This procedure is called *bootstrapping* and it limits the clustering of the whole dataset to a single pass. The whole pre-clustering process is outlined in figure 3.2. It starts by randomly selecting a subset of data points, the size of that subset is passed as an input argument where the user tries to balance between accurately representing the population and minimizing time consumption of the process. That subset is then clustered with the K-means algorithm and the optimal number of clusters determined with the approach detailed in chapter 2.1.1. If the minimum number of iterations has not been reached the pre-clustering process selects another random subset and repeats the process. Otherwise it evaluates convergence. This cycle is repeated until the maximum amount of iterations or convergence has been reached. The minimum and maximum number of iterations must be decided beforehand and should reflect the ratio of the sample size to the population, i.e. if the subset is small with respect to the population more iterations are needed. The convergence criteria is the variance in the number of clusters of at least the minimum number of iterations and at most all of them. If that variance drops below a predetermined proportion the convergence criteria is considered fulfilled and the rounded mean number of clusters is the output of the pre-clustering process.

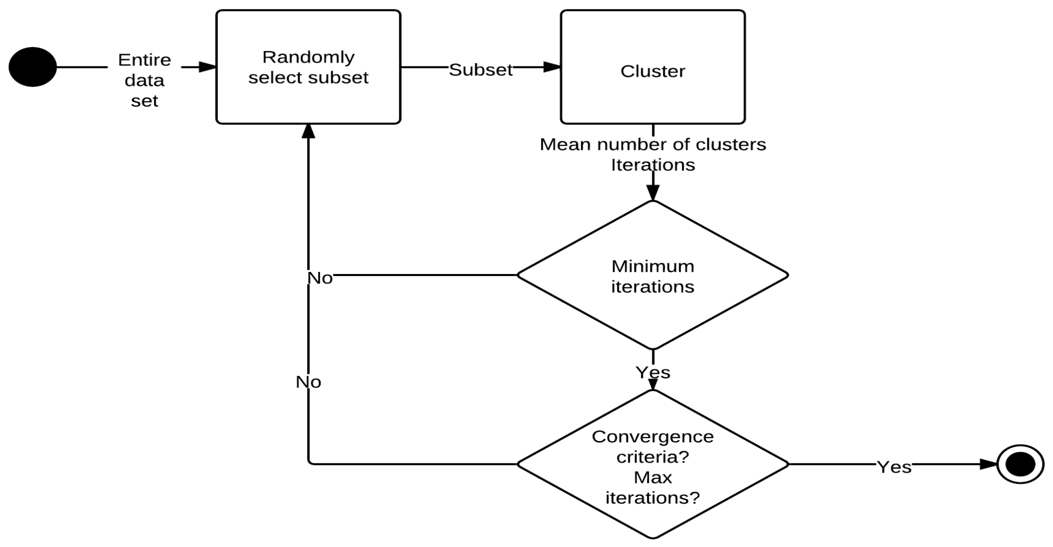


Figure 3.2: The pre-clustering process.

3.2. Feature Selection

When the objective of the clustering task is to divide text documents by topic an obvious feature is the number of occurrences of each word it contains. The first task of the feature selection process is therefore to make a dictionary, listing all strings of alphabetical characters divided by white spaces in the data, their frequency and in how many documents they appear. As mentioned before, high dimensionality makes the clustering problem very difficult and taking into account that the data is written by multiple persons on multiple occasions the dictionary, with every unique text string of a large dataset will be huge.

The inputs of the two clustering algorithms in section 3.3 are slightly different from each other and thus the feature selection process is also different. SALTeD decreases the dimensionality for the K-means algorithm as follows

- By filtering out documents that are known not to describe peer to peer communication, such as system responses, known newsletters in email collections including spam, and documents where neither the recipient nor the author is known.

- Eliminating text strings that have the same character occur more than five times in succession since that text string is very likely just gibberish or only in a single document and thus can not be used as a feature to group that document with any other. Although it is unlikely that any language has a few words that have the same character occurring five times in a row there is a possibility that any number up to that could be abbreviations, common mistypings of certain words or some slang.

- Eliminating all symbols other than alphabetical. Note that this has to be modified for each language.

- Eliminating single character text strings. In most languages this eliminates some stop words, commonly used words that do not describe any real content and all acronyms.

- A crude variation of a document threshold method (DTM) where text strings are eliminated according to their frequency in the dataset.

- The filter model depicted in figure 3.3.

Of the above steps only DTM and the filter model are applied after the data has been read into the central datafile as they work directly on the dictionary. DTM simply eliminates features that do not fall in a predefined frequency range, either they are to rare or to frequent.

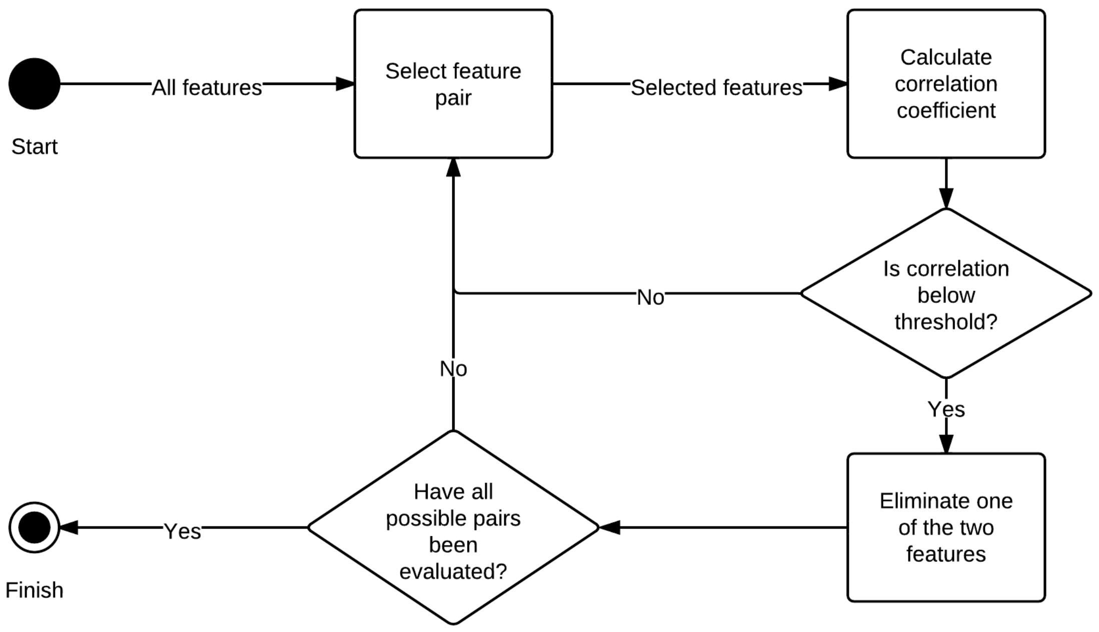

Start
All features
Select feature pair
Selected features
Calculate correlation coefficient
Is correlation below threshold?
No
Yes
Eliminate one of the two features
Have all possible pairs been evaluated?
No
Yes
Finish

Figure 3.3: The K-means feature selection process, k-select.

The remaining features are then put through the filter model in figure 3.3 that iterates through every possible pairing of features and calculates a correlation coefficient defined as the proportion of documents the pair show up in together to the total number of documents the more frequent feature appears in. This coefficient can, therefore, be maximum 1 and minimum 0 and if it drops below a given threshold one of the pair is eliminated. The current implementation of this method does, however, not iterate through every pair. Instead it iterates alphabetically pairing one feature with every subsequent feature and immediately eliminating the latter when the threshold is breached, thus avoiding pairing said feature with any more features. This method will be called *k-select* throughout the thesis since its output is specific for the K-means implementation. Three other rules could have been implemented to choose which one gets eliminated; the other type of relative position in the alphabet, i.e. first and both of relative frequency in the dataset, more or less frequent. Since this is a process of removing redundant features it should be of no consequence which rule is adopted and an arbitrary choice can be made.

The output from the K-means feature selection process is a text file where each document is represented by one row. It is similar to a sparse matrix of sorts where only the non zero elements are recorded. Every row begins with a number n counting the non-zero features of the represented document and then alternating between an index i_k of those features and c_k, their frequency in the document divided by the total frequency over all the documents. The features are sorted so that if $j < k$ then $i_j < i_k$ for each individual row. Any row in the text file will thus look something like this:

```
n i_1 c_1 i_2 c_2 ... i_n c_n
```

where n and i_k are integers and $0 < c_k < 1$. Note that the subscript k counts the non-zero elements of that particular row and i_1 in this row might not be the same as i_1 for any other row.

The input for the CURE algorithm is similar, except a full matrix is used. Every row is a document and each feature has a corresponding column. Otherwise the elements are the same as in the K-means file, document frequency divided by total frequency.

Due to this difference the input file for the CURE algorithm will in general be large. The selection process must, therefore, be altered so that the selected feature space dimensionality will not exceed a predefined size, where memory limitations typically restrain the dimensions to at most 80, with 50 being a reasonable value. Figure 3.4 shows the main steps in the selection process which for this thesis will be called *c-select*.

Initially the user chooses as many features as they see fit and this depends mostly on their knowledge of the data. The number of chosen features can be any value, the

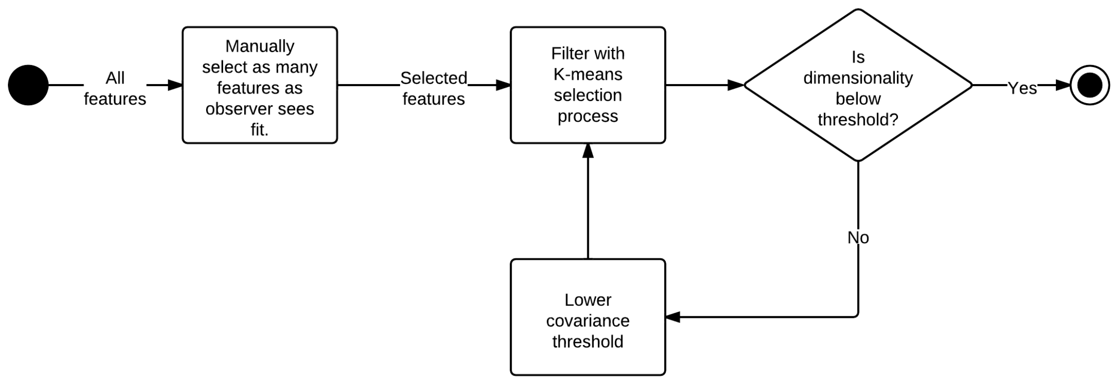

All features
Manually select as many features as observer sees fit.
Selected features
Filter with K-means selection process
Is dimensionality below threshold?
Yes
No
Lower covariance threshold

Figure 3.4: The CURE feature selection process, c-select.

filter method in the preceding part which is depicted in figure 3.3 will reduce them until the dimensionality is low enough by decreasing the threshold for the covariance coefficient in each iteration. How low is pre-specified by the user with consideration to the computational capacity of their equipment. If no a priori knowledge of the data is available the first part of selecting an arbitrary number of features could be done at random and the user only determines the starting size of the subset as well as the lower bound. This is not likely to produce good features and in this thesis it was assumed the investigators would have some a priori knowledge of the dataset.

3.3. Clustering

The framework is implemented around two different clustering methods, K-means and CURE which are both described in section 2.1. Both implementations were programed in C for reasons of efficiency.

3.3.1. K-means

The implementation of the K-means algorithm [71] closely follows algorithm 2.1, is specially designed to handle sparse data like the feature selection produces and reserves memory only for those elements that are non zero. The initial means are randomly selected from the data points. Each data point is assigned to the cluster with the nearest mean and the means are then adjusted accordingly. After the first iteration the process of assigning points to clusters is done by checking if any mean has been moved closer than the currently assigned one, if so the point in question is moved between clusters and the means are adjusted again. This iteration goes on until one or more of the following criteria is met.

- Less than predefined number of points change clusters in a single iteration.
- The residual sum of squares (RSS) changes are below a certain tolerance between iterations.
- The maximum number of iterations has been reached.

Since the datasets that are under investigation can differ widely in size, it is reasonable to define the stopping criteria by some kind of proportional limit. It is easily adjustable for any size of dataset. The bound on number of points that change emails for all tests and evaluations of the framework was arbitrarily chosen as 0.5% of the total number of data points. The tolerance for the RSS changes between iterations was set to 0.1%. The maximum number of iterations criterion is the only one that has to be bound with an absolute value instead of a fraction because considerations must be made for the equipment being used, memory and computational speed.

The output of the *K-means* code is a single column text file with as many rows as the number of documents, each line indicating which cluster the corresponding document belongs to.

3.3.2. CURE

The implementation of the CURE algorithm was done by Eui-Hong (Sam) Han [34]. It follows the algorithm described in section 2.1.2 closely.

The implementation allows the user to control several parameters. In this study only three of those were adjusted and tested, the final number of clusters, the alpha parameter and number of representing points for each cluster.

The final number of clusters is a stopping criterion since CURE is a HAC algorithm which merges clusters and will stop when that goal is reached. The alpha parameter is the tightening factor for the representing points.

3.4. Visualization

The visualization is accomplished in two steps, first by deciding what to show and secondly formating the data so that the image or video will convey as much information as possible.

3.4.1. Manual analysis

If the clustering is successful it can reveal various categories of topics in the document dataset. To identify interesting clusters the investigators must manually analyze each cluster. Depending on the investigator's a priori knowledge, the nature of the dataset and the quality of the clustering, this manual analysis can be done accurately and swiftly or be time consuming and fruitless. For this framework it is assumed that the investigators flip through each cluster, inspect a few documents for content and estimate the topic and decide if it is of special interest. It is therefore essential that they have a good idea of what to look for.

Once the clusters for further investigation have been chosen a script gathers from each document the author, recipients and the date it was published relative to the earliest dated documents. The oldest documents get the time stamp 1, documents that are a day younger get time stamp 2 and so on. This makes the default time step of the analysis a single day. All authors and recipients of the chosen cluster are registered as nodes and the documents as arcs. Two comma separated value (csv) type files are produced, one for nodes and another for arcs. An example and description of those files can be found in the appendix.

3.4.2. Imaging with Gephi

The main part of the visualization is done with Gephi, which is controlled through a graphical user interface (figure 3.5).

The network is imported by selecting *Data Laboratory*, then *Import spreadsheet* and follow on screen instructions. Nodes and arcs must be separately imported from files like the ones introduced in previous section and described in the appendix.

After importing the data, if it is dynamic, the user must select *timeline* from the *tools* menu to be able to view and manipulate the time slices. The overview tab provides various modules such as layout and ranking for making the network aesthetically pleasing and as informative as possible. At first, the positioning of the nodes is

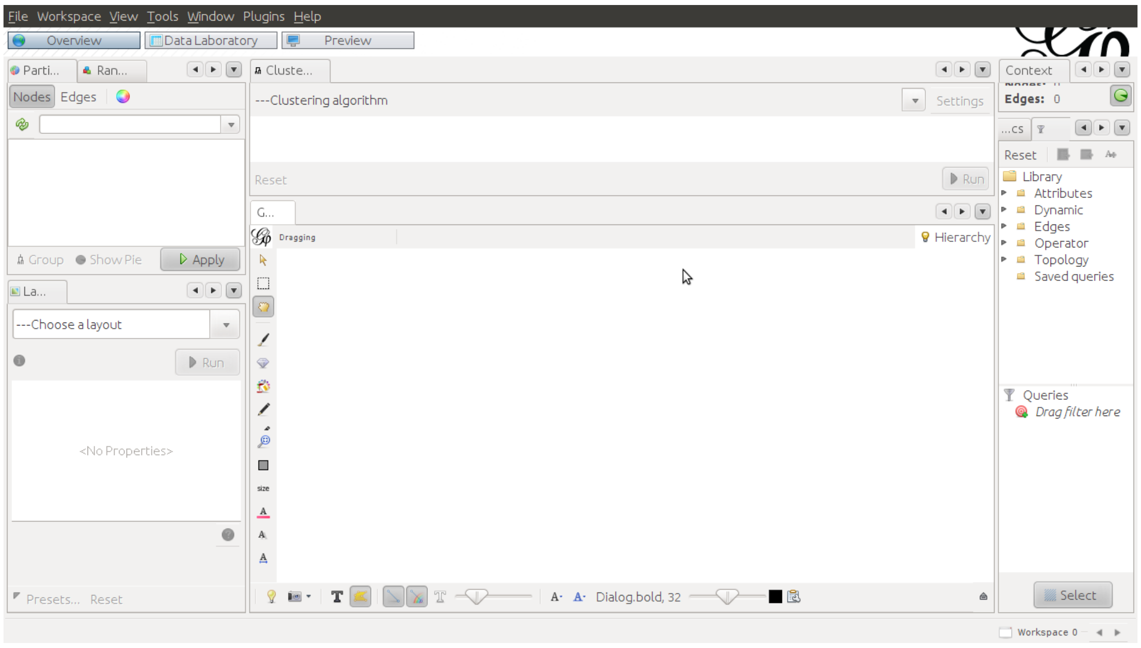


Figure 3.5: A screenshot from Gephi.

random. The layout module also provides various algorithms and means to control their properties and position. With the ranking module the color of the nodes and edges can be changed to reflect their position and connection relative to the network. The result of a random generated dynamic network and an arbitrary layout algorithm, selected from those provided by default, can be seen in figure 3.6.

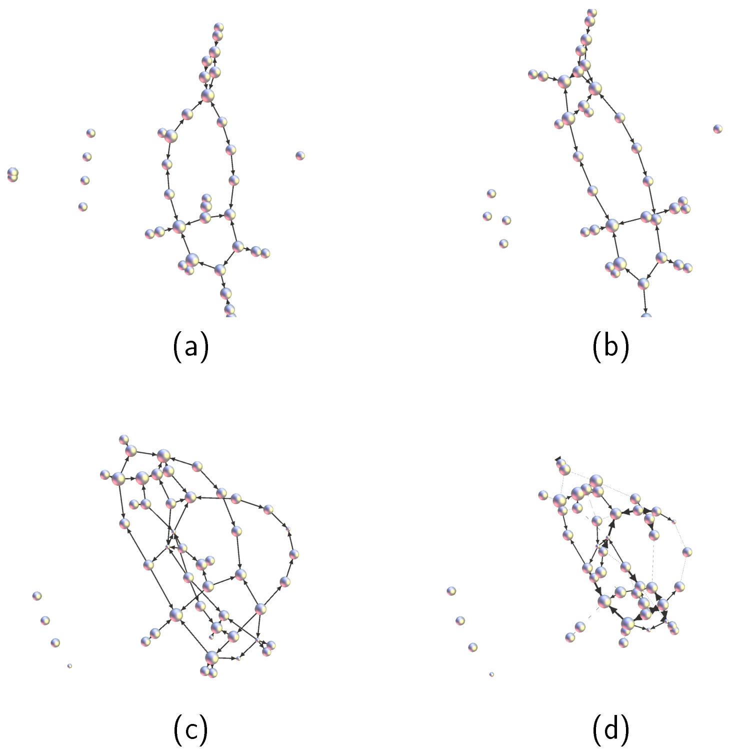


Figure 3.6: An example of output from Gephi, four time slices from the same dataset

The Gephi web site [80] provides extensive support and tutorials for the usage and development of the program.

3.5. Summary

SALTeD can aid investigators in providing forensic evidence when the task at hand is to dig through vast amounts of digitally stored text communication data. Firstly by using bootstrapping to detect the number of topics, and hence the number of clusters, from the entire feature space then reducing the dimensionality by eliminating redundant and irrelevant features. With the reduced feature space and the estimated number of clusters, the investigators can choose from the two clustering methods, *K-means* and CURE, to process the whole dataset. The results must then be explored by hand to see if any of them relate to the ongoing investigation. Those that might are, thereafter, animated and analyzed visually with a program called Gephi.

4. Experiments

The aim of the experiments conducted in this chapter is to evaluate the framework as a whole. Is it applicable and can it be useful for investigators? Although some minor details were tested the purpose of that was not to see if those sections of the framework were efficient or effective but to verify that the framework functioned as planned and possibly explore if the final outcome is sensitive to small parameter changes in individual parts of the whole process.

The tests were done on standard desktop and laptop computers running on Ubuntu 11.10 (oneiric) and Python 2.7.2.

4.1. The test data

For the evaluation of SALTeD two sets of publicly available data where used; the Enron corpus of emails which was chosen mainly for the reason that it has been used previously in a forensic investigation; and The 20 Newsgroup data because it has already been categorized into topics and has multiple authors. The categorization provides means to evaluate the accuracy of the clustering results.

The email dataset was used to test the whole framework and the newsgroup set was only used to test the parts leading up to but not including the visualization.

4.1.1. The Enron Corpus

The Enron corpus is a dataset of emails gathered from the infamous Enron corporation while under investigation following its massive bankruptcy in 2001. The version used for this thesis is publicly available on the Carnegie Mellon web site [13]. The full dataset contains more than 500.000 e-mails in 150 folders, one for every user, most of them senior managers. Each email is in its own text file, an example is given in listing 4.1.

```
Message-ID: <24839976.1075863386732.JavaMail.evans@thyme>
Date: Tue, 18 Sep 2001 12:27:43 -0700 (PDT)
From: 40enron@enron.com
Subject: PLEASE READ: ISSUES ACCESSING INTERNET AND INTRANET
Mime-Version: 1.0
Content-Type: text/plain; charset=ANSI_X3.4-1968
Content-Transfer-Encoding: 7bit
X-From: Global Infrastructure@ENRON <IMCEANOTES-Global+20Infrastructure+40
  ENRON@ENRON.com>
X-To: All Enron Worldwide@ENRON <??SAll Enron Worldwide@ENRON>
X-cc:
X-bcc:
X-Folder: \RBENSON (Non-Privileged)\Benson, Robert\Inbox
X-Origin: Benson-R
X-FileName: RBENSON (Non-Privileged).pst

Access to the Internet and Intranet is currently being affected, throughout Enron.
  Global
Infrastructure is working to alleviate the issue. There is currently no estimated
  time of
completion.

You may be effected in the following manner:
?       slow access to internet
?       no access to internet
?       slow access to intranet
?       no access to intranet

Further updates will be provided
```

Listing 4.1: An example of an e-mail from the Enron dataset.

The dates of the emails span the years 1994 to 2005 and are concentrated in the years 2000 and 2001 (figure 4.1). Due to lack of available computational capacity it was decided to only use a time period of one year and the year 2000 was arbitrarily selected. As can be seen in figure 4.1 that year alone contains nearly 200,000 emails.

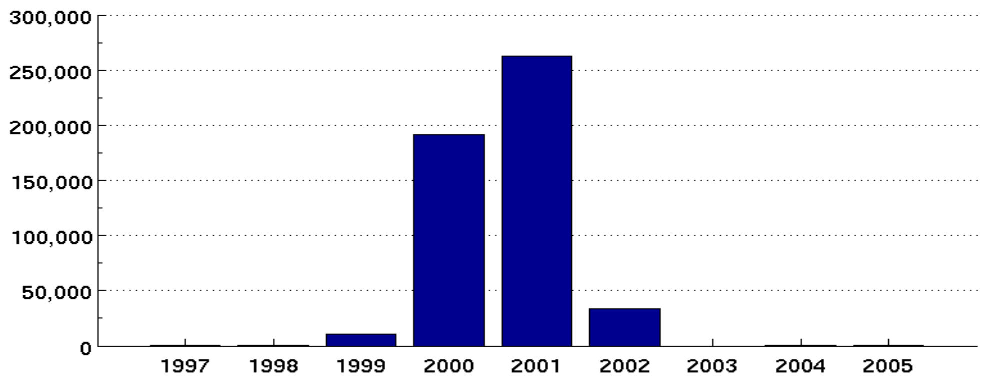


Figure 4.1: Number of the Enron emails over the years.

The version of the dataset used here does not include any attachments but a version including them all is available from E.D.R.M.s web site [79]. Some e-mails were removed after the set was originally published due to personal requests from people not directly connected to Enron.

4.1.2. The 20 Newsgroup Data

The 20 Newsgroups dataset can be found in various versions on the Internet but the original is from Tom Mitchell [60] who collected 1000 articles or discussions from 20 different newsgroups on Usenet listed in table 4.1. There is obviously a topic overlap between some groups and even to such an extent that two groups are basically the same. For an example the groups *Social.Religion.Christian*, *Talk.Religion.Miscellaneous* and *Alternative.Atheism* have closely related topics while *Computers.Windows* and *Computers.OperatingSystems.MicrosoftWindows* are likely to be very similar.

Main category	Subcategories	Group name
Alternative		Atheism
Miscellaneous		For sale
Social	Religion	Christian
Computers		Graphics Windows
	Operating systems	Microsoft Windows
	Systems	IBM pc hardware Macintosh hardware
Recreation		Autos Motorcycles
	Sport	Baseball Hockey
Science		Cryptology Electronics Space Medical
Talk	Politics	Guns Middle east Miscellaneous
	Religion	Miscellaneous

Table 4.1: The 20 Newsgroups categorized by topic

The original dataset contains 20,000 documents but the one used for this thesis was obtained from Jason Rennie [69] and contains roughly 19,000 documents after duplicates have been removed. They are sorted into folders and each document is in a separate text file with the first two lines stating from whom the article originates and the subject matter of said article as seen in listing 4.2.

```
From: keith@cco.caltech.edu (Keith Allan Schneider)
Subject: Re: >>>>>>Pompous ass

livesey@solntze.wpd.sgi.com (Jon Livesey) writes:

>>>How long does it [the motto] have to stay around before it becomes the
>>>default?  ...  Where's the cutoff point?
>>I don't know where the exact cutoff is, but it is at least after a few
>>years, and surely after 40 years.
>Why does the notion of default not take into account changes
>in population makeup?

Specifically, which changes are you talking about?  Are you arguing
that the motto is interpreted as offensive by a larger portion of the
population now than 40 years ago?

keith
```

Listing 4.2: An example of a document from the 20 Newsgroups dataset

Additionally the Newsgroups data is complemented with a vocabulary text file. It is a list of 61,188 standalone text strings in the corpus containing only alphabetical characters.

4.2. Pre-Clustering

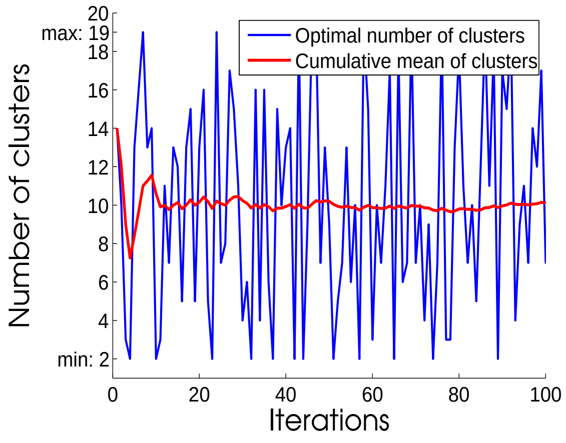
The Newsgroups data was well suited to evaluate this part of the proposed framework since the number of true clusters is approximately known. This part can most likely be generalized for larger datasets, if it can estimate correctly the number of topics in the Newsgroups which by examining table 4.1 should be in the range of 13 - 20 depending on how many similar groups will merge.

The two parameters that must be set, are the maximum number of iterations and the size of the subset. When designing the experiments, the computational capacity and purpose of the process were taken into account. The purpose is to decrease running time since estimating the number of clusters of a small sample should be faster compared to the whole dataset. The parameters were set to a limit of 300 iterations and a 1% fraction of the documents. For the Newsgroups data this accounts for slightly less than 200 documents.

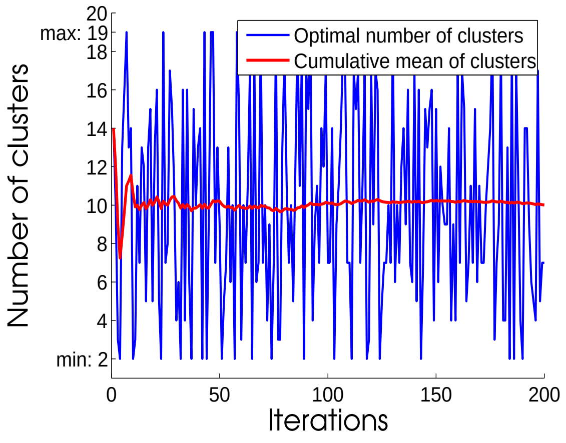
The experiment then explored how the algorithm behaves for different λ from (2.2), testing only two different values $\lambda = 1$ or 0 and not $2m$ because a preliminary inspection suggested that it would always result in two cluster optimum.

4.2.1. Results

When pre-clustering was carried out with the Newsgroups data, it was soon evident that due to computational restraints the sample portion could not exceed 1% of the 19.000 newsletters without making the process unacceptably slow. Anything less than that would not add information and since the computational speed of the program was around 30 seconds per iteration for this number of documents, the number of documents in each subset was set to 200.



(a) 100 iterations



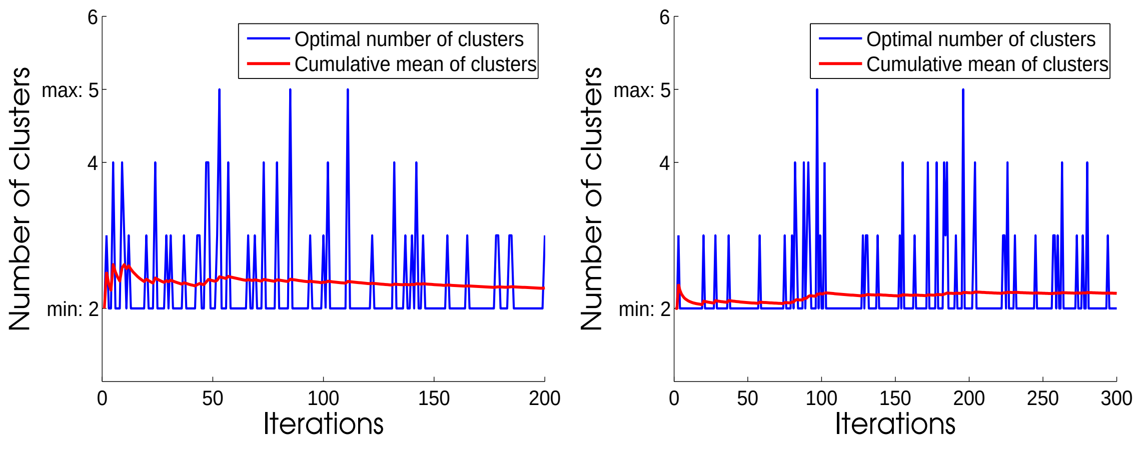
(b) 200 iterations

Figure 4.2: Results for the pre-clustering process of the Newsgroups data with $\lambda = 0$

In figures 4.2 and 4.3, results for two different values of λ are shown. Each figure shows the optimal number of clusters for each iteration, the cumulative mean of those numbers and the maximum and the minimum number of clusters recorded. It is interesting to see how fast the cumulative mean converges and the difference of the mean value and variance between λ values. A summary for the test runs is given in table 4.2

Having $\lambda = 0$ returns a mean value much closer to the original estimate (13 to 20 clusters). There are quite many iterations that reach 19 clusters but also nearly as many that find only 2. As a consequence the variance and standard error are much higher for $\lambda = 0$. It is also interesting to see that the mean seems to converge very quickly for all instances, in approximately 20 iterations, suggesting that only relatively few iterations are necessary to get a useful estimate.

After reviewing the results for the Newsgroups data, it was decided to run the Enron data through the process once with the most promising setup. Only one run was sufficient enough to get an estimate of the number of topics within the email



(a) 200 iterations with $\lambda = 1$ (b) 300 iterations with $\lambda = 1$

Figure 4.3: Results for the pre-clustering process of the Newsgroups data with $\lambda = 1$

Table 4.2: Results for the pre-clustering process.

Figure	λ	Iterations	Standard Error	Variance	Mean number of clusters	Range
Figure 4.2(a)	0	100	5.58	31.13	10.11	17
Figure 4.2(a)	0	200	5.33	28.41	10.01	17
Figure 4.3(a)	1	200	0.64	0.40	2.28	3
Figure 4.3(b)	1	300	0.54	0.29	2.21	3

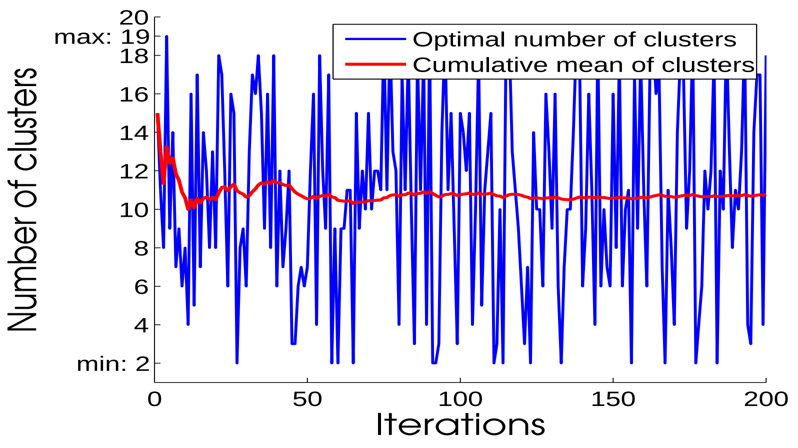


Figure 4.4: Results for the pre-clustering process with the Enron data, $\lambda = 0$, and 200 iterations

collection since the true value is hard to estimate. The setup had $\lambda = 0$ and the sample portion set to 0.1% which would make the sample size around 200 emails per iteration. The Number of iterations was 200 even though the test results from the Newsgroups indicated that only a few iterations were needed, it was decided to err on the side of caution. The results are shown in figure 4.4 and the standard error, variance, mean and range are 5.16, 26.64, 10.77 and 17 consecutively. It is apparent that no more than 50 to 60 iterations are needed to get a reasonable estimate since in iteration 50 the mean is already 10.54 which rounds up to the same as the mean for the entire 200 iterations.

4.3. Feature Selection

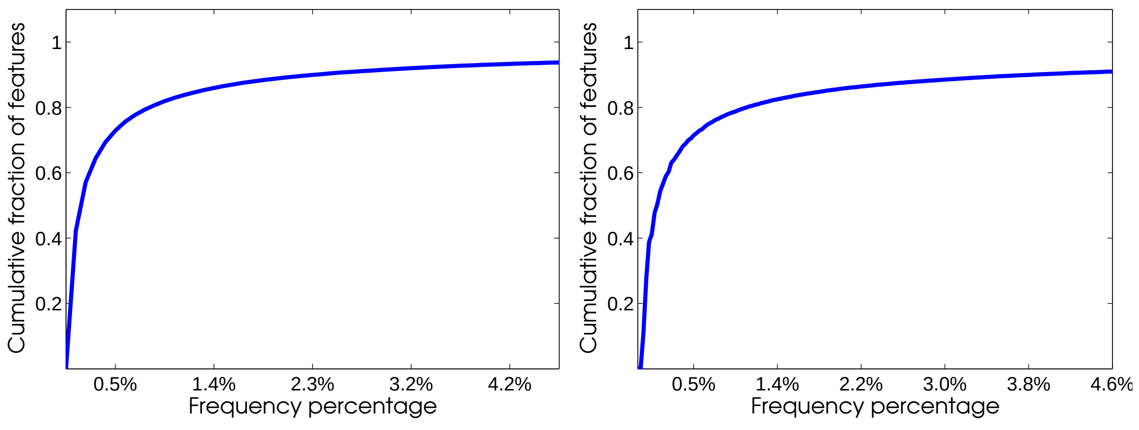
The *K-means* algorithm employs *k-select* for feature selection while CURE uses *c-select*. They are in essence the same, both start with a set of features and then reduce the set with the method shown in figure 3.3. One difference is the size of the initial feature set where *k-select* begins with all the features while *c-select* starts with a user initiated set. Another difference is the stopping criterion, *k-select* iterates once through the whole feature set while *c-select* iterates until a user defined upper bound on the dimensionality is reached.

The quality of the features will only be fully tested when the clustering is performed. It is never the less interesting to see what measures must be taken to reduce the dimensionality reliably, if it can be done at all with this simple method. Also if there are significantly many feature pairs that are mutually inclusive and therefore redundant in such large datasets.

4.3.1. Results

The maximum possible number of features is not known explicitly but a reasonable guess for large datasets is that it could correspond to the number of documents, since a single word within a document could represent a topic.

Without applying the DTM and the filter method, the number of features to select from was 95,470 for the Newsgroups data and 127,157 for Enron. Using the DTM to eliminate only the less frequent features, results in a very steep reduction as can be seen in figure 4.5 where features that have less than 1.4% total frequency make up around 80% of the total feature space. By lower cutoff value of only 2% of the document count, around 1000 plus features remain for either dataset albeit a bit more for Enron than Newsgroups as can be seen in figure 4.6.



(a) The 20 Newsgroups data (b) The Enron corpus

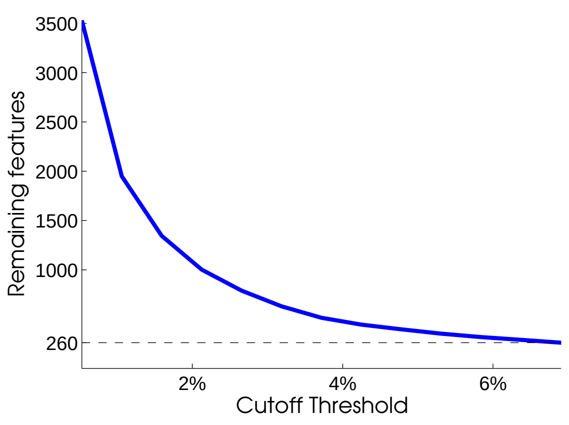
Figure 4.5: Cumulative distribution of the features based on frequency.

This one sided DTM can aid the observer in choosing a suitable interval by visually estimating a threshold value pair that would give a reasonable amount of features. Three arbitrary boundary pairs for both datasets were selected to form a small pool of feature sets. Table 4.3 lists them along with their size and antecedent boundaries. A boundary of 500 and 3000 means that only words that appear at least 500 times in the whole dataset and at most 3000 times are considered as features.

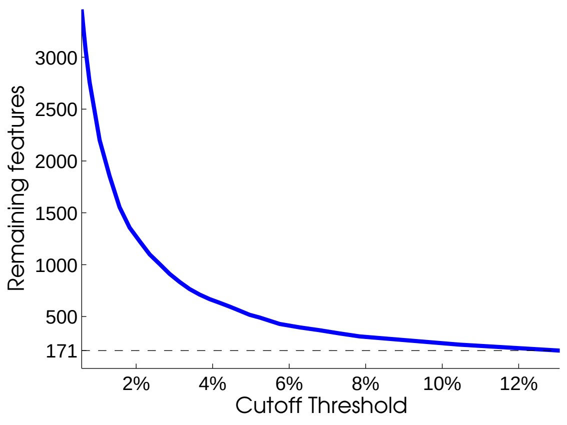
Table 4.3: Feature sets extracted for further investigation

20 Newsgroups				Enron			
nr.	Size	Bounds		nr.	Size	Bounds	
1	690	500	3000	1	867	3500	10000
2	253	1000	3000	2	998	4000	20000
3	138	1500	3500	3	1006	5000	∞

The evaluation of the filter method and subsequently *c-select* revealed that the filter model did not work well due to capacity restriction for the memory. When applied to the full Enron feature set it crashed both computers it was tested on and with the full Newsgroups set it managed to only reduce it by a handful of features for a medium correlation coefficient threshold and a long execution time. It is reasonable to assume that an even lower benchmark will probably not only remove redundant features but also quite relevant ones. The Newsgroups nr. 3 feature set from table 4.3 was the one that the method managed to reduce the most but only by 20 features with the benchmark value of 0.3 which is too low. Thus it was concluded that the ad hoc methods would have to do for now and that CURE would have to be tested on some manually chosen subsets of the features in table 4.3.



(a) The 20 Newsgroups data



(b) The Enron corpus

Figure 4.6: Feature reduction with only a lower cutoff value.

One thing that is worth mentioning is that the feature gathering, i.e. iterating through the central datafile to construct the sparse matrix text file, is computationally by far the most expensive process of the whole framework, except for the filter method.

4.4. Clustering

When testing a clustering method on large datasets it is preferable to define a good measure of quality to be able to evaluate methods and configure parameter values. In this thesis two very different clustering methods were considered, K-means and CURE. The only thing they have in common is the Euclidean distance measure. They are applied to feature spaces with a very large difference in dimensionality, for K-means the dimensions are in the region of 10^5 while CURE has an upper bound of 50 and both have various adjustable parameters.

To determine what outcome is best could be done by manually reading through the documents, classifying them and checking which clustering outcome makes the most sense. However the Newsgroups data is sorted into categories as seen in table 4.1, and it is therefore simple to compare the outcomes to that categorization and see which algorithm and configuration can match that better. To recognize the better match some values are defined. If the number of clusters is K and the number of known categories is M, some categories are likely to overlap so $M \geq K$. Therefore one easily obtained and descriptive value is the total number of clusters the categories are split over, S, i.e. the sum of the number of cluster every category is registered in. If an outcome matches the categorization perfectly then $S = 20$, i.e. the total number of categories, a greater value indicates worse match.

For uncategorized data a direct comparison of two outcomes is the simplest way of weighing the two clustering methods against each other, whether it is the same method with differently set parameters or two completely distinctive methods. For this experiment it is assumed that the pre-clustering estimates are accurate enough. It also has to be taken for granted that the features are optimally selected and therefore in an optimal world two dissimilar but equally good clustering methods should place documents in identical clusters. The dissimilarity measurement in (4.1) of the two outcomes, θ_K and θ_C, is defined as the sum of all pairwise discrepancies.

$$\Delta(\theta_K, \theta_C) = \sum_{\{(\omega_i, \omega_k) \in \Omega | i \neq k\}} \delta(\omega_i, \omega_k) \tag{4.1}$$

where

$$\delta(\omega_i, \omega_k) \begin{cases} 1 & \text{if both outcomes agree of the pair} \\ 0 & \text{otherwise} \end{cases} \tag{4.2}$$

For every conceivable pair of documents, $\{(\omega_i, \omega_k) | i \neq k\}$, in the whole dataset, Ω, a discrepancy, δ, equals 1 if the documents share a cluster in one outcome but not the other and 0 otherwise. The best case scenario would be $\Delta_B = 0$ and worst case scenario Δ_w for n documents would be:

$$\Delta_w = \frac{n^2 - n}{2} \tag{4.3}$$

For two outcomes, K and C, the relative dissimilarity $\Delta_{K,C}$ on dataset Ω is defined as the proportion of total discrepancy with respect to the worst case scenario

$$\Delta_{K,C} = \frac{\Delta(\theta_K, \theta_C)}{\Delta_w} \tag{4.4}$$

4.4.1. Results

Similar to the feature selection experiments the clustering was only moderately successful. CURE failed completely on the full datasets, it managed to complete one execution on the Newsgroups data with a feature set of 19 manually selected words out of 95,470, with the final number of clusters set to 10, $\alpha = 0.1$ and 10 representative points for each cluster. The outcome resulted in $S = 184$ which makes the average topic being split over about 9 clusters. No other performance data of CURE could be retrieved since all other executions of it failed due to full memory.

K-means did a better job and while some results are questionable it never failed to terminate. Table 4.4 summarizes the average performance of 20 executions of the K-means for the feature sets from table 4.3. The first column indicates what feature set was being used, *20N* for the Newsgroups data and *En* for Enron, the second column how many clusters were input to the algorithm, the third the number of iterations it took to reach convergence, next what the final residual sum of squares was and finally the S value where applicable.

Table 4.4: K-means performance data.

Feature set	Clusters	Iterations	RSS	S
20N 1	10	20	2.43	81.4
20N 2	10	18	0.49	88.3
20N 3	10	19	0.17	110
En 1	11	12	0.52	n/a
En 2	11	10	0.46	n/a
En 3	11	12	0.33	n/a

K-means did not require many iterations to reach a small residual sum of squares. Around twenty iterations for Newsgroups and twelve for Enron. The lowest final RSS for the Newsgroups results gives the highest S value. That could indicate that either of those measurements are not suitable for this kind of clustering problems. There is no S value for the Enron data since it is not previously categorized.

The S values of 81.4 and 88.3 are reasonable for the first two sets and suggest that the average topic is being divided among about 4 clusters. The last value of 110 is a bit high where the average topic is put in nearly 6 different clusters. A manual inspection of the results also indicates that the two first outcomes are partly reasonable. There is one huge cluster containing 70-90% of all documents but the rest of the clusters seem fairly good, albeit not optimal. Although the Newsgroups data is known to have equally many documents in every topic, the clustering does not reflect that, indicating an unsuccessful clustering. The initial centers are probably to blame since they are random and the end result is highly dependent on them. Also the number of centers are probably too few as the number of topics are likely to be more than 10 and closer to 15.

Manual inspection of the Enron results reveal that, similar to the Newsgroups data, there is always this one very large cluster but additionally, an intermediate size cluster which also overshadows the smaller ones. Some of the smaller clusters though are very promising, containing only a handful of emails but they are all of the exact same type.

When comparing the performance of K-means on different feature sets, the $\Delta_{K,C}$ measure (4.4) was calculated three times between every outcome for the Enron dataset. The results are listed in table 4.5. The values in that table suggest that it does not seem to matter which feature set is used, K-means delivers very similar results each time.

Table 4.5: Δ comparison of k-means with different feature sets on Enron

Feature sets	$\Delta_{K,C}$
En 1 vs. En 2	0.0696
En 1 vs. En 3	0.1012
En 2 vs. En 3	0.0957

4.5. Visualization

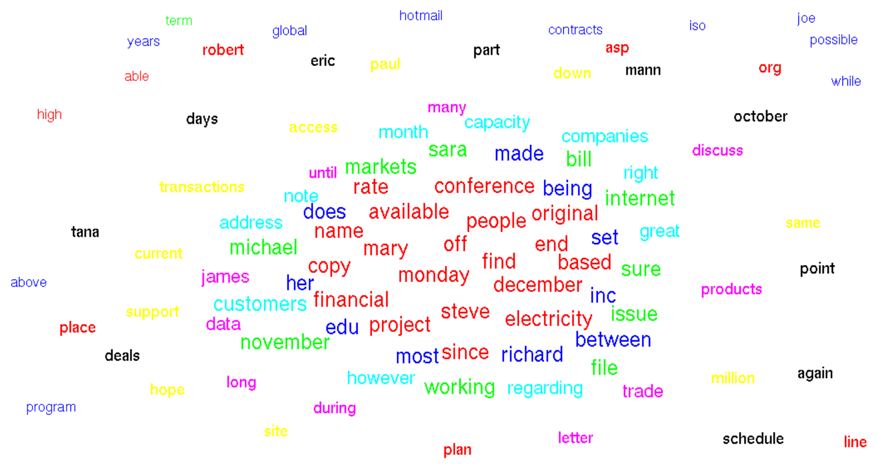


Figure 4.7: A word cloud to better envisage the topic of a particular cluster.

The evaluation of the visual processing was as much a test of the researcher's insight of the data as it was a test of the visualization program. This is a test of the aesthetic

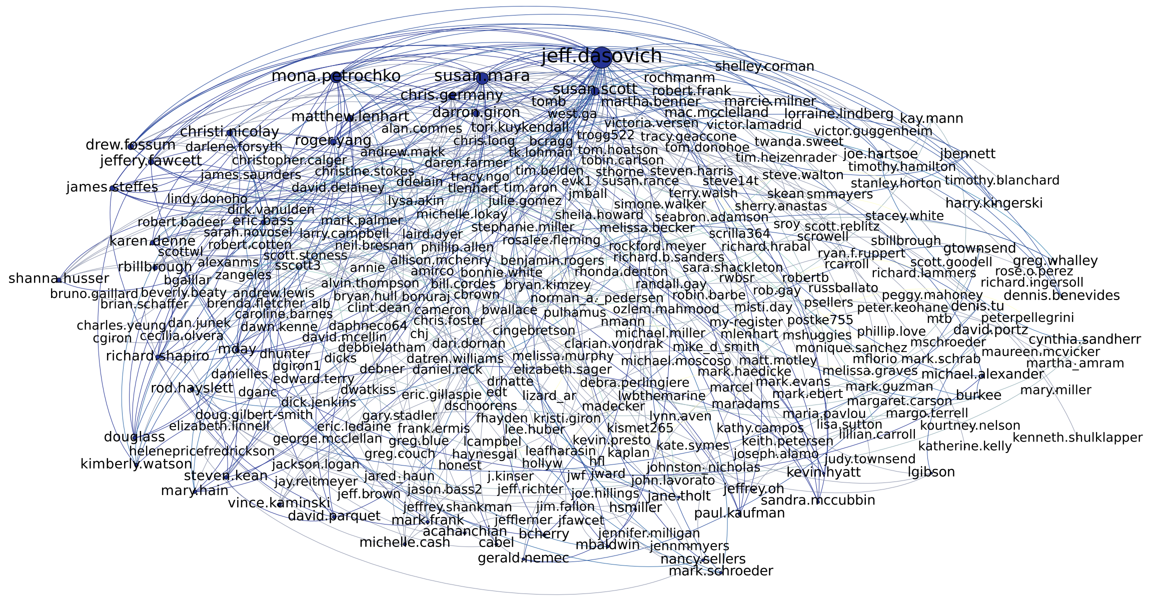
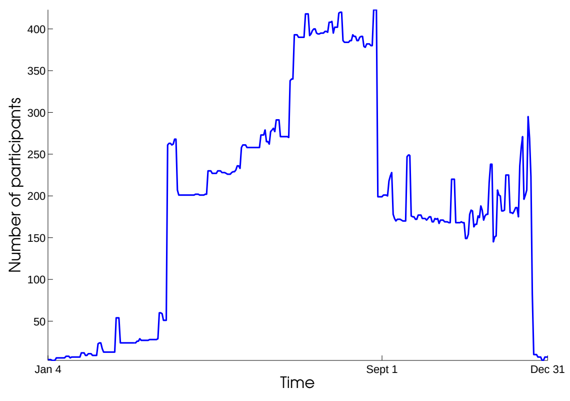


Figure 4.8: The full network involving the topic from figure 4.7

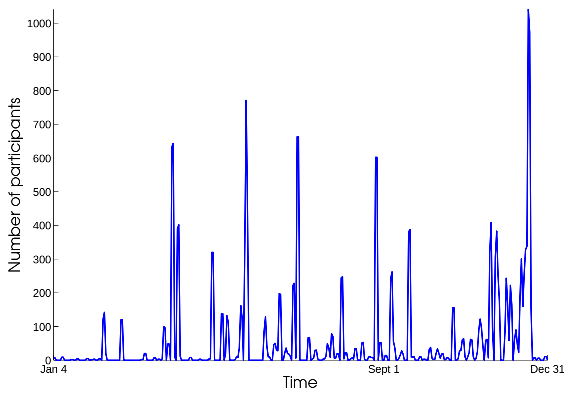
nature or in other words, an endeavor to make a good looking picture from the provided data which in this case is the Enron corpus and the clustering information from the previous section. First it is necessary to inspect the contents of the formerly mentioned clusters and take into account the frequency of newsletters, number of participants versus number of mails and other details which can be quickly estimated by skimming over the groups. It is also a good idea to make a word cloud from the clusters and to see if that gives a better idea of what they contain. Figure 4.7 shows one such cloud for a cluster from the results of clustering with the Enron feature set number 2 which has been chosen to demonstrate the visualization process. From this particular cloud it could be inferred that the topic is related to some kind of projects with words like project, financial, off, bill, companies etc. A manual inspection of the content reveals that it is a topic mainly involving oil and gas pricing regarding diverse projects by Enron and competitive firms. Visualizing the network connected with that topic, might reveal hidden influences or power struggles.

Before importing the network into Gephi, decisions must be made regarding the time intervals when the participants and emails are visible as nodes and edges, to best represent the dynamic in the network. For this experiment nodes will appear simultaneously as the first edge they are connected with and be visible for one week after its last connection if the communications occur with intervals shorter than a week. In any interval between communications that is longer than a week the nodes disappear. Edges will be visible only for the day they occur.

The network is assigned a random layout when it is first imported and does not make much sense. After applying layout algorithms implemented in the program and manipulating the size and color of the nodes the network will look like what can be seen in figure 4.8. The size of a node is proportional to the number of edges connected to it. As can be seen, Jeff Dasovich is a dominant person in the topic, right above Mona Petrochko and Susan Mara. Dasovich is a government relations executive within Enron [68], Petrochko is a trader [15] and Mara is Enron's California director of Regulatory Affairs [66]. Mara was one of the executives informed about Enron's traders stealing from California and on wiretaps from September 2000 she is heard talking about it [1]. By inspecting node and edge frequency with respect to time in figure 4.9, the time around September that year might be of interest for investigators when there is a one edge spike of many and the number of nodes drops significantly.



(a) Nodes in the year 2000

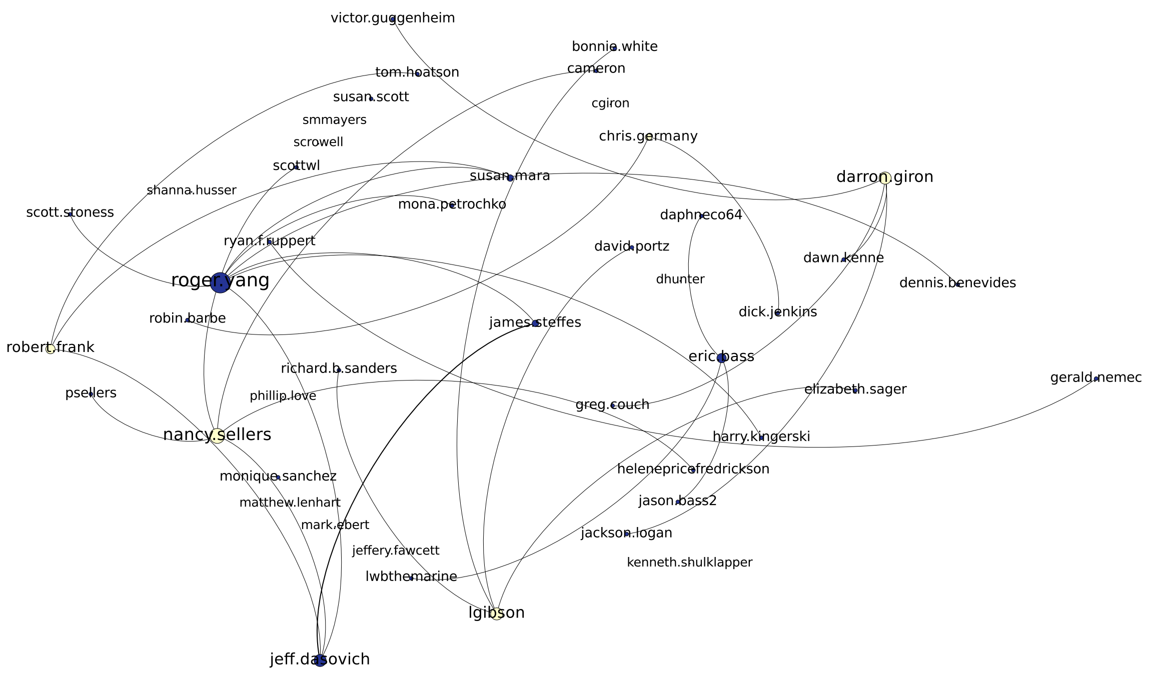


(b) Edges in the year 2000

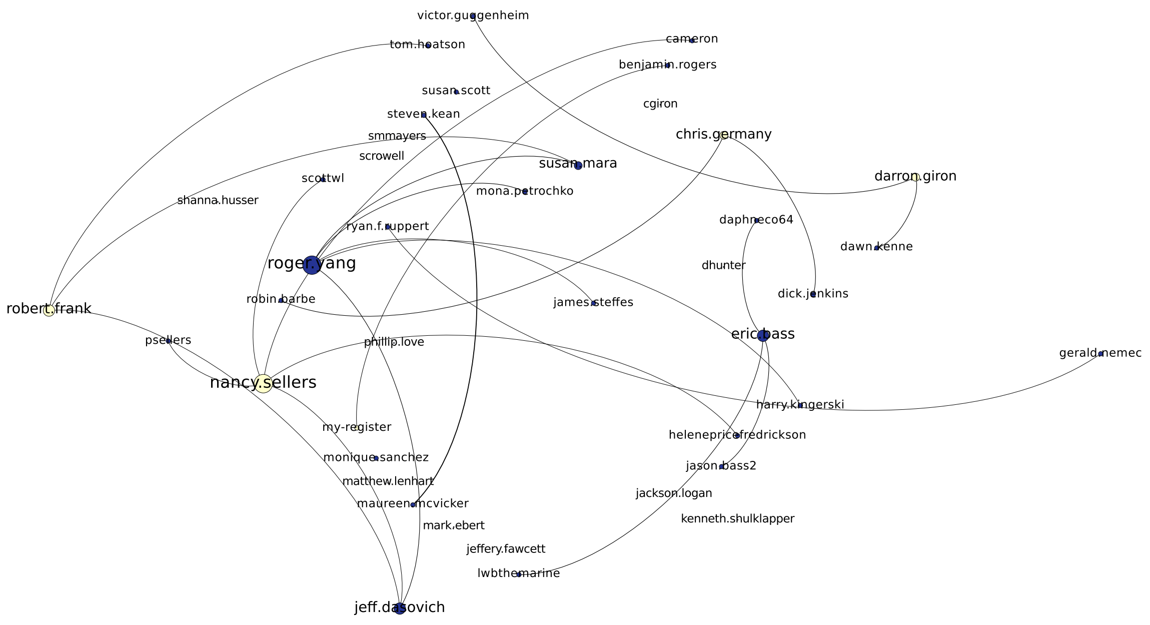
Figure 4.9: Timelines for the frequency for nodes and edges of the Network

By subsequently playing out the dynamic network for this specific topic shows how Mara is not involved in it until around that same time and stays constantly involved throughout the remainder of that year. Even though Mara appears as one of the dominating persons in figure 4.8 for the whole year, she does not seem to contribute that much in each time slice as seen in figures 4.10 to 4.11 where a fraction of the network from late September to early October is depicted in four pictures. What contributes to her status as one of the top contributers over the whole year, is that from the moment she appears she is evenly connected until December 31st. Mara has been confirmed as a person that knew what was going on and her involvement and status as the one of three to four most influential persons of this topic does not come as a surprise. The figures suggest that Petrochko's and Dasovich's might have had some knowledge about Enron's illegal manipulation of the electrical power market. The time of Mara's first involvement in this topic, in September 2000 in the wake of the drop in the number of participants as seen in figure 4.9(a) in addition to the information from the wiretaps could indicate the timeline and the level of her involvement in the fraud.

As has been suggested, such pictures and corresponding videos in addition to other data and reasonable assumptions could benefit investigators in inferring otherwise unattainable information about the structure under investigation, in this case the Enron employees. More information might be readily available through these images but deeper knowledge about relevant laws, the investigation and parties involved is needed to infer more.

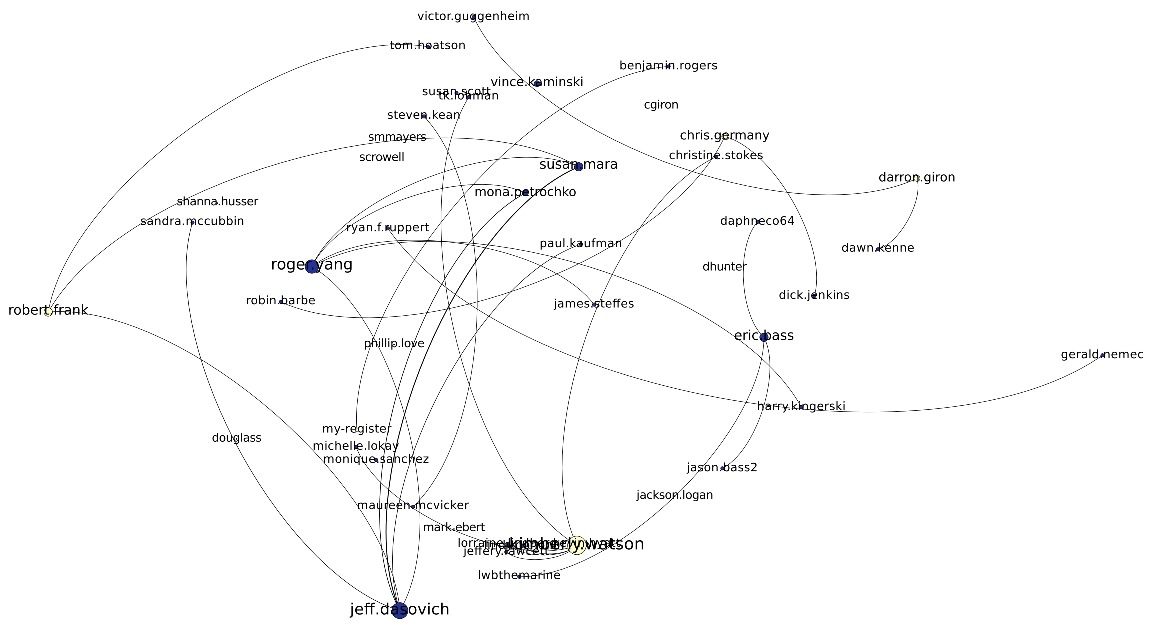


(a) Mid September

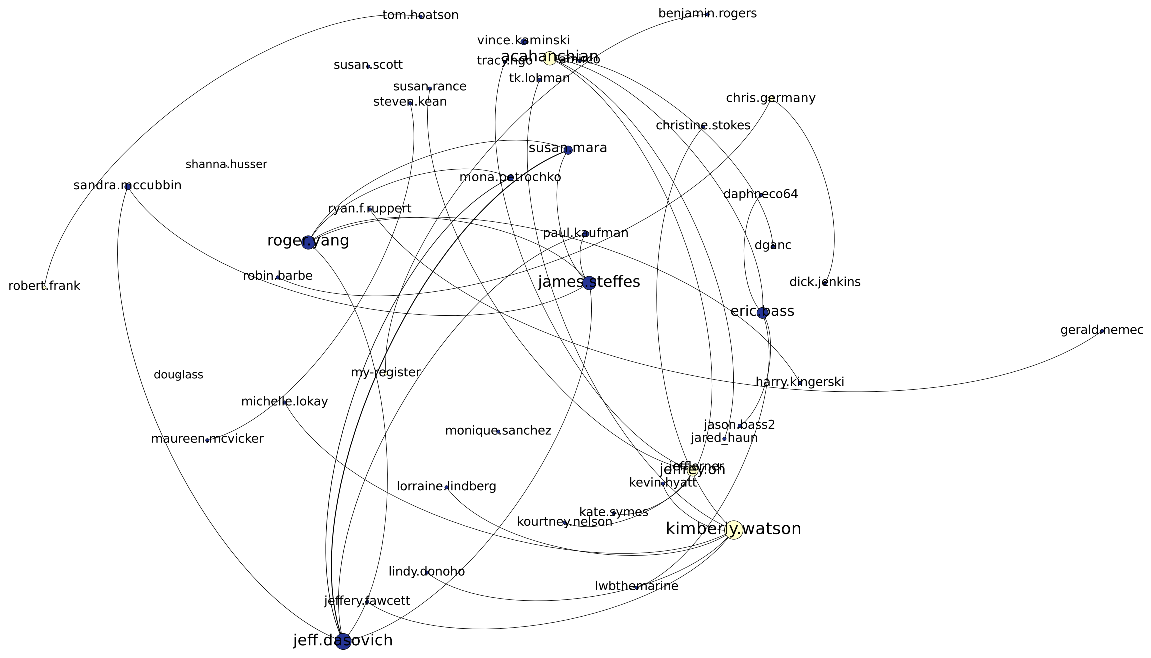


(b) Late September

Figure 4.10: Two, 10 day long time slices of a dynamic network drawn from a topic within Enron in the year 2000.



(a) Early October



(b) Mid October

Figure 4.11: Two, 10 day long time slices of a dynamic network drawn from a topic within Enron in the year 2000.

4.6. Summary

Both successes and failures were observed while evaluating the implementation. Having two different sets of data to experiment with was very useful. The categorized and orderly 20 Newsgroups dataset, was valuable in testing the clustering methods. Without it the cluster quality could have been harder to judge. The unsorted Enron corpus provided peer to peer communication data for dynamic social network analysis.

The first success of the evaluations was the pre-clustering process. It was fast and able to estimate quite closely the number of topics even though the mean value was slightly off. One may suggest when it comes to clustering the whole set to try also to partition it according to the maximum value of obtained clusters in the pre-clustering.

The feature selection experiments were partly successful. The intuitive methods of elimination worked well by reducing the feature space by nearly 98% while the new filter method did very poorly and can not be recommended for further use in its present form.

The clustering was, similar to the feature selection experiments, only partly successful. K-means executed swiftly and without a hitch, albeit the results were not par excellence they can serve their purpose, which is to aid forensic investigators. The CURE algorithm only terminated successfully once and with doubtful results but that can be attributed to computational limitations and lack of better features.

Dynamic social network analysis was successfully used by a non-expert to produce aesthetically and informative images, revealing some suggestive, officially unknown information about executive employees of Enron. Therefore this should be a very powerful tool in the hands of an expert.

While developing the implementations and subsequent tests, large amount of data was produced, scripts, programs, figures, data files and results. All this as well as a detailed step by step user guide are available on the accompanying DVD disc.

5. Discussion

SALTeD is a framework for aiding forensic detectives in their work, so even though the evaluation revolves around testing individual parts of the process it must be looked at from a broad perspective. Some individual parts did not live up to expectations but they can be replaced and/or enhanced. Overall the framework works and has the potential to work very well. The methods that were implemented and tested are only a small fraction of available methods from information retrieval, machine learning, statistics, computer sciences, mathematics, etc. The performance of the SALTeD framework can be enhanced by using more problem specific knowledge about the communication at hand. For example, they may be language specific, prior knowledge about the subject and so on. This framework is hopefully, a first step in the right direction of making use of digital forensic investigation in the future.

5.1. Pre-clustering

The pre-clustering method, with bootstrapping worked well, it managed to estimate the number of clusters quite nicely but not accurately. The mean converged within 50 iterations but to a number of clusters below what was expected. Looking at figures 4.2, 4.3 and 4.4 it is possible that with a full feature set and only a handful of data points the method is overfitting.

If the only problem is to find how many clusters any dataset contains this kind of pre-clustering would work quite well but one should also consider the maximum number of clusters it finds.

5.2. Feature selection

This part of the framework was tested only on a simple implementation but it is a vital part of the whole process. The outcome of all that follows relies on what features are chosen. By choosing bad features the best pattern finders will struggle or fail to find anything that resembles a cluster. So it is of the outmost importance to chose them carefully. Choosing all available features would at best skew the outcome and more likely ruin it completely.

The two methods that were tested, and hoped to complement each other in reducing the dimensionality, were a simple document threshold method and a simple correlation based filter method. Surprisingly the former worked well by eliminating a large part of the feature space while the latter was unsuccessful. The filter method was very time consuming since its function is to iterate over every possible pair of features and calculate their correlation coefficient. If the feature space is $\mathbf{R}^n$ the computations will be $\frac{n^2-n}{2}$ calculations of a correlation coefficient for the worst case scenario. Given that n =95,470 for the Newsgroups and 127,157 for Enron, the calculations could and did take a very long time. It is worth further inspection why this simple filter method fails for the Newsgroups data, where n is so much greater than the number of emails. By reasonable assumptions there should be significantly many pairs of features, that appear often enough in the same documents, to make one of the pair redundant. This might suggest that the implementation is in some way faulty.

While the document threshold method eliminates successfully many features it is highly likely that it could have missed popular slang, abbreviations, and/or very common mistypings. Therefore it should be used with caution.

5.3. Clustering

The partitioning of documents into topics is a central part of the framework. It enables the investigator to exclude irrelevant documents and examine the general conversation within an organization or a group. For this thesis two clustering algorithms were tested, the K-means algorithm and CURE.

The K-means algorithm was partly successful and was able to divide the Newsgroups data into reasonable clusters, although not perfectly. The very large clusters it produced were likely caused by the hyperspherically shaped clusters that limit the usefulness of the method and the number of clusters was less than the number of true topics. Each smaller cluster contained a specific topic and by excluding the larger clusters the outcome could be considered successful. Apart from being able to match the Newsgroups data reasonably as seen by the S values in table 4.4, the K-means algorithm was also fast and terminated correctly every time. Another interesting result for the K-means algorithm was that when comparing different feature sets for the Enron dataset, it did not matter much what set was used, discrepancies were within 10% of what they possibly could be.

Unlike the K-means algorithm, CURE was not successful. For very small fraction of the datasets and two to four final clusters it worked well, albeit not perfectly. When tested on the full datasets it did not work, took days to run and crashed both test computers by filling their memory before terminating. It is strange that CURE did not work since it was originally developed to cluster large datasets. A reasonable explanation for this behavior is that the program that implements the algorithm takes an input of a full matrix instead of a sparse one and that limits the potential feature space to only a small number of dimensions. Although too many features can cause some complications, choosing too few features for clustering topics that might span several other topics and sub-topics is also considered a bad practice. Since the feature reduction methods were not effective in choosing 15-30 features out of many thousands it had to be done manually. Never the less, this is an implementation for forensic investigation so the person choosing the features manually must have extensive knowledge of the data and especially the language used in it to be able to identify dis-ambiguous words.

5.4. Visualization

Visualizing individual clusters in a network is a simple routine which can, however, be made complicated. In the hands of those that can master it and with the right tools the visualization process can become a powerful weapon in any prosecutors arsenal. Choosing the cluster to draw is a simple method of look and see, maybe using word clouds, or something similar, if the clusters are big and drawing them all is not favorable. Timelines of the frequency of participants and connections for each topic could also help in the choice of topics if the time of the offense is known prior to the network analysis. Drops or tops of the frequency that coincide in time with the alleged offense could indicate a topic of interest to the investigators.

The imaging itself demands delicate dexterity in the hands of a professional. To work properly it requires patience and insight. The choice of attributes such as visibility, ranking and position of nodes and edges in the network is rather problem specific. What clearly reveals hidden and novel information for one case might not work at all for another problem. The time of visibility is related to how long one connection of a node has influence in the network, is it a day, a week or weeks? That choice must be made with good understanding of the dynamic communication within the organization under investigation. Node ranking demands the same knowledge since there are multiple ranking parameters for the time being displayed to choose from, such as:

- Degree, the number of total connections each node has.

- Indegree, the number of ingoing connections for each node.
- Outdegree, the number of outgoing connections for each node.

Is a person considered more influential if she is informed of everything that happens and does not contribute to the conversation as much or is the person doing the informing the more influential? In section 4.5 the ranking was done by degree for each timeslice in figures 4.10, 4.11 and the whole network in figure 4.8. By that ranking Jeff Dasovich seemed the most influential in a topic closely related to Enron's illegal electrical market manipulation even though he was not directly connected to that by prosecutors during the trials.

With deeper knowledge and insight to the investigation these information might prove valuable, either for the prosecution or the defendants. Maybe not as evidence but at least as a means to drive a point home to the jury or the judge. With future research and the ongoing development of the open source programs, similar to Gephi, dynamic social network analysis will play an increasing role in criminal forensic investigations.

6. Conclusions and future research

As stated before SALTeD is only a framework providing a general way of aiding the investigation of digital forensic evidence. This framework can and should be developed further as a fully functional program. It has potential, if correctly used and if improved methods will be implemented in each step. Even in its infantile state it was successfully used to infer interesting information from the Enron email corpus about some executive employees. Even if the information is not incriminating, it might point the investigation in the right direction. Clustering with the simplest of algorithms, K-means, and with respect to automatically chosen words from the email body successfully revealed a topic that was used to infer these information. Of course the framework had some shortcomings, notably:

- The computational complexity, making it inefficient on personal computers (pc).
- The results from section 4.4.1 suggest that, most likely, some documents that belong to a specific topic have been overlooked since they were clustered with one of the larger clusters.
- One of the two suggested automatic feature selection methods is rough and messy, and the other is time consuming and ineffective.

By improving the steps of the framework with further research and development, these kind of problems could be overcome. Future research should at least include the following questions and projects:

- With better access to supercomputers and availability of computer clouds, is it necessary to continue developing the framework with the pc in mind?
- Since investigators are usually looking for a problem specific information from a network is an efficient automatic feature selection feasible?
- What happens if the feature selection is moved to the front of the process so that the pre-clustering would have fully reduced features to work with? Would information be lost so that the pre-clustering can not estimate the right amount or close enough to the right amount of clusters or would it stop

potential over-fitting?

- Enhancing K-means to a divisive hierarchical method by using it with only 2 clusters recursively until the clusters are as many as the documents or a stopping criterion has been reached.

- Improve and shape the implementation of CURE for large sparse data.

- Does the field of forensic investigation have any special needs regarding visual analysis?

References

[1] Rafael Azul. Enron tapes expose blatant criminality of corporate America. In *World Socialist Web Site,* 2004. The International Committee of the Fourth International. June 14. 2012, `http://www.wsws.org/articles/2004/jun2004/enro-j15.shtml`

[2] G.H. Ball and D.I. Hall. Some Fundamental Concepts and Synthesis Procedures for Pattern Recognition Preprocessors. In *Proc. Int'l Conf. Microwaves, Circuit Theory, and Information Theory*, pages 281–297, 1964.

[3] Mathieu Bastian, Sebastien Heymann, and Mathieu Jacomy. Gephi: An Open Source Software for Exploring and Manipulating Networks. In *International AAAI Conference on Weblogs and Social Media*, 2009.

[4] Vladimir Batagelj and Andrej Mrvar. Pajek-program for large network analysis. *Connections*, 21(2):47–57, 1998.

[5] Richard E. (Rand Corporation) Bellman. *Dynamic Programming.* Princeton University Press, 1957.

[6] A. Ben-Dor, R. Shamir, and Z. Yakhini. Clustering gene expression patterns. *Journal of computational biology : a journal of computational molecular cell biology*, 6(3-4):281–97, 1999.

[7] S Bender-deMoll. The art and science of dynamic network visualization. *Journal of Social Structure*, 7(2), 2006.

[8] J.L. Bentley and J.H. Friedman. Data structures for range searching. *ACM Computing Surveys (CSUR)*, 11(4):397–409, December 1979.

[9] Manasi Bhattacharyya, S Hershkop, and E. Eskin. Met: An experimental system for malicious email tracking. In *Proceedings of the 2002 workshop on New security paradigms*, pages 3–10. ACM, 2002.

[10] A.L. Blum and Pat Langley. Selection of relevant features and examples in machine learning. *Artificial intelligence*, 97(1-2):245–271, 1997.

[11] Rich Caruana and Dayne Freitag. Greedy attribute selection. In *11th international conference on machine learning*, pages 28–36. Citeseer, 1994.

[12] KG Coffman and A.M. Odlyzko. The size and growth rate of the Internet. *First Monday*, 3(10):l–25, 1998.

[13] Enron Email Dataset William W. Cohen. 2009. Carnegie Mellon, School of Computer Science, March 23. 2012, `http://www.cs.cmu.edu/~enron/`

[14] James S. Coleman. *The adolescent society.* New York: Free Press, New York, NY., 1961.

[15] The state of California, California Public Utilities Commission. Proceeding A9702022, 1997.

[16] Sanmay Das. Filters, wrappers and a boosting-based hybrid for feature selection. In *18th International Conference on Machine Learning*, pages 74–81. Morgan Kaufmann, 2001.

[17] M. Dash and H. Liu. Feature selection for classification. *Intelligent data analysis*, 1(3):131–156, 1997.

[18] Manoranjan Dash, K. Choi, P. Scheuermann, and H. Liu. Feature selection for clustering-a filter solution. In *Proceedings of 2002 IEEE International Conference on Data Mining*, pages 115–122. IEEE, 2002.

[19] R. Davidson and D. Harel. Drawing graphs nicely using simulated annealing. *ACM Transactions on Graphics (TOG)*, 15(4):301–331, 1996.

[20] O. de Vel, A Anderson, M Corney, and G Mohay. Mining e-mail content for author identification forensics. *ACM SIGMOD Record*, 30(4):55, December 2001.

[21] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–38, 1977.

[22] R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification.* John Wiley & Sons, Inc., New York, NY., 2 revised edition, 2004.

[23] Jennifer G Dy. Unsupervised Feature Selection. In Huan Liu and Hiroshi Motoda, editors, *Computational Methods of Feature Selection*, Data Mining and Knowledge Discovery Series, chapter 2, pages 19–39. Chapman & Hall, 2008.

[24] Jennifer G. Dy and Carla E. Brodley. Feature subset selection and order identification for unsupervised learning. In *17th international conference on machine learning*, pages 247–254. Morgan Kaufmann, 2000.

[25] S.T. Eick. Aspects of network visualization. *Computer Graphics and Applications, IEEE*, 16(2), 1996.

[26] John Ellson, Emden Gansner, Lefteris Koutsofios, Stephen North, and Gordon Woodhull. Graphviz - Open Source Graph Drawing Tools. In *Graph Drawing*, pages 594–597. Springer, 2002.

[27] M.L. Fredman, R. Sedgewick, D.D. Sleator, and R.E. Tarjan. The pairing heap: A new form of self-adjusting heap. *Algorithmica*, 1(1):111–129, 1986.

[28] L.C. Freeman. Visualizing social networks. *Journal of social structure*, 1(1):4, 2000.

[29] Linton C Freeman, Cynthia M Webster, and Deirdre M Kirke. Exploring social structure using dynamic three-dimensional color images. *Social Networks*, 20(2):109–118, April 1998.

[30] Thomas M J. Fruchterman and Edward M. Reingold, Graph drawing by force `[U+2010]`directed placement. *Software: Practice and Experience*, 21(NOVEMBER):1129–1164, 1991.

[31] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. CURE: an efficient clustering algorithm for large databases. *SIGMOD Record*, 27(2):73–84, 1998.

[32] Rachid Hadjidj, Mourad Debbabi, Hakim Lounis, Farkhund Iqbal, Adam Szporer, and Djamel Benredjem. Towards an integrated e-mail forensic analysis framework. *Digital Investigation*, 5(3-4):124–137, March 2009.

[33] Mark A. Hall. Correlation-based feature selection for discrete and numeric class machine learning. In *17th International Conference on Machine Learning*, pages 359–366. Morgan Kaufmann, 2000.

[34] Eui-Hong Han, 2005. CURE *Computer program*, February 6. 2012, `http://www.msdnproject.com/code/97954.htm`

[35] David Harel and Yehuda Koren. Graph Drawing by High-Dimensional Embedding Drawing Graphs in High Dimension. *Journal of Graph Algorithms and Applications*, 8(2):195–214, 2004.

[36] J.A. Hartigan. *Clustering algorithms.* John Wiley & Sons, Inc., New York, NY., 1975.

[37] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer Series in Statistics. Springer New York, New York, NY., 2nd edition, 2009.

[38] M. Himsolt. Comparing and evaluating layout algorithms within GraphEd. *Journal of Visual Languages and Computing*, 6(3):255–273, 1995.

[39] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, September 1999.

[40] A.K. Jain and R.C. Dubes. *Algorithms for clustering data.* Prentice-Hall, Inc., Engelwood Cliffs, NJ., February 1988.

[41] A.K. Jain and P.W. Duin. Statistical pattern recognition: a review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):4–37, 2000.

[42] Anil Jain and Douglas Zongker. Feature Selection: Evaluation, Application, and Small Sample Performance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(2):153–158, 1997.

[43] George H. John, Ron Kohavi, and Karl Pfleger. Irrelevant features and the subset selection problem. In *Proceedings of the 11th Int'l Conf. Machine Learning*, pages 121–129. Morgan Kaufmann, 1994.

[44] Tomihisa Kamada and Satoru Kawai. An algorithm for drawing general undirected graphs. *Information processing letters*, 31(1):7–15, 1989.

[45] T. Kanungo, D.M. Mount, N.S. Netanyahu, C.D. Piatko, R. Silverman, and a.Y. Wu. An efficient k-means clustering algorithm: analysis and implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):881–892, July 2002.

[46] Y.S. Kim, W.N. Street, and Filippo Menczer. Feature selection in unsupervised learning via evolutionary search. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 365–369. ACM, 2000.

[47] Bryan Klimt and Yiming Yang. Introducing the Enron corpus. In *First conference on email and anti-spam (CEAS)*, 2004.

[48] Ron Kohavi and G.H. John. Wrappers for feature subset selection. *Artificial intelligence*, 97(1-2):273–324, 1997.

[49] Daphne Koller and M. Sahami. Toward optimal feature selection. In *13th International Conference on Machine Learning*, volume 1996, pages 284–292. Citeseer, 1995.

[50] Lothar Krempel. Visualizing networks with spring embedders: Two-mode and valued graphs. In *International Sunbelt Social Network Conference. Charleston, SC*, pages 36–45, 1999.

[51] Chris Lattner and Vikram Adve. Data structure analysis: An efficient context-sensitive heap analysis. Technical report, UIUC, 2003.

[52] Edda Leopold and Jörg Kindermann. Text categorization with support vector machines. How to represent texts in input space? *Machine Learning*, 46(1):423–444, 2002.

[53] Hua Li, Dou Shen, Benyu Zhang, Zheng Chen, and Qiang Yang. Adding Semantics to Email Clustering. In *Sixth International Conference on Data Mining (ICDM'06)*, pages 938–942. Ieee, December 2006.

[54] Huan Liu and Hiroshi Motoda, editors. *Computational methods of feature selection.* Data Mining and Knowledge Discovery Series. Chapman & Hall, August 2008.

[55] Huan Liu and Rudy Setiono. A probabilistic approach to feature selection- a filter solution. In *13th INTERNATIONAL CONFERENCE on MACHINE LEARNING*, pages 319–327. Morgan Kaufmann, 1996.

[56] Huan Liu and Lei Yu. Toward Integrating Feature Selection Algorithms for Classification and Clustering. *IEEE Trans. on Knowledge and Data Engineering*, 17(4):491–502, 2005.

[57] J.J. Luczkovich, S.P. Borgatti, J.C. Johnson, and M.G. Everett. Defining and measuring trophic role similarity in food webs using regular coloration. *Journal of Theoretical Biology*, 220:303–321, 2003.

[58] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, page 14, Berkley, 1967. University of California Press.

[59] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *An Introduction to Information Retrieval.* Cambridge University Press, Cambridge, England, draft edition, 2009.

[60] Tom M. Mitchell. *Machine learning.* McGraw-Hill, inc., New York, NY., May 1997.

[61] Pabitra Mitra, C A Murthy, and Sankar K Pal. Using Feature Similarity. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(3):301–312, 2002.

[62] James Moody, Daniel McFarland, and Skye Bender-deMoll. Dynamic Network Visualization. *American Journal of Sociology*, 110(4):1206–1241, January 2005.

[63] Christopher Z. Mooney and Robert D. Duval. *Bootstrapping: a Non Parametric Approach to Statistical Inference.* Sage publications, 1993.

[64] Andrew Y. Ng. On feature selection: learning with exponentially many irrelevant features as training examples. In *15th International Conference on Machine Learning*, pages 404–412. Morgan Kaufmann, 1998.

[65] K. Nigam, A.K. McCallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine learning*, 39(2):103–134, 2000.

[66] JS Okolica and GL Peterson. Using PLSI-U to detect insider threats by datamining e-mail. *International Journal of Security*, 3(2):114–121, 2008.

[67] T.N. Pappas. An adaptive clustering algorithm for image segmentation. *IEEE Transactions on Signal Processing*, 40(4):901–914, April 1992.

[68] Carey E. Priebe, John M. Conroy, David J. Marchette, and Youngser Park. scan statistics on enron graphs, 2010.

[69] 20 newsgroups data set Jason Rennie. 2008. MIT, CSAIL, March 07. 2012, `http://people.csail.mit.edu/jrennie/20Newsgroups/`

[70] J.B. Rosenberg. Geographical data structures compared: A study of data structures supporting region queries. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 4(1):53–67, 1985.

[71] Thomas P. Runarsson and Sæmundur Ó. Haraldsson, 2010. K-means *Computer program* December 2. 2011, `http://notendur.hi.is/~soh9`

[72] J. Skvoretz and K. Faust. Relations, species, and network structure. *Journal of social structure*, 3(3), 2002.

[73] S Stolfo, Shlomo Hershkop, and Ke Wang. Behavior profiling of email. In *1st NSF/NIJ Symposium on Intelligence & Security Informatics(ISI 2003)*, Tucson, Arizona, 2003.

[74] Salvatore J. Stolfo, Germán Creamer, and Shlomo Hershkop. A temporal based forensic analysis of electronic communication. *Proceedings of the 2006 national conference on Digital government research*, pages 23–24, 2006.

[75] Salvatore J. Stolfo, Shlomo Hershkop, Ke Wang, Olivier Nimeskern, and Chia-Wei Hu. A behavior-based approach to securing email systems. In *Mathematical Methods, Models and Architectures for Computer Networks Security*, pages 57–81. Springer Berlin / Heidelberg, 2003.

[76] S.J. Stolfo and S. Hershkop. Email mining toolkit supporting law enforcement forensic analyses. In *Proceedings of the 2005 national conference on Digital government research*, pages 221–222. Digital Government Society of North America, 2005.

[77] M.C. Su and C.H. Chou. A modified version of the K-means algorithm with a distance based on cluster symmetry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):674–680, 2001.

[78] D.L. Swets and J.J. Weng. Efficient content-based image retrieval using automatic feature selection. In *Computer Vision, 1995. Proceedings., International Symposium on*, pages 85–90. IEEE, 1995.

[79] EDRM Enron Email Data Set v2, 2012, The Electronic Discovery Reference Model(EDRM). June 16. 2012, `http://www.edrm.net/resources/data-sets/edrm-enron-email-data-set-v2`

[80] The Gephi Team. Gephi, an open source graph visualization and manipulation software, 2008.

[81] M Trier. Towards a social network intelligence tool for visual analysis of virtual communication networks. *Virtuelle Organisationen und Neue Medien*, (Cmc):331–342, 2006.

[82] Jean Vuillemin. A data structure for manipulating priority queues. *Communications of the ACM*, 21(4):309–315, 1978.

[83] Chun Wei, Alan Sprague, Gary Warner, and Anthony Skjellum. Mining spam email to identify common origins for forensic application. In *Proceedings of the 2008 ACM symposium on Applied computing - SAC '08*, pages 1433–1437, New York, New York, USA, 2008. ACM Press.

[84] Eric P. Xing, Michael I. Jordan, and Richard M. Karp. Feature selection for high-dimensional genomic microarray data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 601–608. Morgan Kaufmann, 2001.

[85] C.T. Zahn. Graph-Theoretical Methods for Detecting and Describing Gestalt Clusters. *IEEE Transactions on Computers*, C-20(1):68–86, January 1971.

A. Gephi importation files

Importing into Gephi can be accomplished by multiple means, this thesis uses files with the extension *.csv* (comma separated values). Listings A.1 and A.2 give an example of how these files look like. Columns are actually separated by a tab character and not a comma because the time interval reserves the comma with the format `<[start,end]>`. Every value of the *Id* column in the node file must be unique but labels can be the same. Source and target values in the arc file must be identical to one of the *Id* values.

```
Id                Label             Time Interval
.                 .                 .
.                 .                 .
.                 .                 .
gsullivan         gsullivan         <[87,237]>
mballases         mballases         <[304,306]>
robert.badeer     robert.badeer     <[176,348]>
frank.oldenhuis   frank.oldenhuis   <[175,248]>
kent.miller       kent.miller       <[175,248]>
marty_mcfadden    marty_mcfadden    <[324,349]>
julie.sarnowski   julie.sarnowski   <[346,348]>
steve.kean        steve.kean        <[248,250]>
tk.lohman         tk.lohman         <[175,298]>
.                 .                 .
.                 .                 .
.                 .                 .
```

Listing A.1: An example of a node file ready for importation to Gephi.

```
Source            Target            Time Interval
.                 .                 .
.                 .                 .
.                 .                 .
drew.fossum       martha.benner     <[18,19]>
drew.fossum       martha.benner     <[213,214]>
martha.benner     maria.pavlou      <[344,345]>
martha.benner     drew.fossum       <[344,345]>
drew.fossum       martha.benner     <[126,127]>
drew.fossum       bill.cordes       <[202,203]>
drew.fossum       mary.miller       <[202,203]>
drew.fossum       shelley.corman    <[202,203]>
drew.fossum       martha.benner     <[213,214]>
.                 .                 .
.                 .                 .
.                 .                 .
```

Listing A.2: An example of an edges file ready for importation to Gephi.