FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

MASTER THESIS

Bc. Michal Hušek

Text mining in social network analysis

Department of Theoretical Computer Science and Mathematical Logic

Supervisor of the master thesis: doc. RNDr. Iveta Mrázová, CSc.

Study programme: Informatics

Specialization: Software Systems

Prague 2018

I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 paragraph 1 of the Copyright Act.

In…..... date..... signature

Title: Text mining in social network analysis

Author: Bc. Michal Hušek

Department: Department of Theoretical Computer Science and Mathematical Logic

Supervisor: doc. RNDr. Iveta Mrázová, CSc., Department of Theoretical Computer Science and Mathematical Logic

Abstract: Nowadays, social networks represent one of the most important sources of valuable information. This work focuses on mining the data provided by social networks. Multiple data mining techniques are discussed and analysed in this work, namely, clustering, neural networks, ranking algorithms and histogram statistics. Most of the mentioned algorithms have been implemented and tested on real-world social network data and the obtained results have been mutually compared against each other whenever it made sense. For computationally demanding tasks, graphic processing units have been used in order to speed up calculations for vast amounts of data, e.g., during clustering. The performed tests have confirmed lower time requirements. All the performed analyses are, however, independent of the actually involved type of social network.

Keywords: data mining, social networks, clustering, neural networks, ranking algorithms, CUDA

Názov: Text mining in social network analysis

Autor: Bc. Michal Hušek

Katedra: Katedra teoretické informatiky a matematické logiky

Vedúca práce: doc. RNDr. Iveta Mrázová, CSc., Katedra teoretické informatiky a matematické logiky

Abstrakt: Sociálne siete sú v súčastnosti veľmi hodnotným zdrojom informácií. Táto práca sa zameriava na dolovanie dát pochádzajúcich zo sociálnych sietí. Pojednáva a analyzuje rôzne techniky dolovania dát, konkrétne klastrovanie, neurónové siete, hodnotiace algoritmy a histogramovú štatistiku. Väčšina zmienených algoritmov bola implementovaná a testovaná s dátami z reálnej sociálnej siete. V prípade, že to bolo zmyslupné, výsledky boli vzájomne porovnané. Pre výpočtovo náročné úlohy, konkrétne klastrovanie, boli použité grafické procesory na ich zrýchlenie. Testy takto upravených programov potvrdili nižšie časové nároky. Všetky vykonané analýzy sú však nezávislé na konkrétnej použitej sociálnej sieti.

Kľúčové slová: dolovanie dát, sociálne siete, klastrovanie, neurónové siete, hodnotiace algoritmy, CUDA

Acknowledgements

In the first place, I would like to thank my supervisor doc. RNDr. Iveta Mrázová, CSc for all the help and time spend with my work. I would like to especially thank her for her patience with my not always sufficient progress and my very limited time schedule.
In the second place, I would like to thank my faculty for providing me with the necessary software and hardware. Without them, it would be difficult to finish this work.

Table of Contents

1. Introduction

1.1 Motivation

Nowadays, publically available data grows with enormous speed. Almost every human activity creates a significant trace of digital records. Computers, cameras, sensors etc. are basically everywhere. Everything is digitalized today.
People communicate so often as never before and generate digital records. Many of us create and own digital content, take photos, write blogs, comment on articles, compose music and make many more different kinds of digital data.
Companies use a lot of software applications, which generate the data and logs about anything that happens.
In the third quarter of 2017, Facebook had 2.07 billions of monthly active users, 1.37 billions of them are active on daily basis. About 300 millions of photos are uploaded daily on Facebook.
Every 60 seconds, 510,000 comments are posted, 293,000 statuses are updated, and 136,000 photos are uploaded on Facebook. That is a good example of data growth and importance of services like Facebook.[1]

Recently, the phenomenon, which moves the digital world, represents the social networks. Social networks have changed from simple applications oriented towards chatting and sharing funny pictures into a means shaping digital façade of each person, company, political party etc. People do not print their photos or use specialized digital galleries, they post them on social networks. People make less and less phone calls and rather send a message on social network. Instead of reading newspapers, many people just open Twitter.
Various political parties have risen purely from the social networks, for example the "pirate parties" originally from Sweden. The success of Donald Trump in the race for president of the US was partially achieved using social networks. Most of the most influential media in the USA were not supporting him but 45 millions of people follow his Twitter account.[2]
There were revolutions organized using only social networks, for example the controversial revolutions called "Arabic spring" which started in the year 2010.

However, social networks are not only well known services like Facebook or Twitter. There are many other kinds of social networks. I can create social network from any set of data, where I can identify patterns and relationships between them, not just persons and their communication. I can for example describe the Milky Way galaxy as a social network. The planets, stars, asteroids, etc. are the patterns and gravity between them represents the relations. I can use the supply chain of a company as a social network. Companies are the patterns and supply/demand between them are the relations.

The most popular crypto currency right now is probably Bitcoin and its derivations. But a much more important "currency" of this age are rather the data. People do not need to pay for services like Facebook or Instagram with money. They pay with their personal data they create instead. This data has an enormous business value. If we understand each person, we can namely offer and serve them personalized content and advertisement. Companies are ready to pay well for personalized advertisement.

But collecting these data is just the first step. It is necessary to understand it, find the important information there. The scientific discipline focused on retrieving this hidden information from data is called data mining.

Data mining is the scientific discipline focused on the retrieving new information from the existing data sources. Data mining tries to find previously hidden relationships, attributes, patterns etc. in the available data sets.
In many scientific disciplines, it is common to perform measurements for different experiment, to collect data prepared for next examination and further experiments. However, data mining relies only on the data already created for the different purpose. Data has been mostly collected without the intent of scientific processing or data mining. There are namely many types of data, where data mining can discover some interesting hidden information. Such data can be often of business value that can grow by data mining.

In this work, I do not plan to focus on one particular social network and the way it works. The social networks and their APIs change frequently and it would be limiting to stick to an outdated social network API.
Instead, I would like to examine the model of social networks from a general point of view. Naturally, I need to process the data from some particular social network, but I would like to make my calculations and experiments independent of it. With little effort, it should be possible to replace my data with the data from a different social network and repeat the experiments.

1.2 Methodology

The outcome of this work will be a set of scripts based on the various algorithms like neural networks, ranking algorithms and clustering. This set will be based on Enron email corpus, which I have chosen as the input for my experiments. The algorithms and ideas are usable for many other types of social network, but the scripts cannot be used for a different data without modifying the part processing the input data. The user interface will be simple, so the user understanding the data mining ideas can use it without need to understand my implementation, transformations, etc.
The scripts and the description of the correct settings will be part of this work, so it would be possible to repeat the experiments, with the sufficient hardware and software.

1.3 Thesis structure

This work describes the used algorithms in separate chapters. There are more chapters describing the clustering experiments, because there are more clustering experiments and 2 different implementations are used, CPU and GPU based.

Clustering distributes similar data into clusters according to common attributes. Cluster membership of the data can help to explain hidden attributes. There are many different ways how to perform clustering, many different options how to prepare the data, and what is actually represented by them. The result of clustering experiments is also significantly influenced by the choice of attributes involved in clustering. It this work, I use 2 main views on the data in the clustering experiments. The first one takes each message as the independent pattern and clusters them based on the words used in the message. The second approach considers the persons involved in the

communication as the patterns and clusters them using the frequency of their mutual communication or the dates of the email messages.

To sort the persons in my communication and find the most important ones, I have performed the histogram statistics experiment. This experiment helped to leave out less important persons in the communication and let me focus on the more important ones. The communication of the most important persons can be visualized in the histogram statistics matrix.

Another interesting way how to mine data is to use neural networks. Neural networks are suitable for prediction of unknown communication links. In this work, I use 2 types of neural networks: Hopfield net and bidirectional associative memory. In my experiments, I use these networks to predict the links between the patterns based on the known previous communication. As the input of this algorithm, I have used the data preprocessed in the histogram statistics experiment. I could examine the behavior and capacity of these models with my data based on the real communication.

The next algorithms used in my work are ranking algorithms, such as PageRank and HITS. These algorithms were developed to rank the documents in the graph based on World Wide Web pages, but when modified for my needs, it can help me to find the most important persons in my social network data. These experiments also use the data from histogram statistics experiment as the input. I can compare the results from the ranking algorithms with the results of the less sophisticated histogram statistics. The motivation of both experiments was to find the most important nodes in the given set.

In this work, I examine Enron email corpus. This corpus contains emails from the bankrupted company Enron. Data from Enron were collected during the investigation of its fall, but later were released to public.
Each email message is in separated text file. I can construct social network from this data and perform my experiments on it.

1.4 Objectives of this work

In this work, I want to construct social network based on the Enron email corpus. Then I would like to implement the algorithms mentioned in the previous paragraph and mine the Enron social network with them.
I want to cluster the patterns of Enron social network and examine the reasons why the patterns are grouped in the particular clusters.
I would like to find the most important patterns in the set, using various techniques like ranking algorithms or histogram statistics based on the strength of the relationships between them.
I want to examine the existing algorithms and try to develop the implementation with better performance than the original algorithm.
In the end, I want to evaluate the results and propose an ideas about the found relationships, attributes etc.

2. K-means algorithm

2.1 Motivation

In the previous chapter, I have described some algorithms applicable to social networks. One of the most important areas seems to be clustering. I have chosen clustering as one of the first types of experiments with my data. There are many types of the clustering algorithms I could use. But I have decided to use K-means algorithm. K-means is a relatively simple algorithm to be implemented, debugged or modified if necessary. It offers me many possibilities how to define the attributes I would like to use for clustering. I think it would be good algorithm to start my experiments and it could help me to discover some interesting attributes based on the distribution into the clusters by K-means algorithm.

2.2 Description of the algorithm

This description of K-means algorithm is based on the original work of MacQueen[3] and inspired by "Tutorial on clustering algorithms"[4].
K-means belongs to the traditional clustering algorithms. It was described by MacQueen[3] in 1967. K-means belongs to the family of algorithms based on unsupervised learning. It is one of the easiest clustering algorithms. It was originally developed for the signal processing. But it can be used in many clustering problems including mine.

Formula 1: Formal definition of k-means [5]:

(1)

Given a set of n points S = {p_1 …p_n} in m dimensions, find a set of points B= {b_1… b_k} such that:

$$\sum_{i=1}^{n} [d(pi, B)]^2$$

is minimized. This minimum value is denoted Opt(S,k). Here d(pi,B) is the Euclidian distance from pi to the nearest point in B, d(pi,B) = $\min_{1<=j<=k}$ d(pi,bj)

K-means algorithm divides the set of input patterns into *k* clusters. This *k* is chosen by the programmer.
Finding the ideal *k* can be a NP- complete problem[5]. But there is no known NP-hardness result when the dimension m is fixed and k, the number of clusters, is part of the input. Otherwise, when I do not know the k and m is not fixed, finding ideal k is NP- hard.
The final result of clustering significantly relies on the right choice of *k*. The programmer should have some hypothesis about the possible result. If not, it is still possible to guess this number or make more experiments with different *k* and then compare the results.

Each cluster contains a so-called centroid. Centroid is hypothetical point in the centre of the each cluster. It is calculated from the positions of all the members of the cluster.
It is possible for the cluster member to become centroid, but only by coincidence. Centroids are mostly just calculated somewhere in the space, in the center of all the members of the particular cluster.

The input of this algorithm is the set of n input patterns *S* and number of clusters *k* >1

Algorithm 1 : Implementation of the K-means algorithm used in this work

S: set of input patterns
k: number of clusters
i: initial number of patterns in one cluster
C: set of clusters
c: one cluster from C
p: one pattern from c

-create the set of k empty clusters C
-divide the size of S by k to achieve the initial number of nodes i
-distribute i nodes into each cluster from C
-place the rest of the nodes from S into random cluster (for example 1st cluster)

-repeat until the number of exchanges between clusters is 0
    -calculate centroid for each cluster in C according to Formula 2
    -for each cluster c in C
        -for each pattern p in c
            -find the closest centroid to p, in any cluster from C
            -move the p to the cluster with closest centroid
            -if the best fitting cluster does not equal to cluster c, increment the counter of exchanges

The output of this algorithm is the set of *k* clusters each containing the subset of patterns from *S*. Each pattern s from S belongs to the cluster c from C with closest centroid.
There are more possible ways how to initialize the clusters. The centroids can be randomly initialized before the first points are assigned to the cluster. And then the first distribution is not random but regular, based on the distance of points to centroids.

2.3 Calculation of the centroid

Calculation of the centroid of each cluster is an important part of the whole calculation and clustering. Centroid is calculated as the centre of the cluster. It does not have to fit to any of patterns in the cluster.

Formula 2: calculation of centroid in each cluster in K-means

(2)

Centroid is a vector $V_k = (v_{k1}, \ldots, v_{km})$ Where 1<=k<=K, m>0 , m is the number of dimensions. → xi is the vector representing each node i in cluster c.

$$V_{ki} = \frac{1}{|Vk|} * \sum_{x \in Vk} xi$$

2.4 Conclusion

In this chapter, I have briefly described the K-means algorithm, which is going to be used in my experiments. As I have written before, K-means is simple algorithm, which makes it well suitable for me as the first algorithm to start with. The particular use is described in the next chapters in detail.

3. Enron email data set

3.1 Motivation

This work focuses on the study of the social networks. I need to have some social network to use in the experiments. There are many possible definitions of social networks. This term nowadays associates mostly with the Internet services, but a social network can be also group of people in some relationship, having some interaction, communication etc. Examining these so-called natural social networks would be possible, but I still need some input data, something to use in my experiments. So focusing on the digital social networks or at least digital records or structures of social networks would be the simplest solution. One of the most well known Internet-based social networks are Facebook or Twitter. It would be beneficial to examine these networks. But it can be difficult to obtain the necessary data for the various algorithms I would like to use during my experiments.

Although using the data from Facebook or Twitter would be an interesting option, it is not the only way how to examine the social networks. The algorithms used in this work are not Facebook specific or Twitter specific. They mostly process the data represented as the oriented graph and their original source and format are not important. The most important thing is to use the real, man-made social network as the input for my algorithms. It can be the real commercial social network, email based social network, some Internet forum etc. Any kind of input data should be transformed to the format friendly for my algorithms.

For my experiments, I have decided to use the Enron email data set.

3.2 Enron company

This paragraph is based on the history of Enron published on the Investopedia[6] web page.

Enron was one of the biggest companies in United States of America focused on energy as well as other commodities and services connected to them. It was founded in 1985.

Enron belonged to the biggest and richest companies in the USA by the end of the 20th century. Despite these facts, Enron filed for unexpected bankruptcy by the end of the year 2001. There were more reasons for this enormous fall. One of them was the fraudulent system of accounting. When the company would build an asset, for example a power plant, the expected profit would be immediately projected to the books, even if they did not have any profit from it yet. If the real profit was lower than expected, the company would not take the loss. Instead of that, Enron would transfer this asset to an off the book corporation where the loss would go unreported. This fraud accounting made the false environment, where Enron would never take any loss and lower or no profits cannot hurt it.

This schema was working for a surprisingly long time, mostly because of the CEO of Enron- Jeffry Skilling. He had very good contacts to influential people from Wall Street and the whole financial business. Enron appeared as a booming company for a long time, despite the growing looses. But this schema could not work forever. In April 2001, many analytics started to question the finance and real profitability of Enron.

The main problems of Enron started during summer 2001. In autumn 2001, Enron reported the first official quarter loss. But the reality was much worse. Soon after that, the majority of the board resigned. The finance and accounts of Enron were investigated.
The things only got worse and Enron filed for bankruptcy on 2nd of December 2001. Enron had losses of $591 million and had $628 million in debt, by the end of 2000.
Enron became the world-known example of the wrong leadership. The managers of Enron were over self-confident and obsessed by the so-called black numbers. They were able to cheat and fake anything, to see the profit in the books of Enron.
This huge scandal was investigated by the police in USA. During the investigation, most of the documents were confiscated, including the digital ones. Part of the confiscated digital documents was the full record of email communication of the most important managers and employees of Enron. This set forms an interesting social network. All these data, including the emails were released to the public, after the investigation was completed. Many researchers used these data for their experiments. I have decided to use these data for my experiments as well.

3.3 Enron email corpus

I have downloaded the Enron emails from the following web page: https://www.cs.cmu.edu/~./enron/. It is a web page of the Carnegie Mellon University.
The emails are stored in the plain text files, each email in one file. The files are organized to folders. The corpus contains the emails of about 150 people, mostly senior managers of Enron. In the corpus, there are about half a million of emails, which should be enough to reconstruct the social network between these persons. This dataset was collected and prepared by the CALO project (A Cognitive Assistant that Learns and Organizes). The data of the Enron email corpus were also cleaned and prepared by the researchers from MIT (Massachusetts Institute of Technology). This data was originally made public and posted on the web by Federal Regulatory Commission during its investigation. This data was used for many other research projects before.
There are many possible ways, where the study of Enron email corpus can be useful. First of all, it is of course a good source when attempting to understand what has really happened in Enron. It was the original motivation for collecting all these email by the investigators. This enormous set of emails is also a good material to study human behaviour and communication during crisis. We have the real record of the communication of the top management one of the biggest US companies before its bankruptcy. Another possible use is the study of natural language and text mining, because this data is generated by humans in the real communication. We can also look at this email corpus simply like at a record of a natural social network and study it from this point of view. This is the way I would like to use this data. There are 517,388 plain text files in 3,499 folders.

3.4 The folder structure of Enron email corpus

The Enron email corpus is distributed in the zip file. It contains mostly 2-level folder structure. On the first level, the folders represent the managers of Enron whose emails were confiscated for investigation. It does not mean that the corpus contains only the emails from and to these 150 former employees. It is more like we have all

the emails from the email accounts of these 150 people. So these folders contain emails from different Enron employees, emails from addresses outside of Enron, forwarded messages, etc. The folders on the first level are named in the following way: [surname]-[first letter of first name].
This naming convention looked very useful in the beginning. My naive idea was, that I can use these names of folders for creating the final set of patterns, containing all the persons involved in the communication. Each folder means one person and one person means one pattern. After a more detailed examination, I have realized, that this view on the folders is very simplified. For the more detailed examination about Enron, more nodes than just the owners of the folders are necessary.
The names of the folders on the second level vary in each first level folder. They are probably the names of the folders inside the email account of each person. The names of these folders also looked very useful at the beginning. Most of the first-level folders contain subfolders with the names like "inbox" etc. So from the first naïve look, I was thinking it would be possible to use these names to determine for example send or received emails. But I have soon realised, that it was very naïve idea. The second-level folders were all user organized so I cannot expect any rules strict enough to be able to use these names in my experiments. I have decided to completely ignore these folder names in my experiments. In some experiments, I have decided to ignore the whole folder structure.

3.5 The structure of the email message

As I have written above, the emails are stored in Enron email corpus as plain text files, each email in one file. These messages do not contain only the text, but also all the technical fields of the standard email message. All the email text files use UTF-8 encoding, which makes them well readable and usable.

Example 1 : The example of the message from the Enron email corpus

```
Message-ID: <28704291.1075859174954.JavaMail.evans@thyme>
Date: Mon, 5 Nov 2001 07:08:06 -0800 (PST)
From: hai.chen@enron.com
To: harry.arora@enron.com
Subject: vol book spreadsheets
Mime-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit
X-From: Chen, Hai </O=ENRON/OU=NA/CN=RECIPIENTS/CN=HCHEN2>
X-To: Arora, Harry </O=ENRON/OU=NA/CN=RECIPIENTS/CN=Harora>
X-cc:
X-bcc:
X-Folder: \Harry_Arora_Jan2002\Arora, Harry\Inbox
X-Origin: Arora-H
X-FileName: harora (Non-Privileged).pst

Files are bigger than the limit, so I am trying one by one.

Harry,  I don't have access to M:\power2\options.  I'll request the setup, and here
are the spreadsheets for vol book.
```

Each email message contains the following fields:

Message ID
This field contains the identification number of each message. The unique ID can be useful in the identification of each message. But it does not contain any information useful for clustering of classification. This field also contains the string identifying the client program used to create this message. But most of the programs are the same, probably because of some internal policy of Enron, so once again; this field does not hold any important information for my experiments.

Date
This field contains the date and time of the creation of the message by the sender.
This date-time is in the format:
[day of the week], [number of day] [month] [year] [hours]:[minutes]:[seconds] [time zone]
This format is well readable by a human, but the usage by program could be more complicated because of mixed string and number expressions. The date is very useful information for my clustering or classification experiments and it is used in them.

From
This field is the standard part of every email message. It contains the email address of the sender of the message. This field is used to determine the owner and creator of the message. It is much more reliable than the name of the folder where the email is placed. This field is used in all my experiments, because the sender is the crucial information about the email message.

To
This field is also the standard part of each email message. It contains the email address or addresses of the receiver or receivers. This field is also very important for my experiments, because it determines the receiver or receivers of the message. This information is also the crucial one and it is used in all my experiments.

Subject
Subject field contains a simple string with the subject of the email. When I look on the Enron email corpus like on the simple social network to be represented as a graph, this field is not important. But once I start to look at the content of the messages, subject is maybe the most important field for understanding the topic of the message. It also helps to put the email in a wider context, for example in some thread conversation. In text analysis, the subject is also a good field where to start.

Technical fields
The next fields are Mime, Content-type and Content-Transfer-Encoding. These fields contain just technical information and I do not see any use for them in my experiments right now.

Content
The last part of each email message is the content itself. This content is a plain text.
In the experiments with the graph-like social network, this field it not used. On the other hand, when examining the texts, data mining and advanced clustering, the content of the email is an important source of information. It is maybe the most important part in the sentiment analysis for example.

3.6 Enron email corpus as a social network

While working with the data downloaded from Facebook or Twitter, it is simple to imagine them as the representation of the part of some social network. But in this work, only emails are used for examining social networks. But any record of human long-term communication can be reconstructed as a social network and examined in this way. It does not even have to be a digital communication, but some digital record is necessary. Enron email corpus needs some necessary transformation before it can be used in any experiments. But this transformation differs for each algorithm, so it is described separately in each chapter.

One of the most basic descriptions of the social network is the graph description. This graph consists of nodes and links between them.

Nodes

A node is a basic element of a social network represented by a graph. Without nodes, there is no social network. Each node represents a person involved in the social network communication. In the Enron data set, there are more possibilities how to define the node. One of them is to use the former 150 Enron senior managers. This approach was used in my first, less successful, clustering implementation. However, it can be interesting not to reduce the number of nodes to 150. Each person included in the Enron email communication can then serve as a node. This would give me a much more detailed social network at the cost of its rapid expansion.

Links

The links represent a very important part of social networks described as a graph. The links indicate that the nodes are in some kind of relationship. This relationship can be very different, but the most used kind of relationship in this work is the simple communication. So when there is an edge (link) between node A and B, it means these nodes are communicating. These links can be one-way oriented or both way links. They can indicate that both nodes were equally communicating, or for example when one node was just the sender and second one just the receiver.

One-way link

It is possible to continue and find more objects and relationships similar to the objects and relationships known from social networks like Facebook or Twitter. One of them is follow only (one-way) link.

On Twitter, the basic link between 2 nodes is a one-way follow link.

Later, this one-way functionality was implemented for Facebook as well. The follow link serves for connecting two nodes with different levels of importance. Its typical use is between a celebrity and its supporter. The supporter wants to see all the posts from the celebrity, but the celebrity does not want to see any posts form its supporters. In the Enron data set, there is no technical way to make this restriction to create follow-only links. But we can assume the one-way communication between higher rank employees and lower rank employees like this type of link. It can be typically used for bulk email sending. Later, we can see many examples when one person in communication sends a significant amount of emails to another one, but not vice versa.

Two-way link

A two-way is the most basic connection found in the Enron email corpus. It simply indicates that two chosen nodes send at least one message to each other. This link

will be often used in the experiments described in the following chapters. A two-way link is the basic link for example for Facebook. Both nodes are of an equal status and their communication flows both ways more or less equally.

Groups

The group communication is another typical part of the social networks. A subset of nodes often needs to communicate without considering the rest of the nodes. For example on Facebook, the groups are strictly defined, created and maintained. However, there are many possible ways how to perform group communication on Facebook.

In the Enron email corpus, there is no strict definition of a group. It is just a huge set of emails and no strict definition of the group is even possible. But according to the research of Wilson Garnett we can find some naturally created groups of communicating nodes.[7]

These groups are comparable to groups from ordinary social networks.

3.7 Emails containing possibly important information

First, it is important to determine which emails contain important information. The bulk email about company teambuilding is probably not an interesting source.

One of the possibly important information is the time. The time when the email was sent and the delay until its receiver replied. For example, when one receiver had received 2 different messages at the same time and replied to the first one immediately, but it took him a few hours to respond to the second one. Then we can assume, the first message was much more important than the second one. Now, we also know the whole history of Enron and story leading to bankruptcy. So we can pay a higher attention to emails sent on some critical days or periods. We know when the important meetings were organized, when important news were announced.

Another possibly interesting attribute could be the position of the communicating persons in the company. The emails from a CEO could be more interesting for opinion mining than emails of someone new on the lowest company rank.

The dataset contains data from a longer period, so we do not need to follow the persons with an important position during one chosen period of time. We can for example include the emails of a person who reached an important position later. We can also take into account forwarded emails. When a message was forwarded many times, it is possible, that it contained information, which was important. The next indicator of importance could be the email frequency. A rare email communication could be interesting. For example, when a CEO writes an email to a lower level manager once a year, it probably contains important information.

And it is valid also vice versa. Mid-level managers do not write reports to CEOs on a daily basis.

The work of Garnett Wilson and Wolfgang Banzhaf[7] mentions the creation of some strongly connected components of communication graphs during crisis. These components were mostly created between the members of senior staff and more important managers, so information they exchanged could be again important to consider. Another reason is, that the members of these unofficial groups were probably the persons trying to solve the crisis, the persons taking the most important decisions for Enron. So their emails are also expected to contain important information. On the other hand the same people probably caused the crisis of Enron, so that is another reason why their messages should be important. The next point of

view on the Enron data is via fraud and corruption. Emails about these activities are usually not sent to all employees, so we can expect and detect small groups again.
For these reasons, we can simply conclude, that all the small private discussion groups could contain important information for opinion mining.

3.8 Conclusion

In this chapter, I summarily analysed the Enron email corpus from the point of the view of the social network. I want to use this data, because it is a record of real human communication in a changing situation while solving various serious problems. I have figured out, that this email set can be interpreted as a natural social network. I have identified the most important signs of the social network in this data. Enron email corpus consists of a set of nodes and edges while for the nodes, there can be different types of potential relationships. That indicates, that this data can be used as the material for the study of the social networks. I plan to use it for different experiments based on the social networks.
However, the current format of the data is not very convenient. The email corpus contains simply just the email messages, not meant to be processed in this way. It is necessary to make some data transformations before processing this data. These transformations are described in the following chapters.

4. Email - based clustering of the Enron data set

4.1 Motivation

The Enron email corpus contains potentially many interesting information. Many of them can be found by understanding the story behind Enron, the people and their communication. It is not difficult to find the names of the persons with the highest company rank and then examine their communication. It is also simple to find the important dates in the history of the Enron Company and try to find the communication related to these events. But all these kinds of approach bring very little additional information about Enron. It is necessary to know the important attributes like the company rank or the date before and the experiment can only confirm some hypothesis about Enron. Using these attributes would also bring me only very limited information about the Enron as the social network.

I wanted to find some previously hidden relationships or attributes in the Enron data. During this experiment, I also wanted to use these data as the record of the social network, not just as the record of some unordered communication of the people in the critical circumstances.

As the first experiment, I have decided to cluster these data. Clustering is a good approach to discover the hidden or previously undiscovered relationships between data. The results can point on the previously unnoticed common attributes of the potential clusters. Clustering can be also the good starting point for another future experiments. Discovering the attributes common for each cluster can help for example with some classification experiments. In the next experiments, this knowledge can be used for example for the prediction of the attributes.

4.2 Vector space model

The data I am using are textual, representing communication between the ordinary people. The authors hardly counted with the possibility, that their emails will be used for scientific experiments and social network studies. The format of these data is not suited for any computer processing. It is just a plain text enriched by the information necessary for the email communication. I need the way to represent these textual messages in more technical form. It is important to find some abstraction from the email messages and create unified patterns suitable for processing by my scripts. For this task, I have decided to use Vector space model.

Vector space model is the well known way how to represent text documents or any objects as the vectors of specific values. There are many options how to obtain these values and this approach is usable in many different situations. Vector space model was first introduced by G. Salton in 1975.[8]

In my experiments, the input data are the emails in text documents. There are more possible ways how to cluster these emails but I have decided to use frequencies of the words in the bodies of the emails.

4.3 Description of the used approach

In this experiment, I would like to use emails as patterns. Also, I would like to cluster these emails based on the words used in the body of each email.

Specification of the terms

Cnt counter: This structure is a list of all the words that occur in the body of the particular email message. It contains all the distinct words from that particular email. Each email-based pattern based on email has its own counter. Its name is counter, because it also holds the number of occurrences of the each word from the list. The Python collection counter is also used. It is stored in the object representing the email message in the clustering experiment.

Word vector: It is a list containing the distinct words from all the counter lists. The length depends on the number of unique in all the counters of all the emails. The order of the words in word vector is very important.
The particular vector of each email does not contain the words anymore. There are just zeros when there was no such word in the email counter words or the TF-IDF value if yes. These numbers are ordered in the same way as the word vector and the order is always the same.

Choice of the patterns

In this experiment, I have chosen email-message granularity. I have decided to count every single email message as the particular pattern to be clustered. This approach produces significant number of patterns. The total count is 517388. It is a challenge to process these data correctly within a reasonable time.
The patterns based on each particular email should let me examine the hidden relationships in the data. By relationship, it is usually meant the situation when 2 persons communicate. But there can be many other kinds of relationships. For example there can be a relationship between the most common words in the email message and its sender or receiver. Also it can be possible to determine the topic of the message based just on the words from the body of the email. There can be relationships between the topic or words and the date when the particular email was created. I wanted to be able to discover these hidden relationships and be able to use them in the next experiments.

TF-IDF statistics

TF-IDF statistics is one of possible ways, how to calculate the weights for the clustering based on the frequency of the words in the particular text document.[8] TFIDF means term frequency–inverse document frequency. This coefficient is calculated to reflect the importance of the word in the particular document and also in the whole set of documents. TFIDF is a good way how not to get the results corrupted by the results from some irregular documents, or too short documents, etc.

The TF (term frequency) of the particular word is calculated in the following way:

Formula 3: Calculation of TF value

$$TF(w_d) = \frac{f(w_d)}{|d|} \quad (3)$$

Where d is the particular document from corpus C,
W is the particular word from document d,
$TF(w_d)$ is the TF value of word w in document d,
$f(w_d)$ is the number of occurrences of word w in document d,
|d| is the number of words in document d

The IDF (t inverse document frequency) of the particular word is calculated in the following way:

Formula 4: Calculation of IDF value

$$IDF(w, C) = log \frac{|C|}{k} \quad (4)$$

Where C is the collection of documents,
W is the particular word,
IDF(w,C) is the IDF value for word w in corpus C,
|C| is the total number of documents in C and
k (k<=|C|) is the number of documents containing the word w

The whole TF-IDF value is calculated in the following way:

Formula 5: Calculation of TF-IDF value

$$TFIDF(w, C) = TF(w_d) * IDF(w, C) \quad (5)$$

Stop words

The email-based clustering experiment was successfully clustering the emails using their most frequent words. But a significant proportion of the most frequent words were the words without any particular value for data mining. They were typically articles, prepositions etc. It is technically correct, because these words are naturally very frequent in the English language. But they have little or no value for data mining. The most interesting words for data mining are nouns and verbs. I have decided to try to filter out the words with no or low data mining value. This idea is quite common in text mining. It means I do not need to construct the list of unwanted words by myself. I can use some of the already existing lists. All the Enron communication is in English and there are many existing lists of stop words for English language already. I have chosen the list called Terrier stop list[9] prepared by a member of Bitbucked community Kavita Ganesan. This list contains 733 possibly unwanted words, which is enough to make my results more reasonable. This list is available for free download as the simple plain text file, which fits my needs.

The attributes used in the clustering experiment

In this clustering experiment, I have focused on the content of the email body of each message. Instead of some aggregated statistical values about the patterns, I have simply created the list of all the words in the body of each email message with the corresponding TF-IDF value based on the number of their occurrences. The main idea is, that people solving the same problems, or responding to some message in the same thread may often use the same words. The same people can also use the same words often, which can make me able to identify the sender of the email just from the

text. It is also possible, that the people who know each other for longer time, co-workers or friends may use similar vocabulary. Any of these ideas can lead to interesting clustering results.

Other collected attributes

While pre-processing the patterns to be clustered, more than just words are collected. There are many other attributes for each mail, which are read by the program and stored to the object representing each pattern. But their value is not used in the clustering vector unique for each pattern. This vector contains only the TF-IDF values for each word from the body of the particular email. However, these attributes are useful while reading the email and later for presenting the representative values of each cluster.

These attributes are following ones:

Table 1: Attributes not used in clustering experiment

Name of the attribute	**Description**
Sender	Sender of the email message
Receivers	The list of all the receivers
Subject	The whole subject of email stored in one string
Body	The whole body of email stored in one string
Length of subject	Number of character in the subject
Length of body	Number of characters in the body
Type of email	If the email is a standard one, a reply or a forwarded one

Creation of the input patterns

An input pattern is formed to each of the messages in the Enron email set. It means each message should be opened, read and closed. This causes a significant slowdown of the whole program, because opening, reading and closing files can be a costly operation. The input of the following algorithm is the whole Enron email set. The output is the set of patterns.

Algorithm 2: Creation of the input patterns

S: set of patterns
f: folder from Enron email corpus
m: one email from f
n: pattern representing m

```
-create an empty set of patterns S
-for each folder f from the Enron email set
        - for each message m in folder f
                -read the sender of the message m
                -if there is just one receiver
                        -read the receiver
                -else iterate receivers
                        -read each receiver
                -determine the type of the message m
                -read the body of the email
                -split the body into words
                -count the length of the body
                -create the counter of all the words in body
        -create the pattern n representing the email
        -insert n into S
```

The word vector is the list of all the words present in all the bodies of emails.
The counter of words in the body of the email is the most important clustering attribute in this experiment. But in this form, this list is not suitable for clustering. It could be maybe clustered in the modified way. But I did not want to use it. I wanted to use the standard clustering methods. It means that the transformation of this attribute was necessary.

Algorithm 3: Calculating the clustering vector

S: set of all the patterns
n: one pattern from S
v: word vector
cnt counter : (see specification of terms)

```
- for each pattern n in the set of patterns S
        -create the empty vector v of the length of word vector
        -compare the words in word vector with the words from counter cnt of each
node  n
                -when the word from the counter is found in word vector, the TF-IDF
                value is  written to vector v in the same position as is the position of
                the particular word in the word vector
-insert the vector v into node  n
```

Clustering
Clustering of these patterns works according to the description in the chapter about k-means clustering. It was one of the goals of this experiment. The clustering stops,

when the nodes stop being moved between the clusters. The algorithm takes a significant amount of time, because of the length of the vectors, that need to be iterated, compared etc. in each calculation.

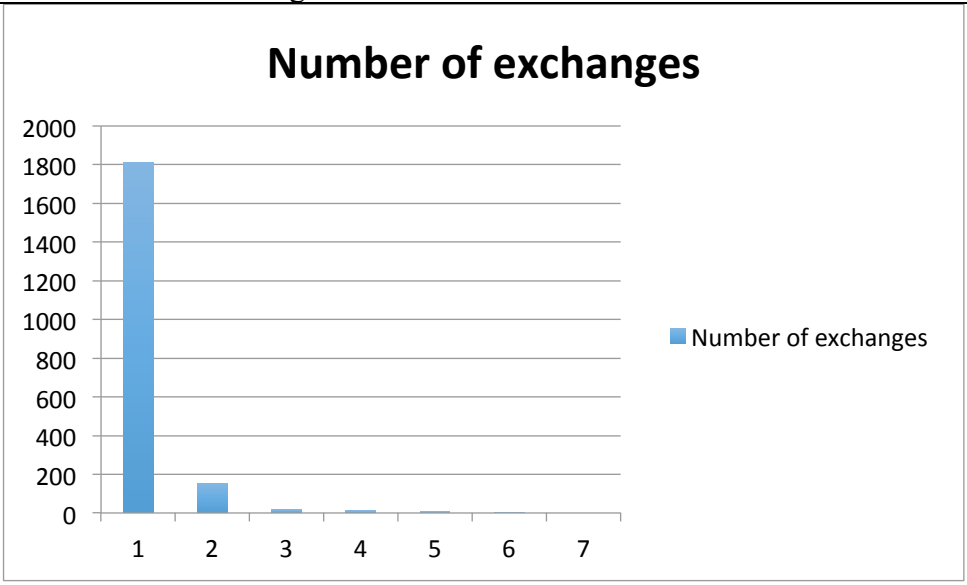
4.4 Test of the clustering on a small subset of the input data

I have performed my first clustering experiment with the email- based algorithm to test it and prove the idea that it can work. In the beginning, I was afraid that the sparse vectors, mostly filled with zeros could cause problems during clustering. I had similar problems during the experiments with the Bidirectional associative memory later. But the information contained in the vectors seems to be sufficient for division of the data into clusters.

I did not use the whole Enron data set for this experiment. I have used just its small subset, messages contained in one of the subfolders only. The chosen subfolder was the subfolder of Phillip Allen.

I was observing the way how the clustering worked to understand if the chosen attributes, way of clustering etc. was a correct one. In the following graph, there is shown the number of pattern exchanges between the clusters. We can see how the number of exchanges dropped after the 1st iteration dramatically. This is logical, because in the beginning, the placement of the patterns into clusters was random. Since the 2nd iteration, the algorithm was already distributing the patterns according to the centroids of clusters.

Graph 1: number of exchanges between clusters in each iteration



4.5 Results of the clustering using a subset of the input data

In every experiment, it is important to be able to read and interpret the results. It is possible to obtain many types of data representing the clusters in this experiment. I focused on the information connected to the examination of the social networks. The following attributes were collected from the clusters:

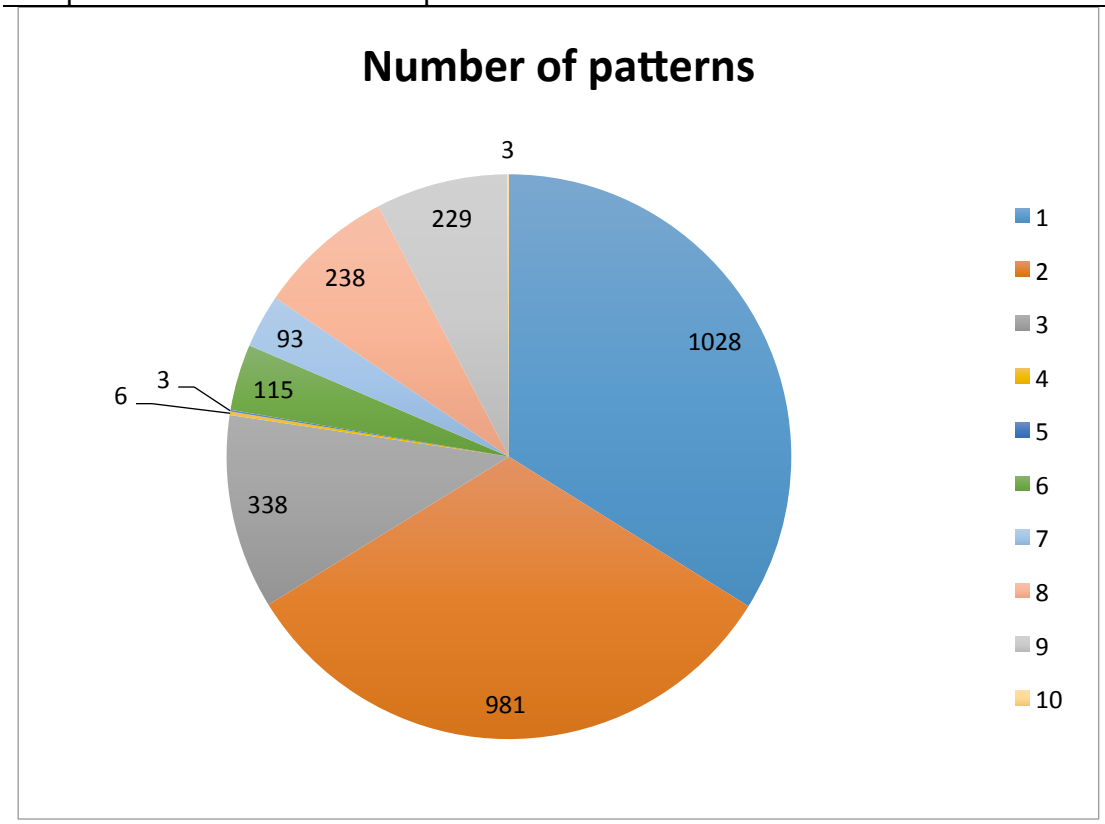
Number of patterns

This attribute is probably the simplest one; it is a count of the patterns in each cluster. This information does not contain much value for the data mining, but it is useful for debugging of the program and checking its correctness. While clustering the data based on the real social network, it is very rare when all the patterns fit to 1 or 2 clusters only. So checking the numbers of the patterns in the clusters can help me to discover the problems with the data, program, chosen attributes etc.

Table 2: Number of patterns in each cluster

Cluster ID	Number of patterns
0	1028
1	981
2	338
3	6
4	3
5	115
6	93
7	238
8	229
9	3

Graph 2: the distribution of the patterns in the clusters



Number of primary emails

This attribute is the count of the patterns based on the primary emails in the particular cluster. Primary email is neither an email that was forwarded nor it is a reply email. This attribute can be useful when examining the most frequent words

and phrases in the patterns of the cluster. With a low number of primary emails in the cluster, it is possible that the high number of occurrences of some words is based on the fact that the cluster contains many times forwarded emails or reply emails. With the high number of primary emails, the results are becoming more relevant and interesting.

Number of reply emails
The next attribute is the count of the patterns based on the reply emails. This attribute is closely connected to the attribute of primary emails.

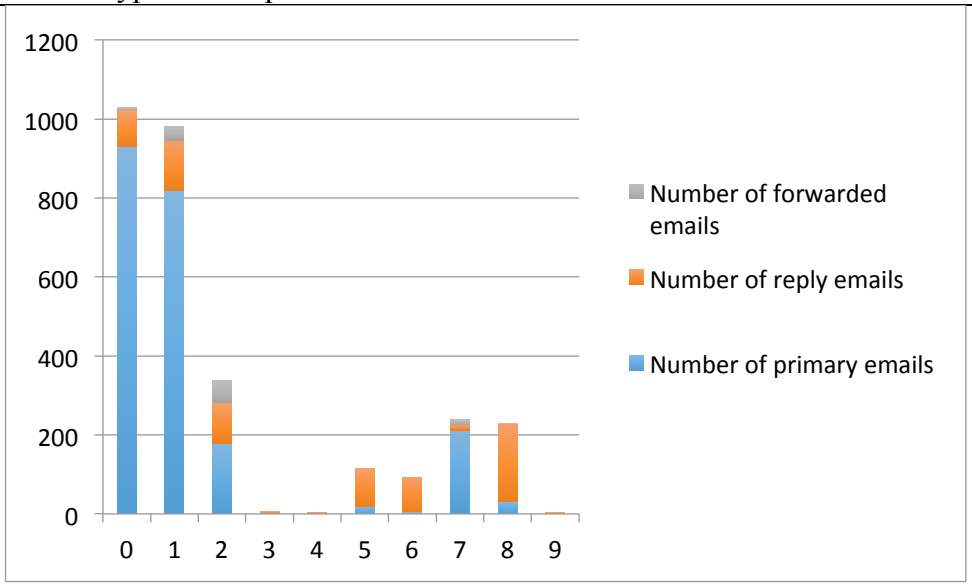
Number of forwarded emails
Number of forwarded emails is the count of patterns based on the forwarded emails. This attribute is closely connected to the attributes of primary and reply emails.

Table 3: Number of emails of each type in the clusters

Cluster ID	Number of primary emails	Proportion of primary emails in cluster	Number of reply emails	Proportion of reply emails in cluster	Number of forwarded emails	Proportion of forwarded emails in cluster
0	930	90%	94	9%	4	1%
1	819	83%	127	13%	35	4%
2	179	53%	102	30%	57	17%
3	3	50%	3	50%	0	0%
4	0	0%	3	100%	0	0%
5	18	16%	97	84%	0	0%
6	5	5%	88	95%	0	0%
7	212	89%	15	6%	11	5%
8	32	14%	196	86%	1	0%
9	0	0%	3	100%	0	0%

There is an interesting difference between the numbers of primary emails in the clusters with above average size and small ones. Small clusters often contain more forwarded or reply emails than big clusters. Big clusters contain mostly primary emails. Big clusters like 0,1,2 or 7 contain from 74% to 88% of primary emails. On the other hand small cluster 9 contains only 21% of primary emails, but 79% of reply emails.
These differences can be a coincidence, but they can also mean, that the group conversations, long email threads etc. tend to fit to the same cluster. The similarity between the email messages from the same conversation is high and similarity between an email from conversation and email of a completely different context is very low. This gap may lead to the situation, when the cluster is filled with the emails form the same thread or conversation and the cluster stays relatively small. And the rest of emails, which are not members of any significant communication thread form their own clusters.

Graph 3: the types of the patterns in the clusters



Average length of subject

The next attribute is simply the average length of subjects of emails used for creating the patterns. This value counts the characters in the subject.

Table 4: Average length of subjects of emails

Cluster ID	**Average number of characters in subject**
0	25.25
1	27.74
2	30.72
3	13.17
4	34.0
5	28.3
6	29.71
7	46.95
8	30.42
9	26.0

Average length of body

This attribute contains the average number of characters in the bodies of emails used for creating the patterns.

Table 5: Average length of bodies of email in particular cluster

Cluster ID	Average number of characters in body
0	1206.73
1	1275.68
2	1039.73
3	24.67
4	63.0
5	142.81
6	286.57
7	1869.11
8	267.28
9	120.0

The average length of subjects of emails is mostly comparable. This length goes from 21 characters to 38. That is not a big difference. On the other hand, the average number of characters in the body of emails differs significantly. It goes from 25 to 3434.

Most frequent senders

This attribute contains 5 most frequent authors of email messages in a particular cluster. These results are deformed, because only a subset of the data was used. This subset belonged to one person – Phillip Allen, so his email addresses appears more often. If the cluster is very small and contains the emails from less than 5 senders, all of them are written in the descending order.

Table 6: Most frequent senders from particular cluster

Cluster ID	5 (or less) most frequent senders
0	phillip.allen@enron.com critical.notice@enron.com ina.rangel@enron.com aod@newsdata.com k..allen@enron.com
1	phillip.allen@enron.com k..allen@enron.com arsystem@mailman.enron.com subscriptions@intelligencepress.com ei_editor@ftenergy.comv
2	k..allen@enron.com phillip.allen@enron.com msimpkins@winstead.com wise.counsel@lpl.com w..cantrell@enron.com
3	k..allen@enron.com phillip.allen@enron.com
4	phillip.allen@enron.com
5	phillip.allen@enron.com
6	phillip.allen@enron.com kathie.grabstald@enron.com
7	no.address@enron.com

	anchordesk_daily@anchordesk.zdlists.com webmaster@earnings.com e-mail.center@wsj.com phillip.allen@enron.com
8	phillip.allen@enron.com edelivery@salomonsmithbarney.com ina.rangel@enron.com enron_update@concureworkplace.com promo@info.iwon.com
9	phillip.allen@enron.com

Most frequent receivers

This attribute is based on a similar logic like the previous one. It contains 5 most frequent receivers of the emails in the particular cluster. If the cluster has less than 5 receivers, all of them are printed in the descending order. Also this attribute is contaminated by the fact that all these emails are from the folder of Phillip Allen.

Table 7: Most frequent receivers from particular cluster

Cluster ID	**5 (or less) most frequent receivers**
0	mike.grigsby@enron.com keith.holst@enron.com frank.ermis@enron.com stagecoachmama@hotmail.com ina.rangel@enron.com
1	mike.grigsby@enron.com keith.holst@enron.com ina.rangel@enron.com k..allen@enron.com pallen70@hotmail.com
2	jsmith@austintx.com k..allen@enron.com mike.grigsby@enron.com keith.holst@enron.com matt.smith@enron.com
3	stouchstone@natsource.com mike.grigsby@enron.com matt.smith@enron.com sneal@enron.com
4	barry.steinhart@enron.com
5	cbpres@austin.rr.com maryrichards7@hotmail.com jsmith@austintx.com mac.d.hargrove@rssmb.com bob.schorr@enron.com
6	jsmith@austintx.com stouchstone@natsource.com cbpres@austin.rr.com bs_stone@yahoo.com louise.kitchen@enron.com
7	pallen@enron.com

	k..allen@enron.com jay.reitmeyer@enron.com mike.grigsby@enron.com matthew.lenhart@enron.com
8	jsmith@austintx.com maryrichards7@hotmail.com cbpres@austin.rr.com pallen@enron.com k..allen@enron.com
9	llewter@palm.net

Most frequent words

The last attribute is closely related to how the patterns are divided into clusters. The patterns are clustered by the words used in the emails and this attribute shows 10 most common words from each cluster. In the previous phases of development of this script, this attribute was the most contaminated one. Before eliminating the stop words, most of these words were prepositions and articles. Also, these words show the need of cleaning of the words: converting all the characters to lowercase or removing rare words-maybe typos. The cleanliness of these words is not perfect yet, but they can already provide us with some interesting information about the patterns in the particular cluster.

Table 8: Most frequent words in each cluster

Cluster ID	**10 (or less) most frequent receivers**
0	phillip subject: forwarded allen/hou/ect ---------- am pm will can gas
1	phillip will forwarded subject: allen/hou/ect am ---------- pm can -
2	phillip 2001 -----original message-----from: allen sent:

	will can know pmto:
3	http://www.caiso.com/systemstatus.html http://www.forbes.com/global/2000/0612/0312072a.html http://www.nwd-wc.usace.army.mil/report/projdata.htm
4	want skills? trading desk?
5	fax will allen phillip number think just time am need
6	will can am want think phillip just let know don't
7	pallen@enron.com k..allen@enron.com jay.reitmeyer@enron.com mike.grigsby@enron.com matthew.lenhart@enron.com 10 will click 2001 - here can [image] new receive email
8	will can need

	am email work know received going let
9	am email tomorrow me. 10 want tonight larry pallen70@hotmail.com.phillip

These results show the possibility of another cleaning of the data. Removing some useless words could be helpful. The data I use are specific, because they are all emails. So it would be useful to adapt the stop words list and add the email specific words with low value for data mining. For example subject, forwarded etc.

4.6 Conclusion

In this experiment, I have successfully clustered the emails according to the most frequently used words in the body of the email. This attribute showed to be a reliable and well working way of clustering these data. I have used and implemented standard and simple way of calculating distances. This algorithm also uses the standard way how to calculate and handle centroids in the k-means algorithm.
While programming this program, also new challenges appeared. The clustering of the whole Enron email set takes a significant amount of time. Opening, reading and closing all of that emails is a long-term operation. Also all the calculations are based on iterating, computing and comparing quite long vectors. All these facts confirm the fact that I am dealing with the typical big data problem. I should think how to speed up and eventually parallelize the algorithm to provide better efficiency and faster computation.

5. Speeding up the computations using GPUs

5.1 Motivation

In this work, I have implemented or will implement several different programs and scripts calculating and solving different kinds of problems. I wanted to use the whole Enron data set as the input data for these experiments.
All these experiments encompassed relatively simple scripts in the beginning, easy to be developed, maintained and run on my own personal computer. But within the course of time, more features became part of these experiments. The scripts became more complicated and running these scripts using the whole Enron data set became more difficult and time consuming. These experiments could run for hours on my personal computer. For the testing purposes, I have also used just a subset of the Enron email set as the input data. This subset can be used while prototyping, developing etc. But when the script is completed, it is interesting to run it using the whole input data.

The problems with performance are not surprising at all. One of the goals of this thesis is to support the work with big data. If I want to examine some real social network, the question of big data cannot be omitted. Social networks usually tend to be associated with big data. I could theoretically use some subset of the social network, but the bigger social network I will use, the more interesting information I might obtain. Quite naturally, processing big data can be difficult using standard hardware and the standard programming approach. Therefore, I needed to find some way to efficiently process big data in my experiments.

The first and most reasonable idea how to speed up my calculations is of course to use a more powerful hardware. A personal computer is hardly a good device for performing big data experiments and calculations. So I have decided to use a more powerful computer located at my faculty, dedicated for the works like mine. Increasing the computational power is of course the way to speed up big data calculations. But I am probably not able to use some more powerful computer each time the size of my input data grows. I should consider the rapid growth of the input data. I should think about a different approach to process big data. One of the possible ideas how to speed up the computing of my experiments is to support my calculations with GPU (Graphics processing unit) computing.

5.2. GPU computing

One of the standard components of IBM PC is the graphics card and its heart is GPU. The development of GPUs was strongly influenced by computer games. Each new generation of games need a more powerful GPUs. Nowadays, use of 20 years old sound card would be theoretically possible, but the use of a 20 years old GPU is not even imaginable. This rapid development of GPUs turned graphic cards into powerful “computers in computers”. They have their own memory, bus and their own powerful processing unit - GPU.
GPU chips are not identical to CPUs (Central processing units). These chips were developed to process nothing but the calculations of graphics. However, it is possible

to use GPUs for different tasks than computing graphics. CPU consists of a few cores optimized for sequential serial processing, while GPU has a massively parallel architecture consisting of thousands of smaller, more efficient cores designed to handle multiple tasks simultaneously. This is an ideal assumption for parallel computations and many more. The program needs to be written in a special way for GPU computation, as it is not possible simply to run standard code on a GPU instead of a CPU. Not every graphics card supports these calculations and also special drivers and libraries are necessary. Programs written for GPUs do not run on GPUs only. They also use the standard CPU. CPU runs the program and only the parts of the program, designed for GPU processing are run by graphics cards.
For a standard computer user, there is hardly any need to use his graphics card for anything else than processing graphics. GPU computing is typically used by scientists, university students etc. who lack the access to expensive supercomputers. I have also decided to use GPU computing to speed up my big data calculations.

5.3. NVIDIA CUDA

In the history of IBM PCs, there were many important manufacturers of GPUs and graphics cards. But nowadays, only 3 relevant manufacturers remained. Intel, AMD and NVIDIA. AMD and NVIDIA offer a way to perform computations on their graphic cards. I have decided to use the GPUs made by NVIDIA and the technology called NVIDIA CUDA (Compute Unified Device Architecture). [10]The main reason is the relatively widespread use of this technology. Another reason is very pragmatic, I had access to faculty computers using NVIDIA GPUs of sufficient power.
CUDA is relatively widespread, nowadays millions of programmers use this technology for a relatively cheap speeding up of their computations. Compatible graphic cards and special CUDA drivers are required for its use.
Programs using CUDA can be written in C, C++ or Fortran. However, wrappers for other programming languages are available. My scripts are written in the Python language, so I also need to use some wrapper to use CUDA. I have decided to use the wrapper library called pycuda.

5.4 Tasks suitable for GPU calculations

Using GPUs for calculations is not the universal solution for every complicated and slow computation. It is a situation comparable with a general parallel processing. It is not possible to parallelize random parts of code that caused significant slowdown. Only some parts of the code, some algorithms, loops etc. are suitable for parallel processing.

A similar situation is with the GPU computing. GPUs were developed to compute graphics, so GPU programmer needs to transform the algorithms to the way similar to the graphics tasks. Graphics tasks typically require relatively simple mathematical operations performed on a relatively large set of input data. Typical is processing of large vectors of data. So if I want to process my data on GPUs, I need to rewrite parts of my code in a similar way. I have more scripts available for this testing, which are suitable for GPU calculations. During my clustering experiments a significant amount of time is consumed by the calculation of the distances between the n-dimensional points described by vectors. This operation is mathematically not

complicated, it is just the Euclidian distance. But this value needs to be calculated for thousands of vectors and in each iteration of the clustering algorithm. This makes clustering a good candidate for use of GPUs.

5.5. Summary

In this chapter, I have described the ideas, which led me to the idea to use GPU computations. My experiments became more complicated and my personal computer was not able to process them anymore. Also, the expected input data for my algorithms are from a category of big data, which also signalizes the need for some more powerful computation means. I have decided to use GPU computing, because it looks like the way to significantly speed up my computations without the need for an expensive supercomputer. The use of NVIDIA CUDA is logical mostly because this technology is widespread and supported by the programming language I use. Also, I have an access to the computers using NVIDIA GPUs supporting CUDA.

6 NVIDIA CUDA based clustering

6.1. Motivation

In the previous chapter, I have described my problems with the speed of some experiments and my ideas how to speed up the computations. I have decided to use NVIDIA CUDA with pycuda wrapper for the Python language.
My application of GPU computing is speeding up the clustering described in Chapter 4. This clustering experiment runs for hours on any computer available for my experiments. A significant reduction of the input data would be possible, but it would also markedly trim the information I can mine from this data. Also it would be impossible for me to even think about using a larger set of input data. A significant amount of time is consumed by repeating the calculations of the Euclidean distance between each pattern and centroid in every iteration of the clustering algorithm. My idea was let GPU calculate these distances.

6.2 Description of the used approach

My idea was to preserve as much as possible from the original non-GPU program described in Chapter 4. The reasons are simple. Only a small part of the program should be rewritten for GPU calculations. Another reason is the comparability of the results. If I keep most of the main parts of the code untouched, I can perform a valid comparison of the results and time necessary to calculate them.

I have been checking the code and trying to find the parts of the code, which are suitable for rewriting for CUDA use. I have focused on the parts of the code, which are the most time consuming. Generally, there are 2 time consuming parts of the code. The first part is logical and cannot be omitted. It is the preparation of clustering patterns. The patterns are based on the emails, so it is necessary to open, read and close each email in the data set. Operations with external files are always costly, especially when it is necessary to process relatively large sets of small text files. The program needs to read all the fields in the particular email and create a new pattern based on the email message.

In my experiment, I am using just the subset of the whole Enron data set. It is the folder and emails of Phillip Allen. It contains 3034 text files. This pre-processing consumes a major part of run time of my script. But, unfortunately, I do not see any way, how to speed up this part of the program using CUDA. It is not possible to use GPU to do input-output operations. Also, there is no way how to use CUDA in processing of the strings of characters from the emails. It is just splitting, cleaning etc. of the strings read from the email messages.

So I needed to focus on other parts of my program. Almost the rest of the time of running my program is consumed by the clustering itself. I have examined the clustering algorithm, if there is any part suitable for GPU processing.

Probably the most frequent type of operations is the calculation of the Euclidean distance between the patterns and centroids of each cluster. Euclidean distance is calculated between the vector of TF-IDF values of the particular pattern and the vectors representing the centroid of each cluster.

Formula 6: Calculation of the Euclidean distance between vectors u and v

(6)

$$distance(\vec{u}, \vec{v}) = \sqrt{\sum_{a=1}^{n} (u_a - v_a)^2}$$

Where u and v are the n-dimensional vectors

The calculation of Euclidian distance between 2 vectors seems to be able to be rewritten for CUDA massive parallel processing. CUDA is very capable to manipulate and calculate with vectors. And all the main data structures of this calculation are vectors. Also the results of partial calculation $(u_a - v_a)^2$ does not depend on each other for different a, so they can be calculated simultaneously.

Modification of the calculation of Euclidean distance

For my calculation, speed is a key attribute. It is reasonable to try every possible way how to save some computation time. The formula for calculating Euclidian distance is set for significant time, but maybe I do not need the exact number representing the distance between the pattern and the centroid of each cluster. For me, the most important information is which cluster is the closest one to the particular pattern.

In my code, the calculation of the Euclidean distance is divided into 3 phases. In the first one is CUDA calculation of $(u_a - v_a)^2$, second is the summing of these results and the last one is the calculation of the square root. But the square root of the summed results does not influence which cluster is the closest one. It means that my results would not change, if I will not use the last part of the calculation, the square root. This change can speed up my clustering.

Formula 7: Modified calculation of the Euclidean distance between vectors u and v

(7)

$$distance(\vec{u}, \vec{v}) = \sum_{1}^{n} (u_a - v_a)^2$$

Where u and v are the n-dimensional vectors

Data types

Pycuda is a specific wrapper making CUDA available to Python programmers. The CUDA code is implemented in the C language and is just embedded to Python code. The Pycuda interface between C and Python is strictly defined. I need to adjust to this format with my program. CUDA cannot work with the standard Python data types. It is necessary to use data types from the library called Numpy. Numpy is a widespread library used in many Python programs. It offers extended abilities to work with the data structures and also offers more advanced collections compared to standard Python.

The vectors containing the values used for clustering are simple Python arrays. So it was not difficult to convert them to Numpy.array. Numpy array is valid data for CUDA calculations based on Pycuda.

CUDA kernel

The most important part of the program using CUDA calculations is the CUDA kernel. CUDA kernel contains the C code to be executed on GPU. It is an independent function with the standard data input. Kernel works in the parallel only way, which is a difference from the standard C function. Special thread management is necessary.

The input of the following kernel are the pointers to the input vectors u and v (*a* and *b* in my code) and the pointer to the array where the results will be stored. It is called *dest* in my code.

Code snippet 1: CUDA kernel

```
mod = SourceModule(""" #include <math.h>
                        __global__ void euclid(float *dest, float *a, float *b)
                        {
                        const int i = threadIdx.x;
                        dest[i] = pow((a[i] - b[i]),2);
                        }
                        """)
```

This CUDA kernel does not calculate the exact Euclidean distance. It calculates only the $(u_a - v_a)^2$ part. The reason is how CUDA works. The whole kernel works in the parallel way only. It is possible to synchronize the threads and force them to write their values into shared variables. But this operation is costly and prevents CUDA from doing what it is best at. So I have decided to perform the rest of the calculation of Euclidian distance out of CUDA kernel. So that is why the output of this kernel is the array containing the $(u_a - v_a)^2$ values. The sum of the *dest* array and square root is calculated after the CUDA kernel completes its calculation.

In Code snippet 1, there is the variable *i* ,maintaining the CUDA threads. It causes the following calculation dest[i] = pow((a[i]-b[i]),2) work as the for-each loop. This is different from the standard C code. The number of iterations performed simultaneously is limited by the number of threads allocated in the block. The block is an important attribute used while executing the CUDA calculation. The block in my program allocates 1024 threads and it is 1-dimensional. Because all 3 arrays (*a,b,desc*) used in the kernel are 1-dimensional.

The rest of the code was preserved in the original way. Only the arrays were converted to the Numpy arrays and the original calculation of Euclidian distance was replaced by the hybrid calculation using GPU.

6.3 Clustering tests with the first version of CUDA based program

I have tested the described program with the same data as the clustering implementation from the preceding chapter. The main reason was the comparability of the results. Another reason was the comparison of speed. The program using GPU calculations should be significantly faster than the original one. I have used the

subset of the Enron email set, the emails of Philip Allen. The results were disappointing. The original program needs about 1.25 hours to cluster the subset of Phillip Allen. The CUDA based program needs about double of that time.

I have expected a significant speed up of the program. But the real speed did not fulfil my expectations. The main reason is probably the overhead connected with the use of CUDA.

The work of CUDA is the calculation of the distance between the vector of each pattern and each centroid.
It is necessary to convert each vector, from pattern and from centroid before using CUDA from standard vector to numpy vector. It may take longer time than I have expected.
Another problem can occur with the calculation of the Euclidean distance. CUDA code calculates only a part of this calculation. The sum of the vector of results is made in the standard code. And this operation probably slows down the script too.
The problem is, there are not many possibilities how to fix the mentioned obstacles without a significant change of my clustering program. The use of numpy the vector is necessary. And my CUDA kernel is parallel only, as I have described before. So it is not possible to make it calculate exact Euclidian distance.

6.4 CUDA clustering with parallel calculation of distances to centroids

After the tests with my CUDA clustering program, I have implemented a different version of the CUDA clustering, trying to fix the problems described in the previous paragraph. The main idea was to reduce the number of conversions and to use CUDA in maximal parallel way, not just to replace the standard code by CUDA based code.

I have examined the code and found a larger part of the code, which can be calculated by CUDA. For CUDA processing, the problem should be vector based and the values in the vector should not depend on each other. In that case, the values can be processed in a fully parallel way and the potential of CUDA is fully used. That was why I have chosen the calculation of the Euclidian distance between the pattern vector and cluster centroid for CUDA processing. The distances from the particular pattern to each cluster are calculated in a sequential way. But the distances from the pattern to cluster centroids are not dependent. They could be calculated at the same time, in a parallel way. I have modified the clustering algorithm based on this idea.

Algorithm 4: Parallel calculating of the distances between patterns and centroids using CUDA

Used variables:
C – Set of all the clusters
d, c - one cluster from *C*
u – centroid vector from *d*
p – one pattern from *c*
v – vector of TF-IDF values of pattern *p*
mv – vector containing |*C*| - times the TF-IDF vector from *p*
cv – vector containing concatenated centroid vectors of all the clusters in *C*
dv- vector containing the result of CUDA calculation
distances – vector of couples : cluster identification – distance from *p* to cluster
a – subvector of *dv*, containing the values for particular cluster

Input of the algorithm:
The input of the algorithm is the set of clusters *C* containing all the patterns *p*

Output of the algorithm:
The output of the algorithm is the set of clusters *C* containing all the patterns *p*, but the patterns have been reorganized according to the clustering algorithm.

```
-create an empty vector cv
-for each cluster d in the set of clusters C
        -concatenate vector cv and centroid vector u from cluster d
- for each cluster c in the set of clusters C
        -for each pattern p in the cluster c
                -create empty vector mv
                -create empty vector distances
                -concatenate |C| - times vector v from pattern p into mv
                -execute CUDA kernel with mv and cv as the input parameters, dv as
                the output parameter
                -divide dv into |C| subvectors
                -for each subvector a
                        -calculate the sum of a
                        -insert the result of calculation of subvector a with the
                        identifier of particular cluster in distances
        -find minimal value in distances and choose a new cluster
```

It means that the parallel parts of the calculations of Euclidian distances between the particular pattern and centroids of all the vectors can be calculated in parallel way, at the same time. This can significantly reduce the time necessary for calculation of distances between patterns and centroids. It also reduces the number of necessary data conversions.

6.5 Modified CUDA kernel

Code snippet 2: modified CUDA kernel

```
mod = SourceModule("""
                        __global__ void euclid(float *dest, float *a, float *b)
                        {
                        int i= threadIdx.x + blockDim.x * blockIdx.x;
                        if (a[i]==b[i]){
                        dest[i]=0.0;
                        }else{
                        dest[i] = (a[i] - b[i])*(a[i] - b[i]); }


                        }
                        """)
```

The CUDA kernels from snippet 1 and snippet 2 are almost similar, with 2 differences. First is the condition which checks if values entering the calculation are equal. If yes, no calculation is necessary. The second change is the change in the calculation of the thread variable *i*, controlling the threading in the CUDA kernel. The size of the blocks and grids changed from the first implementation.

Calculation of blocks and grids of threads in CUDA

In CUDA, a strict organization of threads is required. The power of CUDA is in the parallel, thread based calculations. The threads are organized into blocks and blocks into grids. It is necessary to provide the correct sizes of the blocks and grids as the input parameter while launching the CUDA kernel. The correct values are very important, but it is not easy to determine them. Also, the correct settings of blocks and grids are different for each kernel and each type of input data. The maximal limit for all the threads in all the blocks and all the grids is based on the actual hardware. So the program needs to be optimized for the particular computer. If the blocks and grids ask for more threads than is the hardware limit, the calculation will not start. Another criterion which should be taken into account is the nature of the processed data. The length of vectors should not exceed the maximal number of threads and also the number of dimensions of input vectors is important. The blocks of threads should have the same number of dimensions as the computed vectors. The blocks can be 1, 2, or 3 dimensional. The ideas in this paragraph are based on the document of Chris Paciorek from University of Berkley. [11]
The limitations and capabilities significantly differ with the version of CUDA framework.

Table 9: CUDA framework version compute capabilities[12]

	Compute capability						
Specifications	1.0	1.1	1.2	1.3	2.x	3.0	3.5
64-bit in global memory	No		Yes				
Max dim of grid	2				3		
Max gridDim.x	65535					$2^{31}-1$	
Max gridDim.y/z	65535						
Max dim of block	3						
Max blockDim.x/y	512				1024		
Max gridDim.z	64						
Max threads/block	512				1024		
Warp size	32						
Max blocks/MP	8					16	
Max threads/MP	768		1024		1536	2048	

Another thread and memory limitations can be summarised into these 4 points:[13]

1. No block can have more than 512/1024 threads in total (Compute Capability 1.x or 2.x-3.x respectively)
2. The maximum dimension of each block is limited to [512,512,64]/[1024,1024,64] (Compute capability 1.x/2.x)
3. No block can consume more than 8k/16k/32k registers in total (Compute capability 1.0,1.1/1.2,1.3/2.x)
4. No block can consume more than 16kb/48kb of shared memory (Compute capability 1.x/2.x)

These are just general rules for creating the blocks and grids. It is difficult to calculate or somehow guess the correct values on the first try. The correct values should be found often in the experimental way. It was the same with the values for my experiments.

CUDA blocks and grids used in my program

The size of the block and grid in my program was set mostly in an experimental way, trying to fulfil the conditions mentioned in this chapter. I am using the following values:

Table 11: Size of dimensions of CUDA block and grid

Type of object	1st dimension	2nd dimension	3rd dimension
Block	20	1	1
Grid	Length of the word vector	1	1

The second and third dimension must be always 1 in my case, because I am using only 1 dimensional vectors for my calculation.

Problem of number of conversions

In this and previous chapter, I am describing many various problems with performance, speed and proposals for their solution. One of them is the problem of conversions. As I have mentioned while describing the way how CUDA kernel works, it is necessary to use Pycuda collections on the input. However, my program was using standard Python collections. I have been converting these arrays before calling CUDA kernel. But this conversion can be relative time consuming and can cause the significant slow down when performed multiple times, which is a case of my program.

I have realized, that the same array is converted many times, which is a waste of computational time. During one iteration of clustering algorithm, the centroids remain the same. And also the vectors inside patterns do not change at all during the whole experiment.

I have modified the program to perform the conversion of pattern vector only once, after the vector and pattern are created. Also, the conversion of the vectors describing centroid is necessary only once per iteration.

This change can save computational time especially during big data experiments, which is my case. It caused significant speed up of the algorithm, which is obvious from the speed comparison.

6.6 Speed comparison

I have tested 4 different versions of the clustering program. All the versions clustered the same data, the emails of Phillip Allen. Also, the same hardware was used for these experiments.

Version 1 is the original clustering program. It uses the standard k-means algorithm without CUDA or any other speed optimization.

Version 2 is the first CUDA version. In this version, the standard computation of Euclidean distance is replaced by CUDA version.

Version 3 is the upgraded CUDA based version using the parallel computation of distance from pattern to centroid vectors. It uses recalculated IDF values, etc. This version is described in part 6.4.

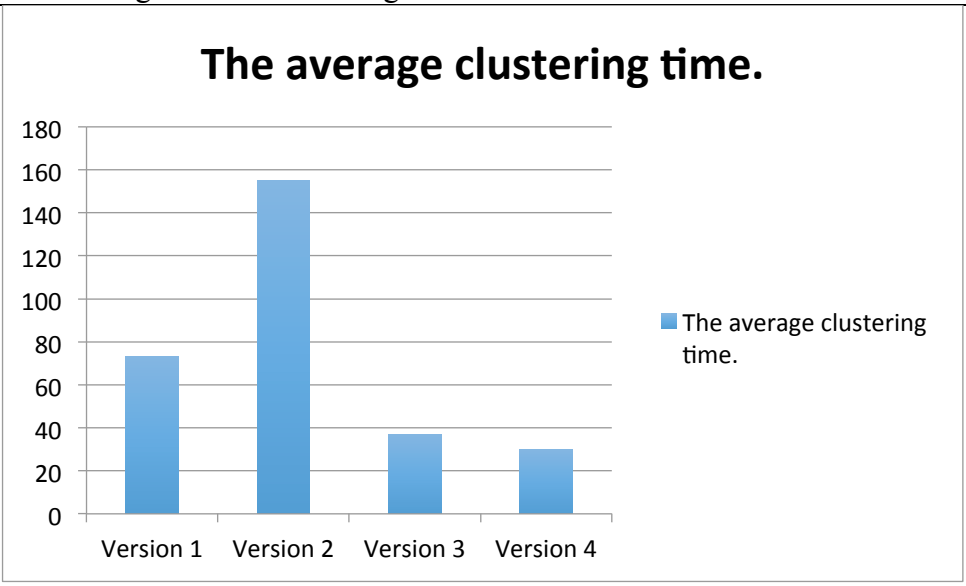
Version 4 is the slightly upgraded version of Version 3. It uses the modified Euclidian distance calculation described in Formula 13 and modified system of conversions described in the previous paragraph.

Table 10: Speed comparison

	Version 1	Version 2	Version 3	Version 4
The average clustering time.	1 hour, 13 minutes	2 hours, 35 minutes	37 minutes	30 minutes

The tests were performed with the same conditions. They were run on the same hardware and the computer was dedicated just for them. I have repeated the tests 10 times and used the average value. I have used the time of the run of the complete clustering program, not just clustering part.

Graph 4: Average time of clustering in minutes



Brief description of used hardware

For my experiment I have used faculty machine dedicated just for my calculations, with the following specifications:

Table 11: Hardware specifications of used machine

Component	Specification
CPU model	Intel(R) Core(TM) i5-4570S CPU 2.90GHz
CPUs	4
Cores per CPU	4
RAM	16 GB
GPU	GeForce GTX 980
VRAM	4 GB

6.7 Conclusion

In this chapter, I have successfully implemented the CUDA kernel into my clustering experiment. The main reason why I have used CUDA was speeding up of the

clustering calculations. This did not work as I have expected. Using CUDA generates some overhead, which causes the slow down of the whole program. The simple replacement of the traditional code by an almost similar CUDA code is probably not enough to speed up my computations. I needed to think about the way to maximally use the way that CUDA works to speed up my program. I have also implemented the second version of the algorithm, with extended and more correct values for allocation of threads via blocks and grid in CUDA interface. Solving of these problems helped me to understand better how threading in CUDA works. The latest version of this clustering program has shown, that CUDA can significantly speed up my calculations. But it was important to understand how to use it in the correct way and solve performance issues. I can say in simplified way, that CUDA works better in the program when processing relatively small amount of relatively long vectors than processing more short vectors. The overhead while converting the data and calling the CUDA API is not unimportant. Also the understanding of the way how clustering works can help me to speed up the calculation. For example with removing the square root from the calculation of the Euclidian distance. From the point of the view of distance calculation, it makes no sense, but from the point of the view of my clustering it is not necessary and it can make the calculation faster.

7. Person - based clustering of the Enron data set

7.1 Motivation

There are many different ways of clustering the Enron data set. In the previous chapter, I have used individual email messages as the patterns for clustering. Another possible perspective can be to consider the persons – senders and receivers of these emails as the patterns. This view based on the persons can offer different results because of the different attributes. Using the words from the bodies of the emails sent or received by particular user would be still possible, but I have focused on yet different attributes in this chapter.

Using persons as input patterns makes it possible to include communication itself as an attribute for clustering, which was used in this experiment.

This person-based clustering experiment was originally the first clustering experiment. But it was significantly modified after the development of email-based clustering, using the experience and source code from it.

7.2 The idea of the first person-based clustering experiment

In this clustering experiment, I have also decided to use the K-means algorithm. As I have written in the chapter about K-means, it is a simple and efficient algorithm. It is also easy to program, debug and modify.

While preparing the clustering experiment, I needed to make some decisions crucial for the character and result of the clustering. The first decision was the choice of the patterns – entities to be clustered. There were more possibilities what to choose. For this clustering experiment, I have chosen all the persons from the Enron email set to be the nodes. Each person is defined by a unique email address. It can happen, that an email address belongs to a robot, email server etc. But there are not many addresses like this and hence they should not contaminate the results. And it is of course always possible to eliminate them.

K-means is a numeric based algorithm. It is intended for data defined by vectors from an n-dimensional space. In that case, the patterns can be simply compared or the distance between them is easy to calculate. I the beginning, I have struggled to achieve this in my clustering experiment. I wanted to be able to use the attributes like:

X has sent about the same amount of emails like Y on the same day

An attribute like this sounds reasonable while clustering persons based on their email communication. But it is not simple to specify attribute for the K-means algorithm to work well and be able to cluster these data. So I needed to find the way, how to encode such facts into K-means clusterable form.

7.3 Description of the used approach in the first person-based clustering experiment.

Creation of the patterns

The first task for my program was to create the patterns. By creating the patterns, I do not mean just the reading of the names of the owners of the main folders. Creating

patterns includes walking the whole tree structure and reading each message. Based on these messages, the respective attributes of each user are calculated. Most of the time of running when running my program has been spent to the create these patterns.
The input of the following algorithm is the whole Enron email data set in its original form. The output is the set of the patterns/nodes N.

Algorithm 5: Creation of the nodes in employee-based clustering experiment
N: set of all the patterns n : one pattern from N
- enter the root folder -enter each subfolder -open each message -read the field : Date -read the field : From -read the field : To -calculate the attributes based on the enumerated fields -create the pattern- object n -insert the new pattern into the set of all the patterns N

Former ideas about attributes of the patterns
In my first implementation of these ideas, I have decided not to use the simple numerical attributes for clustering the nodes. I wanted to describe relatively complicated relationships (for K-means use) and include the different attributes into the same clustering experiment. I wanted to use the links to other persons, dates of frequent communication or numbers of sent and received emails. That made the clustering task difficult and also the finding of the centroid more complicated. I have modified K-means algorithm, hoping to cluster these different data together.

The first implementation of person based clustering.
The first implementation of person based clustering was based on the naïve idea of putting all the possible parameters together and somehow cluster them. The attributes reminded of the vector-space model but it was not the vector space model. Also K-means algorithm was modified in order to fulfil the special needs of my parameters. All these modifications turned to be contra productive in the end.
This implementation was, namely, limited to folder-owners only, and cannot capture well all the persons in the data set.

The attributes of the first person based clustering implementation

The first implementation contained various more or less meaningful attributes used for clustering. Their description follows:

Sent emails attribute: This attribute was based on the amount of all sent emails by the particular person. An email was considered sent, when the email address in the field “From” matched the name of the owner of the particular folder. The algorithm simply counted number of these messages.

Received emails attribute: The attribute based on the number of received emails was calculated in a similar way like the previous one, the sent emails. The program read the emails in the particular user folder and every time it found an email not written by that respective user, it considered the email to be received by that user.
Mutual receivers: The idea for this attribute was based on the fact, that people, who write to the same persons, may know each other. This makes them closer in the real world and it should also make them closer in my clustering experiment. The algorithm made a map (dictionary) of the email address and the number of occurrences. Only the sent emails are used. In the end, the 5 most frequent email addresses were chosen and the list of them was created.

Mutual writers: The idea of this attribute was similar to the idea of the attribute of mutual receivers. People that often receive the emails from the same people could know each other or be at least close in the company hierarchy, in the same team etc. This attribute was also calculated in a similar way like the attribute of mutual receivers. The list of 5 most frequent addresses in received emails was created.

The "friends" attribute: In this attribute, I have tried to figure out, if the people represented as the nodes knew each other directly. The calculation of this attribute was using the list of 5 most frequent addresses from the list used in the calculation of the attribute of mutual receivers. If node A was in the top 5 list of B and vice versa, they were considered as "friends".

The "busy days" attribute: This attribute was based on the idea, that people working together solve the same problems. Solving the same problems also means solving them at the same time if possible. Solving problems includes communication. I have tried to find the peaks in the frequency of the email communication of each person and compare them.

Problems of the first implementation of the person based clustering experiment
The ideas that led me to collect the above mentioned attributes were possibly correct. All the mentioned parameters can contain useful information about the particular pattern provided for each respective person. But my implementation of this type of clustering was not ideal.
The first problem was incompatibility of the attributes. Comparing the busiest day with the person who sends the most emails is probably not the best idea. This incompatibility led to breaking the standard vector-space model and using something hybrid.
This incompatibility led to another problem, namely, the problem of the dimension of the attributes. Each attribute used its own dimension and it was necessary to add some weights to them, to make the logically more important attribute also more important in the calculation. But this correction led to another problem, problem how to set the weights. I did not have any precise way, how to set the correct weights. So in the end, I was just guessing them. Guessing of the weights can simply lead to incorrect results.
The last problem dealt with the corrupted vector space model. I was not able to use the standard K means algorithm to cluster my attributes. It was necessary to calculate the distance between all the pairs of attributes. This means, that the algorithm calculated the distance first based on the common senders, then separately for common receivers etc.

In the end the weights were added and some kind of overall distance was calculated.

The results of the first implementation of the person based clustering
Despite of the described problems, the first implementation of the clustering program was finished and tested. It was possible to cluster the Enron email set. But it was unclear how to interpret the results. Because of the problems described in this chapter, I was not sure what the results say and how to interpret them. I did not know how to verify them.

This person based clustering was originally my first clustering experiment. I have learned a lot during the development of this algorithm but also during the development of the email based clustering experiment. The main idea about the important attributes was probably not completely wrong, its accomplishment was not ideal, especially the idea to mix all the attributes together and cluster them in some pseudo vector space format. So I have decided to make a completely new implementation of person based clustering. It should be based on the same ideas but in must use standard vector space model and standard K means algorithm.

7.4 Rethinking the ideas of person based clustering

I wanted to use more-less same the ideas for the new clustering experiment, but not the same attributes and not in the same way. I have divided the former attributes into three main groups:
Communication based attributes: In this group, there are attributes based on the communication between the patterns representing people.
Time based attributes: This group would contain only the former attribute "busy days".
The rest: This group can contain less important attributes: numbers of sent and received emails.

The first implementation showed, that having unrelated attributes in one clustering experiment is not a good idea. So I have decided to split this experiment into 2 separate clustering experiments. The first one is based on mutual communication of persons from the Enron email set and the second is based on the dates of frequent communication. These 2 experiments should cover the ideas behind the attributes in the first 2 groups. I have decided not to use the attributes from the third group, because of their low importance and incompatibility with the rest of the attributes.

7.5 Person- communication based clustering

The idea of this experiment is based on the communication of the persons from the Enron data set as described in this chapter. But the real implementation is very different from the original program. Instead of some hybrid model, pure vector-space model was chosen. Also, standard K-means is used for clustering.
This implementation of clustering is based on the implementation of email based clustering described in the previous chapter. All the main parts of the code are similar with email based clustering implementation. It uses the same algorithm to read the emails and the clustering loop is also practically identic. Person-communication clustering also uses CUDA for speeding up the calculations.

The patterns are created in a different way than in the original experiment. Not only the persons which own the email folder are chosen as the patterns, but each email address in the used set of emails represents an unique person.
The vector used for clustering of the each pattern is calculated in the following way:

Algorithm 6: creating the patters for clustering

Variables:
E – Enron email set
e – one email from E
F – set of all unique persons found
f – one person from set F
P – set of all the patterns
M – set of emails belonging to the particular pattern
m – one email from M
c – set of couples : email address-number of occurrences.
v – clustering vector.
p – one pattern from P, p=(c,v)

Input of the algorithm: set of all the emails E
Output of the algorithm: set of patterns P

```
-initialize the variables and collections
-for each email e from E
        -if the sender of email e is not in the set F
                -add sender of email e to set F
-for each person in the set F
        -create pattern p, based on person f
        -add all the emails with sender f into p
        -place p into P

-for each pattern p in P
        -for email m in M
                -read sender of email M
                        -Increment the number of occurrences of particular address in
c from p

-for each person p in P
        -iterate c from p
                -iterate all the found persons f in F
                        -if the person from c equals f
                                -write the number of occurrences  of particular person
                                from c to v
```

Person-communication based clustering results
While testing this algorithm, I have faced the similar problem like in the email based clustering, the size of the Enron email data set. While using only the owners of the folders I had less then 150 patterns for clustering. But in this implementation, each email sender represents its own pattern. It means, that the number of patterns rose to thousands. For my experiment, I have also chosen the subset of the whole Enron email set, the emails of Phillip Allen.

The results of person-communication based clustering

I have clustered the emails of Phillip Allen using the person-communication based program described in this chapter. In most of my clustering algorithms, I have pre-set the number of 10 clusters. In this test, only 5 clusters were used. The number of patterns based on persons is namely much lower than number of patterns based on emails, so I expected that 5 clusters would be enough and it would be easier to debug and verify the results.

Table 12: the results of person-communication based clustering

Cluster	Patterns in cluster
Cluster 0	phillip.allen@enron.com
Cluster 1	kim.ward@enron.com, alyse.herasimchuk@enron.com frank.hayden@enron.com, tim.heizenrader@enron.com rebecca.cantrell@enron.com,tiffany.miller@enron.com tim.belden@enron.com,paul.kaufman@enron.com christi.nicolay@enron.com,philip.polsky@enron.com kirk.mcdaniel@enron.com
Cluster 2	k..allen@enron.com
Cluster 3	sarah.novosel@enron.com,lisa.jacobson@enron.com
Cluster 4	ryan.o'rourke@enron.com, james.bruce@enron.com, kam.keiser@enron.com ashley.worthing@enron.com, veronica.espinoza@enron.com , ina.rangel@enron.com carole.frank@enron.com, louise.kitchen@enron.com, msimpkins@winstead.com kathie.grabstald@enron.com, w..cantrell@enron.com, karen.buckley@enron.com technology.enron@enron.com,hunter.williams@grandecom.com,gthorse@aboutcis.com leanne@integrityrs.com, mike.grigsby@enron.com, ray.alvarez@enron.com ksumey@ftenergy.com, members@realmoney.com, capcon@gmu.edu joannie.williamson@enron.com , dleduca714@yahoo.com, katrina_sumey@platts.com jsmith@austintx.com, arsystem@mailman.enron.com, public.relations@enron.com bounce-news-932653@lists.autoweb.com, subscriptions@intelligencepress.com cbpres@austin.rr.com, critical.notice@enron.com, richard.shapiro@enron.com stephanie.miller@enron.com,messenger@ecm.bloomberg.com,yild@zdemail.zdlists.com market-reply@listserv.dowjones.com,aod@newsdata.com,announce@inbox.nytimes.com webmaster@earnings.com, 1.11913372.-2@multexinvestornetwork.com ei_editor@ftenergy.com, yahoo-delivers@yahoo-inc.com, tracy.arthur@enron.com perfmgmt@enron.com, jfreeman@ssm.net, rob_tom@freenet.carleton.ca grensheltr@aol.com, cpa@opthome.com, outlook-migration-team@enron.com matt@fastpacket.net,owner-strawbale@crest.org,calxa@aol.com, billc@greenbuilder.com,gthorse@keyad.com,bobregon@bga.com, geninfo@state-bank.com,steven.matthews@ubspw.com,brad.jones@enron.com, david.port@enron.com,mery.l.brown@accenture.com, kathryn.sheppard@enron.com prizemachine@feedback.iwon.com, renee.ratcliff@enron.com anchordesk_daily@anchordesk.zdlists.com,pallen70@hotmail.com,c..gossett@enron.com gift@amazon.com, monica.l.brown@accenture.com, david.oxley@enron.com heather.dunton@enron.com ,richard.morgan@austinenergy.com melissaspears@open2win.oi3.net, wise.counsel@lpl.com, mondohed@gte.net unsubscribe-i@networkpromotion.com, exclusive_offers@sportsline.com software@mail02.unitedmarketingstrategies.com, greg.whalley@enron.com enron_update@concureworkplace.com, no.address@enron.com, iwon@info.iwon.com jwills3@swbell.net, michelle.akers@enron.com, delivers@amazon.com pam.butler@enron.com, mark.whitt@enron.com, usatoday1430@hotmail.kg adrianne.engler@enron.com ,cbssportsline.com_planters@mail.0mm.com lindsay.renaud@enron.com, itsimazing@response.etracks.com, jeshett@yahoo.com al.pollard@newpower.com, laura.a.de.la.torre@accenture.com, gifts@info.iwon.com

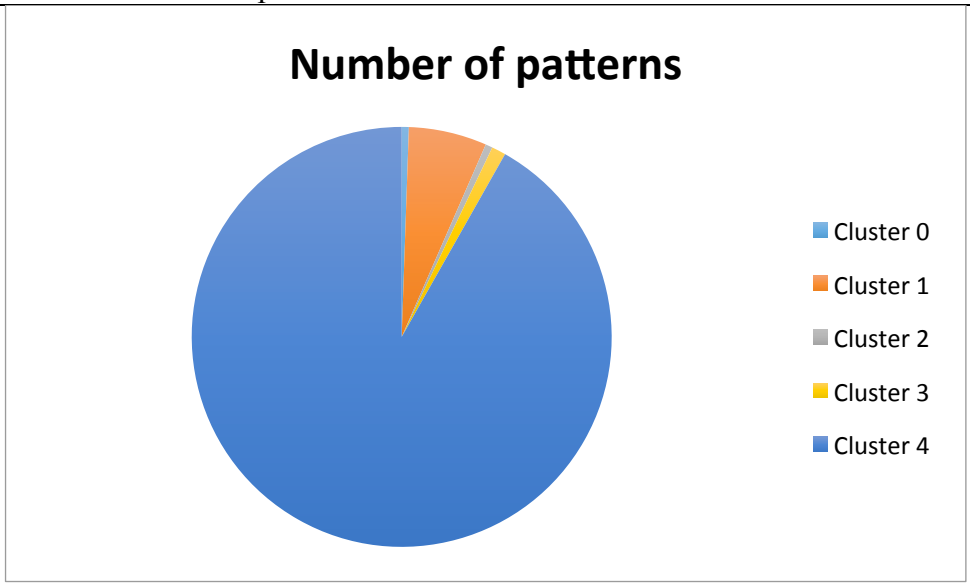
	e-mail.center@wsj.com, richard.hash@openspirit.com ,info@open2win.roi1.net ei_editor@platts.com, announcements.enron@enron.com winnerannouncements@info.iwon.com, edelivery@salomonsmithbarney.com promo@info.iwon.com,noreply@ccomad3.uu.commissioner.com, randy.bhatia@enron.com showtimes@amazon.com,news@prosrm.com, postmaster@glmail2.networkpromotion.com leave-htmlnews-2508405s@lists.autoweb.com, patti.sullivan@enron.com discount@open2win.oi3.net, bmg_support@adm.chtah.com d1131c2e-3c2c-40c2-bb3b-685d6a0d2700@autotoolbox.net, savita.puthigai@enron.com eservices@tdwaterhouse.com, open2win.0ll1.net@mailman.enron.com online.service@schwab.com, gousa6179@hotmail.kg networkcommerce-tdtl20011226@ombramarketing.com, customerservice@tdwaterhouse.com dmallory@ftenergy.com, cpa@oihost.net, ken.shulklapper@enron.com yevgeny.frolov@enron.com, john.lavorato@enron.com, davidsmith@open2win.oi3.net lisa@techxans.org, deanrodgers@kaseco.com, c..aucoin@enron.com networkcommerce-tdcd20011221@ombramarketing.com, tim.o'rourke@enron.com sprint@info.iwon.com, infousa4492@telkom.net, bodyshop@enron.com clickathome@enron.com, keith.holst@enron.com, resources.human@enron.com morpheus@inyouremail.com, networkcommerce-tdsd20011228@ombramarketing.com important_phone_call@response.etracks.com, stephanie.sever@enron.com book-news@amazon.com, rick.bellows@enron.com, chad.landry@enron.com jeff.richter@enron.com, sheri.a.righi@accenture.com, m..tholt@enron.com customer_service@pmail.feer.com, newsletter@pussylickingcunts.com apkpcp@prodigy.net, mac.d.hargrove@rssmb.com, annualconference@prosrm.com vivatrim@open2win.roi1.net, chairman.office@enron.com, oportunity@cells4free.com cpa@optmails.com, txreport@skippingstone.com, biliana.pehlivanova@enron.com editor@theb2bvoice.com, exchange.administrator@enron.com, editor@hersweeps.com customerservice@chaseplatinum.po-1.com, steven.matthews@ubspainewebber.com store-news@amazon.com, software@mail01.unitedmarketingstrategies.com listservices@open2win.1ll0.net, exclusive_ofers@sportsline.com

The results of this clustering are interesting. We can see that cluster 0 contains only one pattern – Phillip Allen. Phillip Allen is so unique pattern that placing it alone in separate cluster should be the correct solution. As the owner of the original folder, his communication is clearly unique in this subset.

Another interesting cluster is cluster number 2, also containing only one pattern: k..allen@enron.com. This email address is also probably connected to Phillip Allen, so I can again expect some unique status of this pattern in the data set.

Most of the patterns were placed into cluster 4. But another interesting fact is, that all the small clusters -0,1,2,3 contain only email addresses from Enron domain. All the external addresses were placed into the biggest cluster number 4 together with some other Enron addresses.

Graph 5: Distribution of patterns in the clusters



7.6 Time based person clustering

The second part of the parameters of the original clustering was based on the time of the sent and received emails. The main idea of this new implementation based on the time, was to count the number of the emails sent by a particular user on the particular day and create the input vector based on these data. It means that the clustering program would compare the number of sent emails on each day between two patterns.

This implementation is again very similar to the email based clustering and especially to person-communication clustering described in this chapter. It uses the same algorithm to read the emails, cluster the patterns using CUDA etc.

The algorithm creating the patterns is similar to the Algorithm 6 described in this chapter. The only difference is, that the algorithm reads the date and time of the email instead of the sender or receiver email address. This difference is so minor, that the new description of the algorithm is not necessary.

Time based person clustering results

To test this algorithm, the very same subset of the data like on the rest of the clustering experiments was chosen. It means the use of the folder of Phillip Allen. Choosing the same subset makes the results and clustering time better comparable. I have also chosen the same number of clusters like in the previous experiment: 5. The expected number of patterns is the same and the same number of clusters make the results better comparable.

Table 13: the results of time based person clustering

Cluster	Patterns
Cluster 0	phillip.allen@enron.com
Cluster 1	ina.rangel@enron.com, subscriptions@intelligencepress.com rob_tom@freenet.carleton.ca, matt@fastpacket.net owner-strawbale@crest.org, calxa@aol.com

	billc@greenbuilder.com, gthorse@keyad.com bobregon@bga.com, hunter.williams@grandecom.com anchordesk_daily@anchordesk.zdlists.com,msimpkins@winstead.com arsystem@mailman.enron.com
Cluster 2	k..allen@enron.com
Cluster 3	e-mail.center@wsj.com, winnerannouncements@info.iwon.com postmaster@glmail2.networkpromotion.com, eservices@tdwaterhouse.com technology.enron@enron.com networkcommerce-tdcd20011221@ombramarketing.com
Cluster 4	geninfo@state-bank.com, mery.l.brown@accenture.com, greg.whalley@enron.com lindsay.renaud@enron.com, leave-htmlnews-2508405s@lists.autoweb.com davidsmith@open2win.oi3.net, kirk.mcdaniel@enron.com, no.address@enron.com mike.grigsby@enron.com, d1131c2e-3c2c-40c2-bb3b-685d6a0d2700@autotoolbox.net kam.keiser@enron.com, webmaster@earnings.com, monica.l.brown@accenture.com wise.counsel@lpl.com, enron_update@concureworkplace.com, james.bruce@enron.com iwon@info.iwon.com,adrianne.engler@enron.com, edelivery@salomonsmithbarney.com ei_editor@ftenergy.com, philip.polsky@enron.com, kathryn.sheppard@enron.com renee.ratcliff@enron.com, pallen70@hotmail.com, mondohed@gte.net unsubscribe-i@networkpromotion.com, leanne@integrityrs.com, delivers@amazon.com kathie.grabstald@enron.com,itsimazing@response.etracks.com, veronica.espinoza@enron.com cbssportsline.com_planters@mail.0mm.com,promo@info.iwon.com, sarah.novosel@enron.com critical.notice@enron.com, aod@newsdata.com, rebecca.cantrell@enron.com brad.jones@enron.com, david.port@enron.com, prizemachine@feedback.iwon.com gthorse@about-cis.com, louise.kitchen@enron.com, c..gossett@enron.com heather.dunton@enron.com, richard.morgan@austinenergy.com exclusive_offers@sportsline.com, software@mail02.unitedmarketingstrategies.com mark.whitt@enron.com,jwills3@swbell.net,carole.frank@enron.com, patti.sullivan@enron.com members@realmoney.com, bmg_support@adm.chtah.com, lisa@techxans.org dleduca714@yahoo.com, katrina_sumey@platts.com, jsmith@austintx.com public.relations@enron.com,bounce-news-932653@lists.autoweb.com, cbpres@austin.rr.com richard.shapiro@enron.com,stephanie.miller@enron.com, messenger@ecm.bloomberg.com yild@zdemail.zdlists.com,market-reply@listserv.dowjones.com, announce@inbox.nytimes.com kim.ward@enron.com, alyse.herasimchuk@enron.com, frank.hayden@enron.com 1.11913372.-2@multexinvestornetwork.com, yahoo-delivers@yahoo-inc.com lisa.jacobson@enron.com, tracy.arthur@enron.com, tim.heizenrader@enron.com perfmgmt@enron.com, tiffany.miller@enron.com, tim.belden@enron.com paul.kaufman@enron.com,christi.nicolay@enron.com,jfreeman@ssm.net, grensheltr@aol.com cpa@opthome.com, outlook-migration-team@enron.com, steven.matthews@ubspw.com gift@amazon.com, david.oxley@enron.com, melissaspears@open2win.oi3.net ryan.o'rourke@enron.com, michelle.akers@enron.com, pam.butler@enron.com w..cantrell@enron.com, usatoday1430@hotmail.kg, jeshett@yahoo.com al.pollard@newpower.com, laura.a.de.la.torre@accenture.com, gifts@info.iwon.com richard.hash@openspirit.com, info@open2win.roi1.net, ei_editor@platts.com announcements.enron@enron.com, noreply@ccomad3.uu.commissioner.com randy.bhatia@enron.com, showtimes@amazon.com, ray.alvarez@enron.com ksumey@ftenergy.com,news@prosrm.com,discount@open2win.oi3.net, capcon@gmu.edu savita.puthigai@enron.com, open2win.0ll1.net@mailman.enron.com online.service@schwab.com, gousa6179@hotmail.kg, joannie.williamson@enron.com networkcommerce-tdtl20011226@ombramarketing.com, customerservice@tdwaterhouse.com karen.buckley@enron.com, dmallory@ftenergy.com, cpa@oihost.net

	ken.shulklapper@enron.com, yevgeny.frolov@enron.com, john.lavorato@enron.com deanrodgers@kaseco.com, ashley.worthing@enron.com, c..aucoin@enron.com tim.o'rourke@enron.com, sprint@info.iwon.com, infousa4492@telkom.net bodyshop@enron.com, clickathome@enron.com, keith.holst@enron.com resources.human@enron.com, morpheus@inyouremail.com networkcommerce-tdsd20011228@ombramarketing.com important_phone_call@response.etracks.com, stephanie.sever@enron.com book-news@amazon.com, rick.bellows@enron.com, chad.landry@enron.com jeff.richter@enron.com, sheri.a.righi@accenture.com, m..tholt@enron.com customer_service@pmail.feer.com,newsletter@pussylickingcunts.com, apkpcp@prodigy.net mac.d.hargrove@rssmb.com,annualconference@prosrm.com, vivatrim@open2win.roi1.net chairman.office@enron.com, oportunity@cells4free.com, cpa@optmails.com txreport@skippingstone.com, biliana.pehlivanova@enron.com, editor@theb2bvoice.com exchange.administrator@enron.com, editor@hersweeps.com customerservice@chaseplatinum.po-1.com, steven.matthews@ubspainewebber.com store-news@amazon.com, software@mail01.unitedmarketingstrategies.com listservices@open2win.1ll0.ne, exclusive_ofers@sportsline.com

The same data were clustered using different attributes, but the results are not completely different. We can see that the special addresses, phillip.allen@enron.com and k..allen@enron.com were again placed to separate clusters, which is interesting. Their communication from both points of view is unique enough, to place them separately.
The results also contain one cluster significantly bigger than the previous ones, which contains most of the patterns. It shows that a small amount of patterns use very different communication schemas from the rest of the patterns.
However the small clusters 1 and 3 are not equal to the results of the previous experiment. The domains of email senders are mixed in the small clusters, I cannot say it is Enron- only, unlike in the first experiment.
But the structure of the clusters is interestingly comparable between 2 experiments described in this chapter.

7.7 Conclusion

Person based clustering was my first and also last clustering experiment in this work. I have implemented the first version, which served as clustering “hello world” for me and made me to think about the choice of patterns, attributes etc. I have identified interesting attributes in the first clustering experiment, but I wanted to create a unique way of clustering them. This approach does not mean automatically a failure, but in this case the modification was not necessary and created significant overhead for finishing the program in a meaningful way. Another problem was incompatibility of the chosen attributes, which made it difficult to use the standard vector space model.
These problems led me to the second version of the person based clustering. In the second version, I have used the experience I have gained during the development of both email based clustering experiments. I have also come to a logical conclusion, that only related attributes should be clustered together. This conclusion split the experiment into 2 experiments, one based of the communication of the people and the second one on the time of the communication.
The results of these experiments were interesting, because these 2 experiments based on different attributes have shown partially similar results. For example, both

approaches confirmed the uniqueness of the owner of the folder and placed Phillip Allen into a separate folder and alone.
These results show, that however the original implementation was not correct, the ideas behind it could lead into meaningful results.
Another interesting point is the reusability of the CUDA kernel introduced in the previous chapter. This kernel and its settings, for example thread management, could be simply reused in this experiments showing universality.

8. Histogram statistics

8.1 Motivation

With the first clustering experiments, I was able to cluster the Enron data to the clusters based on the features of nodes I was interested in. But it was difficult for me to verify the pre-defined clusters. I was able to say that my algorithm works correctly, but it was not clear how to interpret the results. The parameters used to describe the nodes did not use the full potential of data based on the communication. I did not use the frequency and amount of mutual communication between the nodes in the sufficient way. I wanted to better justify the clustering results and understand the links between the nodes. A meaningful solution would be to extend the set of parameters and fine-tune the existing ones.

But during the thoughts about extending the clustering experiment, new interesting questions appeared. In the very first clustering experiment, the nodes strictly reflected the folder structure of the Enron data set. This choice was meaningful from the point of view of my clustering. I wanted to have a fixed set of nodes and see how they can be divided into clusters. When I have a fixed set, the results of the algorithms with different setups are comparable. I was not interested in the discovery of a new, probably less meaningful nodes. But later, I have realized that the discovery of new nodes can be interesting for an improved understanding of the communication in this email-based social network. It means, that I have decided o ignore the folder structure of the Enron data set. Although, it would impact the number of new nodes, and it can provide a lot of new information.

Using this new, extended set of the nodes leads me to another problems and challenges. It is possible to observe, that the network based on the former set of nodes is quite well interconnected. On the other hand, the communication between the nodes from the new, extended set looks much sparser. I wanted to find some way, how to describe and visualize the communication in this network created from the Enron email data set by a different approach. But this idea leads to another challenge. When using only the nodes based on the folders, there was just a little danger of contaminating the results by unimportant nodes. Once each unique email address corresponds to a unique node, a lot of new nodes with a low importance appear. The visualization of this radically increased set of nodes would not be simple. It was another challenge of this algorithm to be able to filter only the most important nodes and visualize just them and their links.

8.2 Description of the applied approach

Specification of the terms

In this chapter, many programming structures and other entities are used. They use sometimes similar data structures and it could be unclear for the reader of this work to understand the actual meaning of the described structures. This is why I have decided to include the simple dictionary of the terms used in this chapter:

1. Node- In this chapter, mostly the graph related view to the data is used. Hence the definition of the node is crucial. The node corresponds to the person involved in the email communication. The node is represented by the email address of the particular person.
2. Edge- In the social network, communication between the nodes is an important part of descripting the network. In this chapter, edges are oriented and not bidirectional. The edge is oriented from the sender of the emails to the receiver. An edge is identified by 2 nodes, often also with the cardinality of this edge.
3. Cardinality of the edge – The frequency of communication between the nodes (number of sent emails) is represented by the cardinality of the particular edge.
4. List of the evaluated edges – A list of all the edges from the data set. Each edge is represented by 2 nodes and its cardinality. This list is used during the first data transformation.
5. Map of evaluated edges – It is a map of all the edges coming from the key nodes. Key node is a node used as the key in the map structure. Edges coming from the key node are represented by the 2nd node (first nodes are the same in the same record, they are the key nodes) and their cardinalities. In the Python programming language, this map is represented as a dictionary structure.
6. Ordered map of nodes- This map was created to keep the same order of the nodes. The key is the order of node and the value is the node itself. In Python programming language, this map is represented as dictionary structure.
7. Communication matrix -The final data structure of the experiments described in this chapter. It contains the cardinalities of all the edges between the chosen nodes. Both columns and rows of the matrix contain the same nodes. This matrix is implemented as a list of lists in the Python programming language.

Description of the algorithm
In the first step my algorithm transforms the extracts from the data set into a list of evaluated edges. The input of this algorithm is the raw Enron email data set. It means 450 MB of text files. The list is created in the following way:

Algorithm 7 : Generating the list of evaluated edges

-create an empty list of evaluated edges
-enter each subfolder in the current folder
    - open and read each file in the current folder
        - find and read the address of the sender node n_1
        - find and read all the receiver nodes $n_2 \ldots n_m$
    -create the tuples from the nodes n_1-n_2 … n_1 …n_m
    -append the tuple to the list of evaluated edges

The output of Algorithm 7 is the List of evaluated edges.
The tuple is represented as a simple string:

Example 2: The format of the tuple in the list of evaluated edges

sender_address-receiver_address

Using this string simplifies both the comparison between two tuples and its processing and further transformations. All the tuples are appended to the list of all

the tuples from all the messages. The tuples contained in the list do not have to be unique at this point of my algorithm. After all the emails are processed, the counter function is called on the list of tuples. As an output, a very convenient data structure is returned. The counter returns a unique list of tuples-edges with their cardinality. The list is ordered by the number of occurrences. The format reminds of JSON, but it is not a JSON :

Example 3: The format of data in the counter object

{'sender1-receiver1' : cardinality, 'sender2-receiver2' : cardinality…}

I have saved this output to the new text file. It was simpler for me to start from this point and avoid generating the list each time. Originally, I was thinking about transforming this almost JSON to JSON and then process it with the existing functions for processing JSON data. The main difference between the formats was that in my file, there were single quotes and JSON prefers double quotes. I have replaced the quotes and changed some less important differences in the formats. The reading as a JSON file did not work as I was expecting. I have discovered, that there are some issues in the data complicating the JSON transformation. There were still some blank spaces, and other characters blocking the JSON processing. I was not successful in removing these characters. However, I realized that the JSON would not provide me all the functionality I need and I still need to process most of these data like a plain string.
The ordered list of links was a step forward from the half a million set of email messages down to a manageable amount of data. In the next step, I have transformed the list of evaluated edges to the map of key-centric edges. In the beginning, the algorithm makes some necessary cleaning of the data. It removes of blank characters, line-ending characters etc.
The input of this algorithm is the List of evaluated edges

Algorithm 8 : Generating the map of key-centric edges

M: Map of key-centric edges
n1, n2 : nodes

-create an empty Map of key-centric edges M
-split the list of edges into the nodes + cardinalities
-for each edge in edges:
    -find node n_1
    -find node n_2
    -find cardinality

    -if the node n_1 is in the map M:
        -add the node n_2 + cardinality into the record of the node n_1
    -else:
        -create a new record for node n_1
        -insert the first value: node n_2 + cardinality of the link between n_1 and n_2

The email address of node n_2 and the value in the map of key-centric edges are delimited by a colon and the records are delimited by a semicolon. The value is again just a plain string with the following structure:

Example 4: Format of the record in the map of key-centric edges

receiver1_address:cardinality;receiver2_address:cardinality

The described structure is more helpful, because it is more node-centric rather than edge-centric. Each record in the map is a list of all the nodes that have an edge with the key-node with their cardinality.
In this step, the ordered map of nodes is also created. The key is the position and the value is the node. This map is used in the next step while creating the final matrix. Also, it servers to keep the same order of receiver nodes during all the steps.
The last step of data transformation was to create the so-called communication matrix.
The input of the algorithm is the map of key-centric edges

Algorithm 9: Generating of the Communication matrix

E: map of key-centric edges
O: Ordered map of nodes
M: Communication matrix

```
-for each record in the map of key-centric edges E
        -for each record in Ordered map of nodes O
                - if node from map of key-centric edges equals to the record from
                        -write the cardinality of edge  to the matrix M
                -else:
                        -write 0 cardinality (this means no edge)
```

The output of this step is a matrix describing the communication between the nodes.
The communication matrix resulting from the previous step already describes the frequency of communication between the nodes represented by their email addresses. But for the visual examination and evaluation a more graphical view would be suitable. So I have decided to plot the matrix of communication. I was considering more possibilities and ways how to plot this matrix, but in the end I have decided not to use any existing program and to plot it using my own program. This solution offers the best capabilities to modify the final plot for me, the colours, the way that it is displayed etc. I have used the python GUI library tkinter. This library offers the possibilities to create a simple window based application with standard objects like buttons, pictures, text fields and many more. For me, the most important feature was the ability to use the canvas for standard drawing. While computing the Communication matrix, the program was also looking for its biggest entry. While the plotting phase, this value was associated with the colour 0,0,0 in the RGB notation. This means that the biggest value will be labelled as black. The lowest non-zero entry is always the same, 1. This value was associated with the colour 255,255,255 in the RGB notation. This means that the smallest entry is labelled as white. The values between 1 and the maximal value are proportionally coded on the grey scale. In the beginning, I was thinking about giving the white colour the value 0. It would look naturally for the human reader. But many fields have values between 1% - 10% of

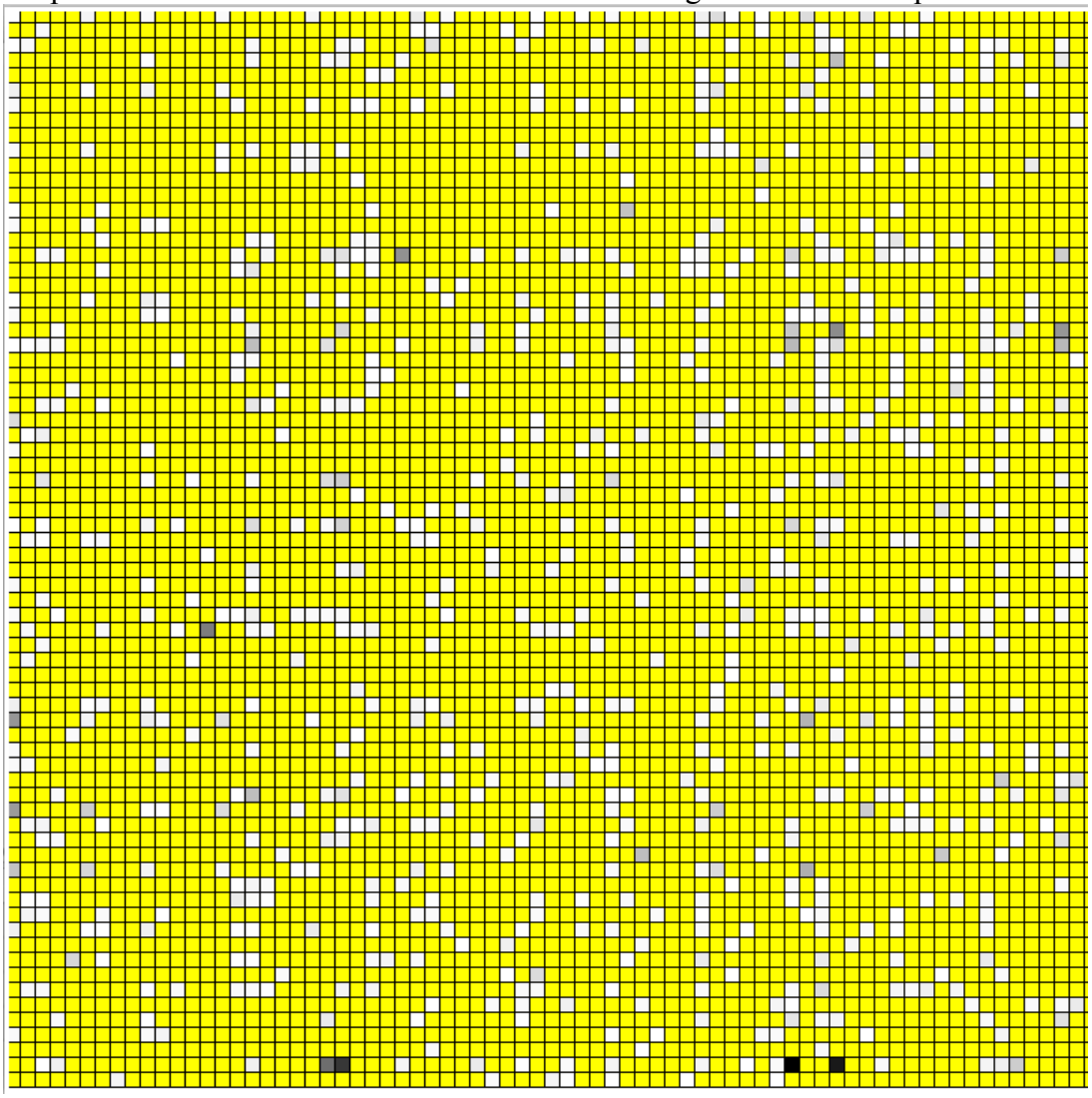
the maximal value. These fields would look almost like empty for the visual examination. That is why I have decided to plot the zero value in the yellow colour. The non-zero values are printed in grey scale, which means they cannot be mistaken for the zero value.

The input of this algorithm is the Communication matrix.

Algorithm 10: Plotting of the Communication matrix
M: Communication matrix i: one line from M s_{ij}: one value from M, placed in line number i and column number j
-for each line i in the Communication matrix M -for each position j in the line of Communication matrix -calculate the colour -plot the square s_{ij} in calculated colour

The default size of squares is 10x10 pixels.
The above described matrix plot offers a significantly better possibility for human visual evaluation. However, the original number-based matrix can serve as the input of the next experiments.

Graph 6: The matrix of communication among 72 most frequent senders



The data cleaning

The original results based on the Enron data set contained too much data for the visual examination of the results. The original list of links contained 18.5 MB of plain text. This corresponds to hundreds of thousands of links. These results are very difficult to display in a meaningful way. And calculating the matrix of communication as described in the previous section could take a significant amount of time. On the other hand, I have discovered, that the most frequent value of cardinality of links is simply 1. These links are probably not even important for the final results of the examination I needed. Therefore, it seems to be of advantage to prune the results and find just the most important subset. Most important subset should contain the nodes, which have above average amount of the edges of the cardinality higher than 1.

The first possibility is to use just the addresses of the senders. Another option would be to put the addresses of the senders as the rows of the matrix and the addresses of the receivers to the columns. (Or vice-versa) But I have decided to put the addresses of receivers on both, columns and rows. This decision was motivated with the fact, that the senders are probably really more important than the receivers. In many of the considered communications, the sender is always one and there is no doubt that it is a very important person for the email message. On the other hand, there could be more

receivers, some just as the receivers of the copy, sometimes not so closely related to the topic of the email. The receiver is always very closely related to the topic of the email.
After pruning, there remained 9660 addresses. It was an interesting progress, but still too many to plot the matrix. The next pruning step was to use just the people with the most contacts to other people from the same set. First, I have used just the contacts with at least 10 links in the same list. In the list of 9660 addresses, 10 links is a very weak condition. But this weak condition reduced the number of addresses radically. Only 1412 addresses had links with at least 10 another addresses. The following table shows the number of addresses with at least the defined amount of links in the data set.

Table 14: Number of patterns with the conditions of minimal number of links

Minimal number of links	**Number of addresses**
10	1412
20	641
30	387
40	267
50	178
60	143
70	116
80	97
90	82
100	72
300	9

This technique proved to be a good choice. The smallest matrix I used was the matrix of 9 contacts with at least 300 contacts. I have identified these 9 people mostly as the former managers of Enron. This observation confirms that this technique finds the most important people from the data set. In my experiments, I have then used the addresses with at least 100 links.

8.3 Examples of the data and the structures

In this section, I would like to show the examples of structures and used data described in the previous section. The amount of all the data or even the chosen 72 nodes is too big to be printed here. That is why I have decided to use only a very small set of nodes. For the examples in this section, I have used only the nodes, which fulfil the condition of at least 300 edges to another nodes from the set. There are only 9 of them. These nodes are represented by the following email addresses in the mail data set:

Table 15: The top 9 patterns represented by email addresses

tana.jones@enron.com
sara.shackleton@enron.com
kay.mann@enron.com
steven.kean@enron.com
jeff.dasovich@enron.com
sally.beck@enron.com
chris.germany@enron.com
gerald.nemec@enron.com
vince.kaminski@enron.com

List of evaluated edges

List of evaluated edges is not yet structured according to the chosen nodes. It is only ordered by descending cardinality of the described edges. Because of that, any meaningful example related to the 9 chosen nodes cannot be displayed. Following example shows the beginning of the List of evaluated edges:

Example 5 : The front of the Counter structure

```
Counter({'pete.davis@enron.com-pete.davis@enron.com': 18282,
'vince.kaminski@enron.com-vkaminski@aol.com': 8632,
'enron.announcements@enron.com-all.worldwide@enron.com': 4412,
'enron.announcements@enron.com-all.houston@enron.com': 3430,
'kay.mann@enron.com-suzanne.adams@enron.com': 3186,
'jeff.dasovich@enron.com-richard.shapiro@enron.com': 2700,
'vince.kaminski@enron.com-shirley.crenshaw@enron.com': 2478, …})
```

Map of evaluated edges

The map of evaluated edges in this example contains only the 9 records of the 9 mentioned nodes. But each record contains not only the edges between theses 9 nodes. It contains all the edges coming from the particular node. It means, that the whole example is still too big for the purpose of this section. So the following example contains just one single record from the map. It is the record containing edges coming from the node gerald.nemec@enron.com and their cardinality.

Example 6 : The record of Gerald Nemec from the Map of evaluated edges

gerald.nemec@enron.com ---> eric.gillaspie@enron.com: 266;mark.whitt@enron.com: 224;mark.knippa@enron.com: 155;barry.tycholiz@enron.com: 141;dan.bump@enron.com: 134;gtownsend@manorisd.net: 110;mark.courtney@enron.com: 108;stephanie.miller@enron.com: 104;ryan.f.ruppert@exxon.sprint.com: 104;chris.hilgert@enron.com: 93;kckrisa@apex2000.net: 79;bswaldrop@aol.com: 73;paul.lucci@enron.com: 68;kay.young@enron.com: 65;debra.perlingiere@enron.com: 62;susan.scott@enron.com: 62;joan.quick@enron.com: 59;steve.hooser@enron.com: 48;david.foti@enron.com: 48;janel.guerrero@enron.com: 47;tyrell.harrison@enron.com: 47;lisa.nemec@enron.com: 46;paul.pfeffer@bakerbotts.com: 45;pat.radford@enron.com: 44;russell.diamond@enron.com: 43;mary.ogden@enron.com: 40;barbara.gray@enron.com: 40;chris.meyer@enron.com: 40;carlos.alatorre@enron.com: 38;brian.hendon@enron.com: 36;pmelcher@aol.com: 36;brant.reves@enron.com: 34;kim.ward@enron.com: 34;colleen.sullivan@enron.com: 32;katie99@tamu.edu: 31;cris.sherman@enron.com: 30;jamie.ginsberg@enron.com: 30;ken.choyce@enron.com: 30;eva.neufeld@enron.com: 30;hbrown@bracepatt.com: 30;shonnie.daniel@enron.com: 29;gnemec@houston.rr.com: 29;michael.legler@enron.com: 28;ryan.ruppert@exxonmobil.com: 28;scott.josey@enron.com: 28;judy.thorne@enron.com: 28;drew.fossum@enron.com: 28;greg.brazaitis@enron.com: 28;arlanz@mexis.com: 28;bruce.rudy@enron.com: 26;peter.keohane@enron.com: 26;staci.holtzman@enron.com: 26;ernie.j.zavaleta@fritolay.com: 26;kevin.howard@gcm.com: 24;hbinder@porterhedges.com: 24;eric.ledain@enron.com: 24;ednemec@earthlink.net: 24;carl.carter@enron.com: 24;dave.fuller@enron.com: 24;steve.schneider@enron.com: 24;rae.meadows@enron.com: 23;andrew.miles@enron.com: 22;brian.bierbach@enron.com: 22;dan.hyvl@enron.com: 22;david.marshall@enron.com: 22;steve.pruett@enron.com: 22;heather.kroll@enron.com: 22;theresa.staab@enron.com: 20;john.kiani@enron.com: 20;paul.miller@enron.com: 20;jean.mrha@enron.com: 19;brian.redmond@enron.com: 19;robert.walker@enron.com: 18;clement.abrams@enron.com: 18;jim.schwieger@enron.com: 18;david.owen@enron.com: 18;james.hoff@enron.com: 18;anncrawford@aec.ca: 18;htebs@huber.com: 17;ellen.wallumrod@enron.com: 17;stuart.zisman@enron.com: 16;julie.gomez@enron.com: 16;ned.higgins@enron.com: 16;rudwell.johnson@enron.com: 16;elizabeth.sager@enron.com: 16;travis.mccullough@enron.com: 16;john.grass@enron.com: 14;jcoleary@porterhedges.com: 14;rebecca.cantrell@enron.com: 14;nnachawa@central.uh.edu: 14;rakhi.israni@enron.com: 14;rhett.jackson@enron.com: 14;darren.vanek@enron.com: 14;john.hodge@enron.com: 14;mbeckworth@velaw.com: 14;nick.cocavessis@enron.com: 14;tbuffington@hollandhart.com: 14;steve.morse@enron.com: 14;marlene.hilliard@enron.com: 13;rusty.belflower@enron.com: 13;lisa.druzbik@enron.com: 13;t..lucci@enron.com: 13;mark.taylor@enron.com: 12;mandy.bowles@email.moore.com: 12;stephanie.balette@enron.com: 12;keegan.farrell@enron.com: 12;robert.hill@enron.com: 12;anne.koehler@enron.com: 12;twporter@porterhedges.com: 12;lauri.allen@enron.com: 12;howton.k@ghc.org: 12;mhughes@hollandhart.com: 12;joego@calpine.com: 12;jastone@southernco.com: 12;nelson.ferries@enron.com: 11;eva.rainer@enron.com: 11;chris.sonneborn@enron.com: 10;csanmarco@kkrw.com: 10;james.haden@enron.com: 10;mark.skoyk@enron.com: 10;paul.pfeffer@enron.com: 10;greg.johnston@enron.com: 10;ipc_oil@msn.com: 10;mike.smith@enron.com: 10;edward.gottlob@enron.com: 10;roger.balog@enron.com: 10;nabil@enron.com: 10;scott.monson@enron.com: 10;brad.blevins@enron.com: 10;alan.bransgrove@xemkt.com: 10;bo.barnwell@enron.com: 10;bruce_rudy@mailcity.com: 10;andrew.edison@enron.com: 9;pkdaigle@neosoft.com: 9;rab@beggslane.com: 9;raimund.grube@enron.com: 9;bill@katzlaw.com: 9;matt.gillaspie@brentonbank.com: 8;aavalos@pnm.com: 8;lhand@lemle.com: 8;hbender@porterhedges.com: 8;bgpenn@marathonoil.com: 8;houston<.ward@enron.com>: 8;emmye@dracospring.com: 8;barbara.waldrop@cox.com: 8;bates@tristategt.org: 8;bradb@calpine.com: 8;ron.tapscott@enron.com: 8;paul_pfeffer@bakerbotts.com: 8;chris.germany@enron.com: 8;arnold.eisenstein@enron.com: 8;robert.kilmer@enron.com: 8;kckrisa@worldnet.att.net: 8;john.dushinske@enron.com: 8;mconner@plainsgt.org: 8;mawhitt@aol.com: 8;hsmiller@msn.com: 8;lauragammell@hotmail.com: 8;lance.legal@enron.com: 8;lal.echterhoff@enron.com: 8;teb.lokey@enron.com: 8;trevor.woods@enron.com: 8;snemec@shelton.org: 8;candace.bywaters@enron.com: 8;ken.loch@enron.com: 8;cheryl.king@enron.com: 8;richard.shepherd@enron.com: 8;tom.shelton@enron.com: 8;don.baldridge@enron.com: 8;ed.mcmichael@enron.com: 8;ruth.concannon@enron.com: 8;tsanthon@southernco.com: 8;susanc@katzlaw.com: 8;susan.elledge@enron.com: 7;stacey.richardson@enron.com: 7;nora.dobin@enron.com: 7;mtgoper@misnet.com: 6;john.gordon@enron.com: 6;louis.dicarlo@enron.com: 6;neta.zitnik@enron.com: 6;patricia_cuero@dell.com: 6;bieraugel@efortress.com: 6;amcmullen@canscot.com: 6;ronald_brown@kindermorgan.com: 6;audrey.oneil@enron.com: 6;dena.pawlowski@enron.com: 6;wsnyder@saylid.com: 6;kenneth.kaase@enron.com: 6;oscar.dalton@enron.com: 6;frank.vickers@enron.com: 6;e..haedicke@enron.com: 6;arvel.martin@enron.com: 6;mlojo@truecos.com: 6;ken.kaase@enron.com: 6;jbates@tristategt.org: 6;richard.sanders@enron.com: 6;david.bargainer@enron.com: 6;peter.meier@neg.pge.com: 6;davis.thames@enron.com: 6;@enron.com: 6;jack.simunek@enron.com: 6;jordan.mintz@enron.com: 6;sophiedavenport@hotmail.com: 6;teresa.bushman@enron.com: 6;jeffrey.hodge@enron.com: 6;andy.unverzagt@enron.com: 6;morris.clark@enron.com: 6;steven.curlee@enron.com: 6;lisa.hesse@enron.com: 6;mark.castiglione@enron.com: 6;jessica.wentworth@enron.com: 6;ssnyder@shelton.org: 6;virginia.c.levenback@williams.com: 6;kpearson@bmoh.com: 6;ted.bland@enron.com: 6;dwight.beach@enron.com: 6;janet.edwards@enron.com: 6;jeff.gilliam@enron.com: 6;legal<.schuler@enron.com>: 6;kjurica@syntexsolutions.com: 6;karen.gruesen@enron.com: 5;jennifer.d.sanders@us.andersen.com: 5;becky.spencer@enron.com: 5;shahnaz.lakho@enron.com: 5;ktycholiz@houston.rr.com: 5;mgillaspie02@pjc.com: 5;leslie.hansen@enron.com: 5;jim.coffey@enron.com: 5;dax@uamps.com: 4;sara.davidson@enron.com: 4;ryan.f.ruppert@exxon.sprint.net: 4;george.weissman@enron.com: 4;karen.jones@enron.com: 4;jdecker@entergy.com: 4;sandi.braband@enron.com: 4;parking.transportation@enron.com: 4;btipton@flowserve.com: 4;ron.coker@enron.com: 4;s..ward@enron.com: 4;roger.ondreko@enron.com: 4;kgreen@tristategt.org: 4;patrick.wade@enron.com: 4;kwebster@eprod.com: 4;jdixon@calpine.com: 4;dax@uams.com: 4;rstribling@tristategt.org: 4;erica.braden@enron.com: 4;heidi.withers@enron.com: 4;katiemcmahon99@hotmail.com: 4;ann.white@enron.com: 4;renee.alfaro@enron.com: 4;julia.murray@enron.com: 4;kbradley@reliantenergy.com: 4;jackie.morgan@enron.com: 4;mark.h@mail.utexas.edu: 4;paul.drexelius@cinergy.com: 4;jzivley@earthlink.net: 4;amy.felling@enron.com: 4;lisa.mellencamp@enron.com: 4;kimberly_evans@reliantenergy.com: 4;dorothy.mccoppin@enron.com: 4;michael.eiben@enron.com: 4;mmiller3@austin.rr.com: 4;michel.nelson@enron.com: 4;lmnemec@shellus.com: 4;pnemec@msn.com: 4;rswindle@bmoh.com: 4;janet.wallis@enron.com: 4;kfleuriot@hou.projectconsulting.com: 4;gerald.lofton@enron.com: 4;suzanne.adams@enron.com: 4;hjosey@aol.com: 4;mary.clark@enron.com: 4;dclearfield@wolfblock.com: 4;audrey.pullen@enron.com: 4;connie.sutton@enron.com: 4;cheryl.marshall@enron.com: 4;dixonr@flash.net: 4;joe.parks@enron.com: 4;hitekdude55@netscape.net: 4;bootes@plainsgt.org: 4;jtaylor@plainsgt.org: 4;jstafford@patinaoil.com: 4;t..hodge@enron.com: 4;dina.snow@enron.com: 4;jnemec@riviana.com: 4;todonnell@hollandhart.com: 4;rstribling@tristategt.orgj: 4;svanhooser@houston.rr.com: 4;miguel.vasquez@enron.com: 3;w..vickers@enron.com: 3;n..gray@enron.com: 3;ejenkins@kelleydrye.com: 3;kyle.purvis@enron.com: 3;max.sonnonstine@enron.com: 3;richard.deming@enron.com: 3;fillip@gateway.net: 3;becky.zikes@enron.com: 3;body.shop@enron.com:

3;dean.drozdiak@enron.com: 3;swilliams@reliant.com: 3;louis.allen@enron.com: 3;brudy@umich.edu: 3;credit<.williams@enron.com>: 3;teresa.mandola@enron.com: 3;amelia.alland@enron.com: 3;gary.bryan@enron.com: 3;maureenperea@hotmail.com: 3;swalsh@powereng.com: 3;stacey.neuweiler@enron.com: 2;joanne.rozycki@enron.com: 2;mawhowel@southernco.com: 2;kim.stanley@enron.com: 2;carol.clair@enron.com: 2;nicole.cortez@enron.com: 2;lillie.pittman@enron.com: 2;william.e.nicas@exxon.com: 2;mary.johnson@enron.com: 2;

Communication matrix

Using the algorithm described in the previous section, the following communication matrix was generated. The order of the lines and columns is the same as the order of email addresses in the beginning of this section. This matrix is used for plotting the matrix and as an input to the next experiments described in the following chapter.

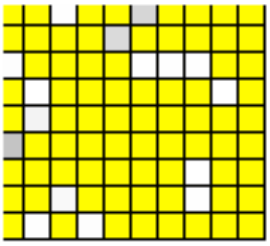
Table 16: The communication matrix of top 9 patterns represented by email addresses

	sara.shackleton@enron.com	steven.kean@enron.com	kay.mann@enron.com	sally.beck@enron.com	jeff.dasovich@enron.com	tana.jones@enron.com	gerald.nemec@enron.com	chris.germany@enron.com	vince.kaminski@enron.com
sara.shackleton@enron.com	0	0	18	0	0	505	0	0	0
steven.kean@enron.com	0	0	0	0	456	0	0	0	0
kay.mann@enron.com	22	0	0	0	0	4	12	16	0
sally.beck@enron.com	0	4	0	0	0	0	0	0	6
jeff.dasovich@enron.com	0	153	0	0	0	0	0	0	0
tana.jones@enron.com	715	0	0	0	0	0	0	0	0
gerald.nemec@enron.com	0	0	0	0	0	0	0	8	0
chris.germany@enron.com	0	0	70	0	0	0	0	11	0
vince.kaminski@enron.com	0	19	0	26	0	0	0	0	0

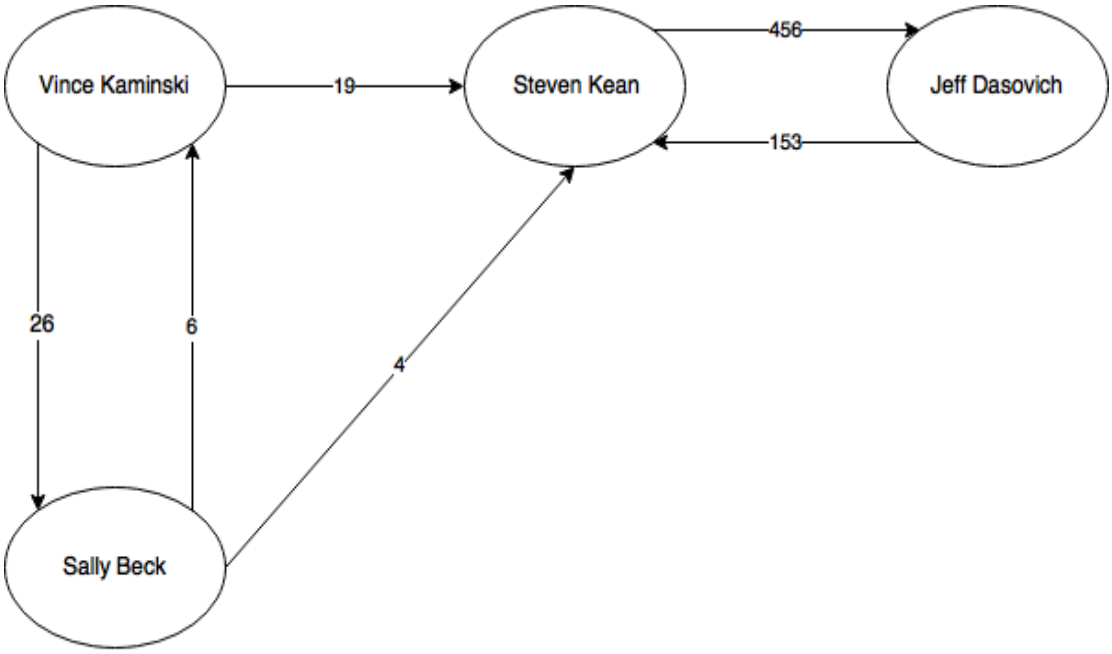
Plotted matrix

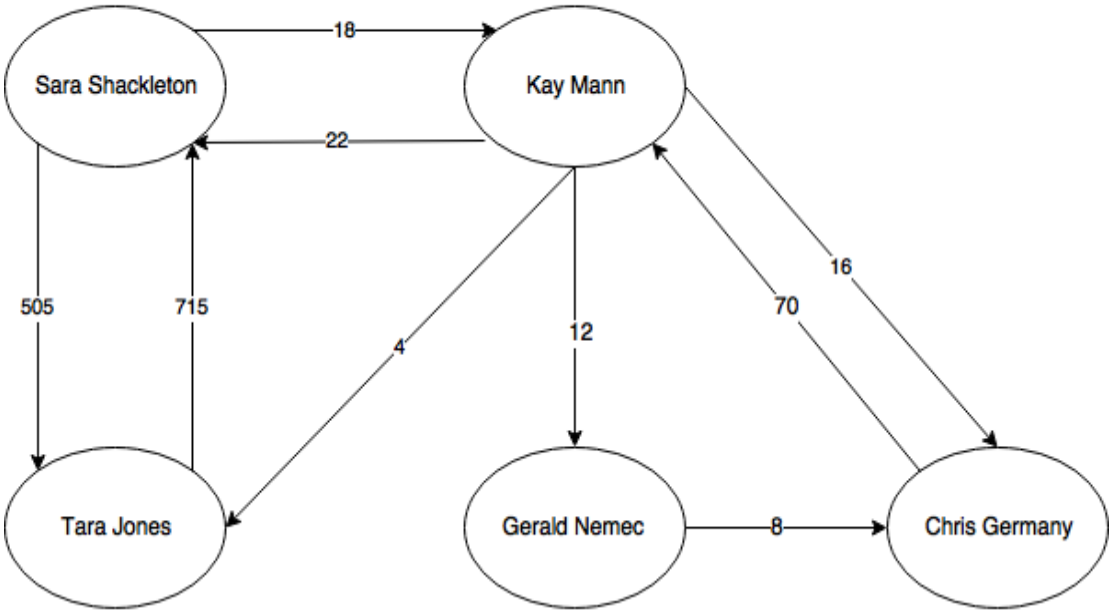
The following matrix is the visualization of the Communication matrix. It was displayed by the algorithm described in the previous section. We can see how sparse this matrix is. As we see in the next example, most of the members of this set did not even communicate together and there are 2 subgroups created.

Graph 7: The matrix of communication of 9 most frequent senders



Graph 8: visualisation of the nodes and edges





Note: the node chris.germany@enron.com contains also the loop edge. This edge is not meaningful for the graph so it was omitted.

8.4 Conclusion

In this chapter, I have examined the Enron data set from a different point of view than in the previous experiments. I was able to visualize the communication between the nodes in a user-friendly way for visual examination. I have faced the big data problem and I needed to find and display only the most important nodes and their communication. The data structure I have created is not only helpful for visual examination but it can also serve as the input data for the next experiments. It is a significant step forward for the next experiments, because the matrix of values is much more simple to be used than the half a gigabyte of plain text messages. It is also possible to find interesting patterns in the data. For example, I would expect that 9 most important nodes would be interconnected. In the reality, I have realized that the communication in the whole data set is irregular. Even the testing 9-member matrix is divided into 2 disconnected subgraphs.

The techniques used to clean the results proved to be appropriate. The most important persons according to my techniques are also important persons from the hierarchy of Enron. For example, Steven Kean worked as a chief of staff of Enron, Kay Mann served as Assistant General Counsel in Enron or Chris Germany served as a transportation specialist for 14 years in Enron. Also none of the addresses for corporate bulk mail sending made it to the most important nodes. The above-mentioned facts indicate that the performed experiments were meaningful and helpful.

9. Hopfield net experiments

9.1 Motivation

There are many different views on the social networks and many different tasks connected to data mining them. I have used the clustering and performed experiment to find the most important patterns in the set. All these experiments were based on the known communication, based on the Enron email set. But it could be an interesting task to try to find the and fill the unknown communication between the patterns in the social network. For this purpose, I can use neural networks such as Hopfield net.

9.2 Hopfield net description

Neural networks were rediscovered in the 1980`s after a decade of stagnation. The reason was the discovery of back propagation algorithm, which solved the exclusive-or training problem. One of the popular algorithms from that age is Hopfield net.
The Hopfield net is a recurrent auto associative neural network introduced in 1974 by Little[14] but popularised in 1982 by John Hopfield[15]. This network is well suitable for the recognition of patterns based on a predefined training set.
A Hopfield network consists of a set of mutually interconnected neurons. Each neuron is connected to every other neuron in the network, except to itself. There are no loop-connections in the Hopfield network. Input or output values of the neurons are typically just -1 or 1. The values of neurons are updated asynchronously. The update of each neuron depends not only of the neuron itself but also on the values of other neurons in the net.

Asynchronous network

The description of Hopfield net is based on the book of Raul Rojas[16] .
Hopfield net works as an asynchronous network. It means that each neuron computes its value at random times. The order of the updated neurons is also chosen randomly. At each time, only the value of one neuron is computed. There is no parallelism possible. I assume, that there is no time delay between the computation of the new value and update of this value on particular neuron. We adopt the additional simplification that the state of a neuron
is not changed if the total input is zero. This means that we leave the sign function undefined for the argument zero. Asynchronous networks are of course more realistic models of biological networks, although the assumption of zero delay in the computation and transmission of signals lacks any biological basis.

9.3 Description of the applied approach

The input pattern

As the input patterns, I have used the communication matrix described in the previous chapter. However, I needed to make some modifications. For the Hopfield net, the expected input data is just a plain bipolar matrix indicating if there is a connection between the respective neurons or not. Also the condition of no loops is

requested. My input matrix is a symmetric 72x72 matrix. There are not just bipolar values, but any value describing the number of sent messages. Also the condition of no loops might be not preserved. It means that my input matrix should be transformed before it could be used in the Hopfield net. The preparation algorithm iterates my matrix and each value greater than 0 turns to 1. 0 values are turned to -1. Also, the values on the diagonal are turned to 0, the original value has no effect. The diagonal must contain only zero values because of the condition of no loop connections between neurons.

The weight matrix

The weight matrix is crucial for working of the neural network. It can be described in the following way:

Formula 8: The weight matrix

(8)

$$W = \begin{pmatrix} w_{11} & w_{12} & \cdots & w_{1n} \\ w_{21} & w_{22} & \cdots & w_{2n} \\ \vdots & & \ddots & \vdots \\ w_{n1} & \cdots & \cdots & w_{nn} \end{pmatrix}$$

Where w_{ij} can be interpreted as an influence of neuron i over neuron j and reciprocally.

The value of each w_{ij} is calculated in the following way:

Formula 9: Calculation of the element w_{ij} of the weight matrix W

(9)

$$Wij = \sum_{k=1}^{m} \big((inputMatrix[k][i])\big) * \big((inputMatrix[k][j])\big) \text{ for i!=j}$$
$$\text{and } Wij = 0 \text{ for i=j}$$

Where:
i is the number of the row in the weight matrix W
j is the number of the column in the weight matrix W
m is the dimension of the pattern

The testing input vector

The choice of the testing vector is important, because in the future, this vector will be the input for which I want to find the correct fitting pattern. For the testing, I have used one of the rows from the original input matrix. I have randomly damaged about 10%~15% of the values. I did not test the algorithm with more damaged values, because the amount of rows in the input matrix is quite high and similarity between rows is also significant.

This input vector is represented with the vector of neurons of the same length. Each neuron has the corresponding value from the vector assigned.

The convergence of the net

The Hopfield net should converge to a local state. Convergence indicates, that the values are not changing anymore and the network reached the possible answer about similarity of the testing input vector and one of the many training patterns.

During each iteration, random neuron is chosen. Its value is recalculated according to the following formula:

Formula 10 : Recalculation of the value of one neuron

$$v = sgn(\sum_{i=1}^{m} neuron[j] * weightMatrix[i][j]) \quad (10)$$

Where:
Neuron is the pattern presented to network.
J is the number between 1 and dimension of the pattern.
I is dimension of the network
m is the length of neuron

In the end of each iteration the new neuron vector is compared to the old one. If the vector does not change ,the network is considered as converged and the algorithm stops.

Algorithm 11: Calculating the weight matrix

Used variables:
i,j,k - iterators
weightMatrix – weight matrix
inp – array of training values

```
initialize empty weightMatrix
for i within the range 0 and dimension
        for j within the range 0 and dimension
                if i==j:
                        weightMatrix[i][j]=0
                else
                        suma=0
                        for k within the range 0 and length of inp
                                suma=suma+inp[k][i]*inp[k][j]
                        weightMatrix[i][j]=suma
```

The next step is recalling the pattern equal or similar to some pattern in the training data. Especially the ability to recall the pattern, which is only similar to the pattern in

the training data, is interesting from the point of view of data mining. It is possible to discover similarity between the patterns, which is not clear on the first sight.

Algorithm 12: Recall of the pattern using Hopfield net

Variables:
T: The matrix of training data
W: The weight matrix
k, i, j: iterators
n: number of rows of W
m: number of columns of W
sum: partial calculation of value of one neuron
neurons: the vector containing the values presented to the network
original_neurons: the vector of values of neuron vector from previous iteration

-create the matrix of input data *T*
-calculate the weight matrix *W* based on *T*
-while the network did not converge
    -for *k* in range within 0 and *n*
        -find random *j* within the range 0 and *n*
        - for *i* in range 0 , *m*
            sum+=(*neurons*[*j*])*(*W*[*i*][*j*])
            -if *sum* >0 then *neurons*[*j*] =1 else *neurons*[*j*] =-1
    -compare the *original_neurons* with *neurons*
        -if *original_neurons* and *neurons* do not differ
            -network converged

9.4 Using Hopfield network to predict the communication links

Hopfield network is often used for recovering fuzzy data, recognizing graphical patterns and many other tasks. But I would like to use this algorithm in the way helpful for my work. Using the Hopfield net for data mining is also possible. But it is necessary to modify the input data and task in the way Hopfield net can process it.

As I have mentioned in this chapter, I have used the data from the communication matrix from the chapter about histogram statistics. This matrix needed to be transformed to fill the needs of the algorithm, which means only zeros on the diagonal and only -1 or 1 values elsewhere.
On the Internet, there are many examples of a successful use of Hopfield network with the similar input data. But these artificial input data have often comparable amount of zeroes and ones. However, my real data contain mostly zeros (-1 in my format). My program is suffering some inaccuracy because of this fact.

In my experiment, I am simulating the situation when I have only partial information about the communication of the some pattern (person). But I assume that the unknown node is from the same group of people, part of the same communication graph so I can expect that there are at least some mutual connections. I can represent

the known connection state with 1 respectively -1 like in the whole Hopfield net. The unknown state of communication is represented by zero.
The main idea is to present the vector like this to Hopfield network and let the network fill the missing parts of communication (zeros) with the recognizable states of communication (1, -1) based on the training data.
The result of this experiment is the prediction of the communication of the new node with the rest of the nodes present in the training data.

Problems of my Hopfield net experiment
As the training data of my Hopfield net, I have used communication matrices created in the histogram statistics experiment. I wanted to use the small one (Graph 7) and then the big matrix (Graph 6). However the experiment with the small matrix was already unsuccessful.
The weight matrix was correctly calculated and program tried to recall the presented pattern.
But the recall always failed, converging to the vector of -1s. This problem repeated for each presented pattern.
The reason why this experiment was unsuccessful is the limited capacity of the Hopfield network. The matrix with dimensions 9x9 was already too big for the network to remember. I have also tried the bigger matrix, 72x72 but the result was the same, which was not surprising.
I decided to reduce the number of patterns in the training data. The input matrix was reduced in the following way:

Example 7: the training data of Hopfield net experiment

[[**0, 0, 1, 0, 0, 1, 0, 0, 0**],
[**0, 0, 0, 0, 1, 0, 0, 0, 0**],
[**1, 0, 0, 0, 0, 1, 1, 1, 0**],
[0, 1, 0, 0, 0, 0, 0, 0, 1],
[0, 1, 0, 0, 0, 0, 0, 0, 0],
[1, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 1, 0],
[0, 0,1 , 0, 0, 0, 0, 1, 0],
[0, 1, 0, 1, 0, 0, 0, 0, 0]]

The patterns marked bold belong to the reduced set used in the experiment. In this matrix, there are ones and zeros. The zeros are converted to -1 on the input of algorithm. The algorithm accepts 1 and -1 only.

The results of my Hopfield network experiment
For my first experiment, I have used the training data displayed in the Example 6. After the network was trained, the following pattern was presented to the network:

Example 8: The pattern presented to the Hopfield net

[0, -1, 1, 0, -1, 0, -1, -1, -1]

This pattern is based on the first row in the training data. 3 values were hidden, replaced by 0. This experiment simulates the situation, when the communication is

just partially known. Unknown communication is represented by zeros. The task of the Hopfield net is to identify the pattern and correctly fill the missing information.

Example 9: The output of Hopfield net after the pattern from Example 7 was presented to it.

[-1, -1, 1, -1, -1, 1, -1, -1, -1]

The answer is correct; Hopfield net correctly filled the missing information about communication.
The program calculated and used the following weight matrix:

Example 10: The weight matrix from my Hopfield net experiment

[0, 1,-1, 1,-1, 1, 3, 3, 1]
[1, 0, 1, 3, 1,-1, 1, 1, 3]
[-1, 1, 0, 1,-1, 1, -1,-1, 1]
[1, 3, 1, 0, 1,-1, 1, 1, 3]
[-1, 1,-1, 1, 0,-3,-1,-1, 1]
[1,-1, 1,-1,-3, 0, 1, 1,-1]
[3, 1,-1, 1, -1, 1, 0, 3, 1]
[3, 1,-1, 1,-1, 1, 3, 0, 1]
[1, 3, 1, 3, 1, -1, 1, 1, 0]

Parameters of execution
My implementation of Hopfield net usually converges in the 2nd iteration. The program runs less the 2 seconds on the average laptop computer (Core 2 Duo, 2.53 GHz CPU with 8 GB DDR3 RAM) with the described data, so no advanced hardware was necessary.

9.5 Conclusion

Hopfield net is one of the simpler neural network algorithms, but it can be used for link prediction based on the data like my data from Enron data set.
The biggest challenge was the limited capacity of the network. With the reduction of patterns in the input data, it is possible to achieve the results with interesting success rate and be able to use this program to predict the unknown connections in the social network.
However I did not succeed to use the whole output matrices of the Histogram statistics experiment as the training data of Hopfield network. The reason was the mentioned capacity problem.

10. Bidirectional Associative Memories

10.1 Motivation

Hopfield net is obviously not the only type of neural network capable of capturing unrelated relationships among the considered objects. Another type is the Bidirectional associative memory (BAM). BAM was first introduced by Bart Kosko[17] in 1988. The main difference between the Hopfield network and a BAM is that BAM uses 2 layers of neurons instead of one. It is also a hetero-associative type of memory, which means that BAM can return another response to the presented input pattern.

10.2 Description of the BAM algorithm

BAM accepts a limited number of pairs of bipolar strings as the input data. It stores them in its weight matrix. The main goal is the ability to recall the second part of the pair, after the first one is presented to BAM. Also the string which is close to the one of the known strings should be recognized as one of them. This idea is similar to the Hopfield network.
The correctness of the result depends on many things, for example on the size of the input data. Another important attribute is the quality of the data. The input data in the examples have often about the same number of minus ones and ones. This situation is ideal, but it sometimes does not happen when dealing with the real data. As I have mentioned before, my data consists mostly from minus ones.

The first step while using BAM is the calculation of the weight matrix, which encodes the pairs of input data:

Algorithm 13: Computation of BAM matrix
Input of the algorithm: Set of couples of strings containing zeros or ones *P*
Output of the algorithm: BAM matrix *M*
Variables: P: set training patterns p: training pattern P x,y: pairs of strings from p i,j : iterators M: calculated BAM weight matrix
-convert values of *P* to bipolar -for each pair *p* from *P* -*x*= first string from pair *p* -*y*= second string from pair *p* -for *i* in the range 0 to length of *x* -for *j* in range 0 to length of *y* -*M* [*i*][*j*]= *M* [*i*][*j*]+*x*[*i*]**y*[*j*]

The next step is the identification of the pattern presented to the network. The pattern presented to the network is new and it is not contained in the training data.

Algorithm 14: recall of the associated string from BAM network
Input of the algorithm: BAM weight matrix *M,* input pattern
Output of the algorithm: recalled pattern
Variables: M: calculated BAM weight matrix s : the input string presented to BAM r: the actual output x,y,i: iterators
-initialize r as vector of zeros -for *x* in range 0 to number of columns in *M* -for *y* in range 0 to number of rows in *M* -*r* [*x*]= *r* [*x*] + *s* [*y*]* *M* [*x*] [*y*] -for *i* in range 0 to length of *r* -if *r*[*i*]<0 then *r*[*i*]=-1 else *r*[*i*]=1

The result of the described algorithm is the string recalled from the BAM.

10.3 Results of my experiment with BAM

In this work, there are mentioned several algorithms which were more or less suitable for my experiments and for my data. Some of them needed to be slightly modified to work in the way I wanted. But BAM was one of the more serious challenges.

As the input, I have used the communication matrix of the 9 most important persons described in the chapter about Histogram statistics. The task was similar to that one formulated in the Hopfield net experiment, which means I wanted to be able to identify potentially unknown string based on the input string.

The problem I was facing is the general similarity of the input data. These strings are very sparse, mostly consisting of zeros (minus ones in my program). About 80% of the data are zeros.

With this low number of ones, BAM works reliably with very low number of input pairs only. The 9 pairs from my example are already too much. The results are not satisfying.

Example 11: Input pairs of my BAM program
[[0, 0, 1, 0, 0, 1], [0, 0, 0]], [[0, 0, 0, 0, 1, 0], [0, 0, 0]], [[1, 0, 0, 0, 0, 1], [1, 1, 0]], [[0, 1, 0, 0, 0, 0], [0, 0, 1]], [[0, 1, 0, 0, 0, 0], [0, 0, 0]], [[1, 0, 0, 0, 0, 0], [0, 0, 0]], [[0, 0, 0, 0, 0, 0], [0, 1, 0]], [[0, 0, 1, 0, 0, 0], [0, 1, 0]], [[0, 1, 0, 1, 0, 0], [0, 0, 0]]

The input of the program were the pairs of strings based on the communication matrix of 9 most important persons from the Histogram statistics chapter

Example 12: Example output of my BAM program, 1st experiment

```
[0, 0, 1, 0, 0, 1] --->  [1, 1, 1]
[0, 0, 0, 0, 1, 0] --->  [1, 1, 1]
[1, 0, 0, 0, 0, 1] --->  [1, 1, 1]
[0, 1, 0, 0, 0, 0] --->  [1, 0, 1]
[0, 1, 0, 0, 0, 0] --->  [1, 0, 1]
[1, 0, 0, 0, 0, 0] --->  [1, 1, 1]
[0, 0, 0, 0, 0, 0] --->  [1, 1, 1]
[0, 0, 1, 0, 0, 0] --->  [1, 1, 1]
[0, 1, 0, 1, 0, 0] --->  [1, 0, 1]
```

In the second experiment, I have decreased the number of couples in training data from 9 to 3. The first 3 couples from the example 2 were chosen. However, the results were still incorrect:

Example 13: Example output of my BAM program, 2nd experiment

```
[0, 0, 1, 0, 0, 1] --->  [1, 1, 1]
[0, 0, 0, 0, 1, 0] --->  [0, 0, 1]
[1, 0, 0, 0, 0, 1] --->  [1, 1, 1]
```

The program calculated and used the following weight matrix in the second experiment:

Example 14: The weight matrix of BAM network during the second experiment

```
[[3, 3, 1],
[ 1,  1, 3],
[-1, -1, 1],
[ 1,  1, 3],
[-1, -1, 1],
[1,  1, -1]]
```

10.4 Conclusion

BAM network was not able to work correctly with so low number of inputs as 9. The main cause is probably the nature of these inputs. They consist mostly of zeros and they are very similar. BAM consists of 2 layers of neurons unlike Hopfield net, but it is probably less suitable for the tasks connected with data mining on the Enron email corpus based communication matrix as I have used.

11. PageRank algorithm

11.1 Motivation

There are many ways how to evaluate the importance of the nodes and patterns in a social network. When the social network is represented in a correct way, also the algorithms not originally meant for social networks can be used for its analysis. One of possible algorithms is the algorithm called PageRank. This algorithm was originally developed for ranking of mutually interconnected web pages. PageRank was introduced in the year 1998 by the founders of Google Inc, Sergey Brin and Larry Page. The search engine of Google is build on the PageRank algorithm. But it is of course not the only one criterion for marking a page as important by Google search engine. PageRank is also patented by Google Inc and it is named after Larry Page.[18]

The description in this chapter is based on the description from the original paper of Google founders [19] and the description from the book of Bing Liu. [20]

The main idea

PageRank is based on a democratic-like idea of voting. In the beginning, pages vote for the most important pages in the current network. This voting is based on the so-called hyperlinks between pages. If there is a hyperlink from page A to page B, the PageRank algorithm interprets this hyperlink as the vote of page A for the page B. The page with a higher amount of votes becomes more important. The votes from important pages are more valuable to the voted pages. A higher PageRank value of the more important page means for example a better position within search engine results, or a higher importance of information from this page. The main goal is to measure relative importance of the documents in the given set.

11.2 Algorithm description

The PageRank of each document in the network is calculated independently, based only on the links. Search queries are not taken into account.

Each document in the network evaluated by PageRank has the following attributes:

In-links : hyperlinks pointing on the particular document. The loop-links are usually ignored.

Out-links : hyperlink pointing to the other documents from the particular document.

The importance of the particular document is measured by the so-called prestige.

The more in-links the particular document has, the higher is its prestige. The documents pointing to the particular document also have their own prestige and these links are more important and help to increase the prestige of the particular document more. The resulting prestige is called PageRank score.

The links between the patterns are represented with the input matrix, similar to the idea of communication matrix used in the previous chapters. This matrix should be stochastic, irreducible and aperiodic.

If there is a document with no outbound links, it would be disconnected from the graph and the calculation would not converge. In this situation, the fake links to other documents are added. These links have the power of the PageRank score of the disconnect document divided by number of nodes.

Formula 11, Calculating of the PageRank score.

$$P(i)=(1-d)+d\sum_{(j,i)\in E}\frac{P(j)}{Oj} \quad (11)$$

Where P(i) is the PageRank score of the document i,
P(j) is the PageRank score of the document j,
Oj is the number of out-links from the document j,
d is residual probability, usually 0.85
And E is the set of all the hyperlinks

It means the prestige of the document j is divided between all the documents where the hyperlinks form j are pointing. Prestige of the particular document i is calculated as the sum of the partial prestiges of the documents pointing to the document i.

Power iteration
Power iteration is the algorithm for calculating the page rank values. I can say in a simplified way, that this algorithm calculates the scores of all the nodes iteratively using the Formula 13.
The following snippet is the power iteration used for calculation of PageRank values organized in square matrix. The code is originally from github project [18], it is written in Python language.

Algorithm 15: Power iteration used for PageRank calculation

```
While number of iterations < maxIterations and EuclideanNorm(delta) > epsilon:
    oldState = state.copy()
    state = state.dot(transitionProbs)
    delta = state - oldState
```

Where:
maxIterations is the integer value of maximal allowed iterations
oldState is the state of matrix in the beginning of iteration, it is pandas set of couples: node – value
state is the object of same format as oldState, containing the states of nodes after the matrix multiplication

A significant advantage of the PageRank algorithm consists in its ability to detect spam or fake Web pages. A fake Web page will not have many in-links which would result into a low PageRank score. It would make pages like that irrelevant for Web search.
The fact that PageRank does not consider the queries makes this algorithm difficult to influence or manipulate the search results.

11.3 Example use of PageRank algorithm

Before using PageRank with more complicated input data, I have used smaller, 9x9 matrix.

It is the communication matrix of top 9 most important persons from the histogram statistics experiment. It should be more simple for me to evaluate the results.

Table 17: PageRank score of the top 9 most important persons

sara.shackleton@enron.com	0.131
steven.kean@enron.com	0.200
kay.mann@enron.com	0.134
sally.beck@enron.com	0.029
jeff.dasovich@enron.com	0.187
tana.jones@enron.com	0.101
gerald.nemec@enron.com	0.045
chris.germany@enron.com	0.145
vince.kaminski@enron.com	0.029

PageRank algorithm put the highest score to Sara Shackleton followed by Steven Kean. As you can see in the graph 1 in the chapter about histogram statistics, the communication graph of these 9 former Enron employees is disconnected, forming 2 sub graphs. The most important nodes of these 2 sub graphs are nodes of Sara Shackleton and Steven Kean. In the second sub graph, Kay Mann could look more important. This node has many out links but just few incoming links. In this situation, the much lower PageRank score is correct.
This PageRank experiment converged after 57 iterations.

Ideas about use of PageRank in my experiments
PageRank was developed to weight documents placed on the World Wide Web. However, it should be possible to use it also for the data I am using in the experiments. In this work, I am using basically 2 main views on the Enron data set. One is email based, which is probably not suitable to be used with the PageRank algorithm, because the structure of emails is not possible to rationally convert to the form similar to web documents with links. The second one is person based, which looks more compatible. I can use the persons as the nodes and the emails between persons as the hyperlinks. I can simply in-links and also out-links in this context. PageRank finds originally the most important web pages on the Internet. But it can help me to find the most important persons in the data set, which is valuable information in the study and data mining of the social networks. With PageRank, I can find the most important persons in the set in more correct way than with my previous histogram statistics experiment.

11.4 My PageRank experiment

My PageRank experiment is based on the ideas discussed in the previous paragraph. I have used the persons as the nodes in the network and their communication as the links between them. So each email sent by particular person becomes an out link. I can apply the idea of interconnected web pages to interconnected persons with their communication.
Working with the persons and their communication is not new in this work. I have examined the mutual communication in the simplified way in person- based clustering, associative memories and also in the histogram statistics experiment. In a simplified way I mean that I consider only the existence of the email sent by person A to person B, but I do not examine the content of the message.

In this experiment, I also needed to choose a subset of all the possible persons, otherwise I would have probably the set of too many emails from too many persons which would be just slightly interconnected.

Choice of input data for my PageRank experiment

In the previous paragraph, I have mentioned the need of a reduced set of persons in the network and also the fact, that I have already used the representation of the Enron data set as the set of persons interconnected by their communication. I have used the same approach in the histogram statistics experiment.

I have decided to use the data collected and processed in histogram statistics experiment as the input for my PageRank experiment. The output of histogram statistics experiment was the matrix of mutual communication of 72 most important persons chosen by the criteria described in the chapter about histogram statistics experiment.

The input matrix is displayed in Picture 1 in the chapter about histogram statistics.

The implementation of the PageRank algorithm

The PageRank algorithm is relatively simple and strictly defined since its first introduction. I have decided to use some public available implementation, which is compatible with my data and can be used without significant modifications. I have chosen the simple Python based implementation using Pandas library available on Github. The author of the original implementation is Ashkon Farhangi. [21] and the source code is available on GitHub under standard Apache license.

11.5 The results of the PageRank experiment

I have calculated the following PageRank values for the 72 most important persons from Enron email set:

Table 18: The PageRank values for 72 most important persons from Enron email data set

The email address of person	PageRank value
mark.taylor@enron.com	0.047
hunter.shively@enron.com	0.008
phillip.allen@enron.com	0.016
ginger.dernehl@enron.com	0.005
patti.thompson@enron.com	0.007
mary.cook@enron.com	0.015
liz.taylor@enron.com	0.006
larry.campbell@enron.com	0.008
enron.announcements@enron.com	0.002
elizabeth.sager@enron.com	0.014
debra.perlingiere@enron.com	0.008
benjamin.rogers@enron.com	0.003
daren.farmer@enron.com	0.003
shirley.crenshaw@enron.com	0.017
marie.heard@enron.com	0.009
mike.mcconnell@enron.com	0.012
steven.kean@enron.com	0.042
sherri.sera@enron.com	0.005

m..love@enron.com	0.003
kay.mann@enron.com	0.007
janette.elbertson@enron.com	0.011
susan.mara@enron.com	0.020
james.steffes@enron.com	0.032
stanley.horton@enron.com	0.009
rick.buy@enron.com	0.018
shona.wilson@enron.com	0.006
maureen.mcvicker@enron.com	0.016
david.forster@enron.com	0.016
mike.grigsby@enron.com	0.015
mark.greenberg@enron.com	0.005
robin.rodrigue@enron.com	0.003
mary.hain@enron.com	0.007
lynn.blair@enron.com	0.006
darron.giron@enron.com	0.010
christi.nicolay@enron.com	0.007
john.arnold@enron.com	0.023
kimberly.watson@enron.com	0.012
shelley.corman@enron.com	0.015
michelle.cash@enron.com	0.007
tori.kuykendall@enron.com	0.008
richard.sanders@enron.com	0.012
vince.kaminski@enron.com	0.024
matthew.lenhart@enron.com	0.020
chris.germany@enron.com	0.003
kate.symes@enron.com	0.002
rod.hayslett@enron.com	0.007
louise.kitchen@enron.com	0.028
tana.jones@enron.com	0.032
outlook.team@enron.com	0.004
twanda.sweet@enron.com	0.009
dan.hyvl@enron.com	0.009
drew.fossum@enron.com	0.010
richard.shapiro@enron.com	0.037
sara.shackleton@enron.com	0.028
john.lavorato@enron.com	0.043
alan.comnes@enron.com	0.025
eric.bass@enron.com	0.019
carol.clair@enron.com	0.018
rosalee.fleming@enron.com	0.005
jeffrey.shankman@enron.com	0.018
scott.neal@enron.com	0.006
mark.haedicke@enron.com	0.017
phillip.love@enron.com	0.018
sally.beck@enron.com	0.015
errol.mclaughlin@enron.com	0.008
david.delainey@enron.com	0.021

susan.scott@enron.com	0.012
d..steffes@enron.com	0.007
gerald.nemec@enron.com	0.005
chris.dorland@enron.com	0.003
jeff.dasovich@enron.com	0.040
kevin.hyatt@enron.com	0.018

In the Table 14, 9 rows are marked yellow. These 9 persons are the members of the set of 9 most important persons from the histogram statistics experiment.
The average PageRank score of these 72 scores is 0.0139. The average score of these 9 persons is slightly higher: 0.0215, that can support the ideas and algorithm which have chosen them.
We can also see another interesting fact. For example, the email address enron.announcements@enron.com made it to the top 72 addresses. This address most probably belongs to the account for internal bulk mail sending. Most of the people received emails from this address, which caused this address to appear in the 72 most important addresses. But the PageRank score of this address is only 0.002083, which is below average and supports the idea of a bulk mail sending account.
It is important not to forget, how these 72 (9) people were chosen. They were not chosen based on the company position but the frequency of communication and the number of people they were writing emails. This strategy probably cannot show the member of the board, because they usually do not communicate with hundreds of employees. Choosing people because of they company position may sound logical, but strategy based on communication is much more suitable for my experiments. With this strategy, I should be able to find on some point important nodes in completely unknown data, without having any information about the set. Which is more important and interesting.

PageRank algorithm has chosen different list of the most important persons from the data set. It is not surprising, because PageRank is more advanced tool than simple histogram statistics.
Histogram statistics can be manipulated with spam sending node, which is not possible in PageRank. PageRank has chosen the following persons as the 10 most important ones:

Table 19: 10 most important persons and their score, based on the PageRank experiment

Email address	PageRank score
mark.taylor@enron.com	0.047
john.lavorato@enron.com	0.043
steven.kean@enron.com	0.042
jeff.dasovich@enron.com	0.040
richard.shapiro@enron.com	0.037
tana.jones@enron.com	0.032
james.steffes@enron.com	0.032
louise.kitchen@enron.com	0.028
sara.shackleton@enron.com	0.028
alan.comnes@enron.com	0.025

Jeff Dasovich was Enron employee, government relation's executive. It is possible, that his email communication and its importance significantly increased during the fall of Enron and investigation.
Louise Kitchen was important gas trader active in many Enron transactions. It is not possible to find a information about each person chosen to the list of the most important persons, because the bankruptcy happened in the time, when the pages like LinkedIn did not exist or contained less amount of data.

The higher company rank and importance of the persons in the top 10 list chosen by PageRank support the idea of usability of this algorithm for data like mine. It is also important to notice that the input of PageRank algorithm, the matrix of communication of 72 persons was pre-chosen by the histogram statistics experiment, which could influence the results.

Parameters of execution
The end of power iteration was defined as a state when the difference between previous and current state is smaller than epsilon. If this will not be fulfilled in 1000 iterations, the program is terminated. Epsilon was set to 0.00001. However, the algorithm converged after 25 iterations. The program needed about 2 seconds to finish on average laptop computer (Core 2 Duo, 2.53 GHz CPU with 8 GB DDR3 RAM), so no advanced hardware was necessary.

11.6 Conclusions
The PageRank algorithm is the algorithm developed to measure the relative importance of the document in the set of documents interconnected by means of hyperlinks. It was originally developed for evaluating the importance of the pages on World Wide Web. However, it can be used in many different situations, for example for measuring the importance of persons in a social network, too.
I have successfully applied this algorithm to the Enron data set and was able to calculate the PageRank score for the persons and their communication as the links between them.

12. HITS algorithm

12.1 Motivation

The main objective of the Hyperlink-Induced Topic Search (HITS) algorithm is similar to PageRank. It is used to identify the most important documents in the set of documents interconnected by means of hyperlinks, typically World Wide Web. It was introduced in the year 1998, the same year as PageRank, by Jon Kleinberg.[22]
An important difference between PageRank and HITS is that HITS is not an offline algorithm, it is also query dependent.

The main idea of HITS

HITS provides the evaluated documents with 2 kinds of ranking. It is an authority ranking and a hub ranking. For this ranking, HITS also uses the in-links and out-links like PageRank.
Authority ranking is based on the number of in-links. When many pages point to some page, it is expected that this page contains some important information which can be trusted. This page becomes authority in some topic, many people, pages, documents trust it.
Hub ranking is measured by the number of out-links. Mostly in the earlier times of World Wide Web, the most important pages contained long lists of relevant links to other pages. A typical example of this kind of page was the original concept of Yahoo. Yahoo started as the long, complex, structured and relevant list of hyperlinks to other pages. This idea is a long time obsolete, replaced by Web search engines like Google. These Web pages can be considered as the hub pages of today, thus their Hub ranking according to the HITS algorithm would be higher.
The key idea of HITS is that a good hub points to many good authorities and a good authority is pointed to by many good hubs.

12.2 Algorithm description

Description of HITS algorithm is based on the description in the book of Bing Liu[20]
The HITS algorithm needs to collect the relevant pages about some topic first. This collection is search query connected. The input of this algorithm are the documents on World Wide Web.

Algorithm 16: collecting of relevant web pages by HITS

Used variables:
q – search query
e- search engine
t- number of taken web pages from query results, usually 200
W – set of web pages returned by query
p – one web page from W
S – set of web pages prepared as input for HITS
k – number of web pages considered in initializing the in and out links, usually 50

-computer sends query *q* to the chosen search engine *e*
-*e* returns the set of results and the first *t* most relevant is taken
-*t* web pages create the set *W*
-create an empty set of web pages *S*
-for each web page *p* in *W*
    -put *p* into *S*
    -take top *k* web pages pointing to *p* and put them to *S*
    -take top *k* web pages where *p* points and put them to *S*

The query is dependent on the particular use of this algorithm, also, the order of the pages depends on the query. In my experiment, the query is represented by the choice of the most important persons provided by the histogram statistics experiment.

The output of this algorithm is the set of web pages *S*. The *t* is usually 200 and *k* is usually 50.

[20]HITS works on the pages in S, and assigns every page in S an authority score and a hub score. Let the number of pages to be studied be n. We again use G = (V, E) to denote the (directed) link graph of S. V is the set of pages (or nodes) and E is the set of directed edges (or links). We use L to denote the adjacency matrix of the graph.

Formula 12, adjacency matrix calculation

$$L(i,j) = \begin{cases} 1 \; if \; (i,j) \in E \\ 0 \; otherwise \end{cases} \quad (12)$$

Let the authority score of the page i be a(i), and the hub score of page i be h(i). The mutual reinforcing relationship of the two scores is represented as follows:

Formula 13, authority score calculation

$$a(i) = \sum_{(j,i) \in E} h(j) \quad (13)$$

And similarly the hub score:

Formula 14, hub score calculation

$$h(i) = \sum_{(j,i)\in E} a(j) \quad (14)$$

After the update of the scores is performed, it is also necessary to normalize the authority and hub scores of each document. The hub score is divided by the square root of the sum of the squares of all hub scores, authority score is similarly divided by square root of the sum of the squares of all authority scores.

Algorithm 17: calculating authority score and hub score by HITS

-assign authority score and hub score 1 to each page in the set *S*
-for each page *p* in *S*
    -update the authority score *as* of p
    -update the hub *hs* score of *p*
    -normalize *as*
    -normalize *hs*

Possible use of HITS in my experiments

The possibility to use HITS in my social network experiments is similar to the use of PageRank.

I can use the persons involved in email communication as the pages in HITS. The emails between the persons can be interpreted as the hyperlinks and the calculation of authority and hub scores would be possible.

The interpretation of these results can be interesting. PageRank simply indicates the most important pages in the network, graph etc. But HITS indicates 2 types of important pages. It can be interesting to see what kind of persons would have higher authority score or hub score. Also, unlike PageRank, HITS is query oriented, so I need to consider also the choice of some equivalent of search query for my experiment.

12.3 My HITS experiment

In my HITS experiment, I have faced the similar challenges like in the PageRank experiment. Both these algorithms were intended to be used with World Wide Web documents and not set of emails. I needed to transform the idea of HITS for use with my social network, Enron email data set. The hub score and the authority score can be interesting information about persons in the data set. However, the interpretation of results can be more difficult than in PageRank experiment. The PageRank score in my experiment, can be interpreted simply as the importance of the

node in my set, based on the communication. Hub score and authority score of HITS experiment can be also interpreted as some kind of importance, but it is interesting to think independently about hub and authority score.
If the persons has the higher hub node, it can be interpreted that this person is the frequented author of emails. But because of the nature of the HITS algorithm, simple spamming would not cause up average hub score. It means that this person needs not only to send many emails, but also receive emails from important persons/nodes.
The meaning of higher authority score is probably reversed but using the same logic. The person with higher authority score should be the receiver of up average amount of emails. Both information can be interesting in the examination of the social network.

Choice of input data for my HITS experiment
As I have mentioned before, there are 2 main points of view on the date in my work. One is email centric, second is person centric. For the HITS experiment, the person based approach is much more suitable. It is a similar situation to the PageRank experiment. But the input data cannot be in the same format. Unlike HITS, PageRank can work with the simple communication matrix. The data representation for HITS should be slightly more complicated for representing 2 scores for one node. I have decided to represent one node as a specific Python object, which is closer to the representations in my clustering experiments.

The implementation of HITS algorithm
As the base for the implementation of HITS algorithm, I have used the program of Matt Garret from Github.[23] This program contained the scoring algorithm, but it was not directly applicable on my data.
I needed to transform the emails from data set into objects representing persons and containing the hub and authority score. But this task is nothing new, I needed to perform similar thing in most of my experiments. So I could reuse significant part of the code developed for the creation of patterns in person based clustering. This code could be simplified, because the patterns in this experiment contain only the identification – email address, hub and authority score.
The last stage of development was connecting my part of code for patterns creation to the HITS algorithm created by Matt Garret.

12.4 Results of my HITS experiment

For my experiment, I have chosen different input data than for PageRank experiment. As I have mentioned in the previous paragraph, the HITS implementation I have used shared some part of the code and ideas with person based clustering, so I have decided to use the similar input data, which means the emails of Phillip Allen. The choice of emails from folder of Phillip Allen represents the query used in HITS algorithm.
Turning all the email senders from this folder into patterns leads to hundreds of patterns and not all of them carry any important information. So I have reduced the output of the algorithm only on patterns which have hub or authority score greater than 0.2 .

Table 20: results of HITS experiment with the emails of Phillip Allen.

email address	hub score	authority score
phillip.allen@enron.com	0.250	0.025
ina.rangel@enron.com	0.219	0.121
stephanie.miller@enron.com	0.0	0.190
tim.heizenrader@enron.com	0.111	0.073
frank.hayden@enron.com	0.061	0.107
tim.belden@enron.com	0.002	0.138
ray.alvarez@enron.com	0.110	0.025
w..cantrell@enron.com	0.198	0.063
veronica.espinoza@enron.com	0.171	0.0
mike.grigsby@enron.com	0.159	0.370
karen.buckley@enron.com	0.144	0.059
james.bruce@enron.com	0.190	0.053
jeff.richter@enron.com	0.058	0.101
keith.holst@enron.com	0.058	0.356
m..tholt@enron.com	0.173	0.256
randy.bhatia@enron.com	0.058	0.131
ryan.o'rourke@enron.com	0.260	0.131
michelle.akers@enron.com	0.155	0.0
technology.enron@enron.com	0.231	0.0
carole.frank@enron.com	0.251	0.096
patti.sullivan@enron.com	0.136	0.177
ashley.worthing@enron.com	0.223	0.096
kam.keiser@enron.com	0.242	0.128
biliana.pehlivanova@enron.com	0.167	0.112
john.lavorato@enron.com	0.058	0.141
louise.kitchen@enron.com	0.137	0.039
k..allen@enron.com	0.353	0.587

Phillip Allen can serve as the check of the whole approach. He appeared in the reduced set of nodes and his hub score is up average. On the other hand, his authority score is sub average. Phillip Allen can really serve as the hub of this subset because all these emails are from his folder. It can explain higher hub score, but lower authority score.

The significant differences between the hub and authority scores in the results indicate, that HITS is applicable to the tasks like mine. The results are not just copy of PageRank score, because the different hub and authority scores have been calculated.

Parameters of execution

This HITS algorithm performed 100 iterations before printing the results from the previous paragraph. This number was pre-set. The original code and my modifications were both written in Python 3. No special hardware or software is necessary.

12.5 Conclusions

HITS is another algorithm developed for ranking of web pages that can be used in my experiments with social networks. However, the modification of HITS for use with the emails from Enron is more difficult than the modification and use of PageRank. I needed to consider the correct query implementation and be able to interpret the correct meaning of the obtained authority and hub scores.

However, the implementation described in this chapter shows, that the use of HITS with the email based social network is possible and the WWW documents can be successfully replaced with the persons and links between the documents can be described using the mutual communication of the persons.

13. Implementation

13.1 Motivation

Important part of this work is the implementation of the described algorithms. Without the implementation, my whole work would be just theory. I need to prove my ideas and perform the experiments.
I have programmed most of the scripts in this work. The rest of the scripts (PageRank, HITS, BAM) are downloaded from Github or based on the scripts from Github. My implementations are naturally based on the description of the algorithms quoted in this work. The reason why this approach was used is the modifiability.
I needed to process the data based on the text files and it was necessary to transform Enron email corpus to the collections suitable for the experiments. I also needed to significantly modify the k-means algorithm for use with CUDA. There are frameworks and libraries, which could be used to perform my experiments, but it could be difficult to perform the performed modifications.
It was also simpler to integrate python scripts from Github to my experiments than use a framework for the scripts I did not personally implement.

13.2 Programming language and used environment

Choice of a programming language is and important decision in the beginning of the development. Not all the programming languages have the same abilities and are not equally powerful in the given tasks. Choice of programming language often means also the choice of platform. Sometimes, the specialized libraries are not platform independent, so it is necessary to consider this before the choice of the language.
I have chosen Python programming language for my experiments. Python is well suitable for my experiments, because it supports the libraries I need (for example pycuda for CUDA), and it works well on the computers I have used for calculations.
The program was developed and tested in Unix environment.

In the beginning, I have performed all my calculations on my aging laptop computer. It has 2.53 GHz Intel Core 2 Duo CPU and 8 GB DDR RAM. It has sufficient computational power for the development of the most of the scripts described in this work. But this computer was not capable for the more complicated calculations, namely clustering, using more data or CUDA. For these calculations, I have used the computer described in table 11 in detail, with Intel Core i5 CPU and 16GB RAM.

13.3 General structure of my application

My application consists of several independent scripts. Usually, one experiment means one script. This is true for all the experiments except for Histogram statistics experiment. Histogram statistics is divided into 3 steps. Each step means a separate script. The main idea why to divide this experiment into 3 scripts was the scalability. I did not know, how long time it will take to process all the necessary data and I wanted to make computational “check points”. Another reason is the fact, that the 3rd script from the Histogram statistics experiment is different from the rest of the scripts of this experiment and also from the scripts from the different experiments. This last

part of the experiment uses tkinter library and plots a graph. The rest of the scripts in my work are terminal only.

13.4 Conclusion

The structure of my application is mostly described in the chapters about particular algorithms. The decisions, which led me to the final structure of the application, are also described in these chapters. All the scripts can work separately, with the correct input.

In this chapter, I wanted to mention the decisions, which shaped my application but were not included in any other previous chapter.

I have also briefly described the necessary environment, but the detail description will be provided in the documentation.

Also, all the necessary source codes are provided on the optical disc, which is part of this work.

14. Summary

In this work, I have used various data mining techniques and algorithms to examine the social network. I would like to summarize my work and try to compare the used algorithms from the point of view of my data mining experiments and their results.

14.1 My application

My application consists from the set of scripts written in Python 3 language. It is not a monolith application; it is possible to run the experiments separately. But all the experiments use the same input data, the Enron email corpus.
Python is supported by CUDA framework, which was important for my work. It also integrates very well to Linux environment, which was important because of the operating system on the machines I have used to compute the results. Python has also many other libraries and frameworks which helped my development, for example Pandas or Tkinter.

14.2 Accomplishments of my work

In my work, I have implemented and test various algorithms using the social network created from the Enron email corpus. I succeeded in identification of the most important nodes in this social network. Ranking algorithms and histogram statistics experiment were able to find persons with above average importance in the Enron company, just using their email communication.
I have developed implementation of K-means clustering algorithm with significantly lower time requirements using CUDA. This script was successfully tested with Enron data social network.
I have used neural networks for prediction of hidden links in the social network. But these experiments were very basic and did not use the full potential of neural networks.

Table 21: High-level comparison of used algorithms:

Name of the algorithm	**Scalability**	**Time requirements**	**Interpretability**	**Running in parallel environment**
K-means clustering	***	**	**	***
Histogram statistics	**	***	**	**
Hopfield net	*	***	***	*
BAM	*	***	***	*
PageRank	***	***	***	*
HITS	***	***	***	*

14.3 Further work

All my experiments led to reasonable result, but significant upgrade would be possible. Date from social networks often belong to the category of Big data. In this work, I have always used a subset of the whole Enron email corpus. The highest number of used patterns was 3034. It could be interesting to perform my experiments with the full Enron data set, which would mean significantly higher amount of patterns and possibly more important results.
The implantation of CUDA kernel significantly speeded up the clustering experiments. But this approach could be used on another experiments. I could try to implement similar upgrade for the other algorithms used in this work.
In the various part of this work, I mention the universality of used ideas but my scripts are developed to work with the set of emails in separate text files only. It would be useful to run these experiments with the data from different type of social network and prove the universality of the algorithms.
Many experiments in my work produced interesting results and showed to be a good choice, maybe with the exception of neural networks. Neural networks are powerful tool and it would be worth to give them more time and maybe use other type of neural network, try to develop more successful version of my program. The most important attribute would be the capacity. When 9x9 matrix is too big, it is difficult to obtain interesting and useful results.

Bibliography

[1] Dan Noyes, “The Top 20 Valuable Facebook Statistics,” *Zephoria.com*. [Online]. Available: https://zephoria.com/top-15-valuable-facebook-statistics/. [Accessed: 22-Dec-2017].

[2] Donald Trump, “@realDonaldTrump,” *Twitter.com*. [Online]. Available: https://twitter.com/realdonaldtrump. [Accessed: 30-Dec-2017].

[3] J. MacQueen, “Some methods for classification and analysis of multivariate observations,” *University of California Press, Berkeley, Calif.*, 1967.

[4] “A tutorial on clustering algorithms,” *Home.deib.polimi.it*. [Online]. Available: http://home.deib.polimi.it/matteucc/Clustering/tutorial_html/kmeans.html. [Accessed: 30-Dec-2017].

[5] Meena Mahajan, Prajakta Nimbhorkar, and Kasturi Varadarajan, *The Planar k-Means Problem is NP-Hard*. The Institute of Mathematical Sciences, Chennai 600 113, India., 2007.

[6] Chris Seabury, “Enron Collapse,” *Investopedia*. [Online]. Available: http://www.investopedia.com/articles/stocks/09/enron-collapse.asp. [Accessed: 30-Dec-2017].

[7] Garnett Wilson and Wolfgang Banzhaf, *Discovery of Email Communication Networks from the Enron Corpus with a Genetic Algorithm using Social Network Analysis*. IEEE, 2009.

[8] G. Salton, A. Wong, and C. S. Yang, *A vector space model for automatic indexing*. Cornell University, 1975.

[9] Kavita Ganesan, “Terrier stop words list,” *Bitbucket*. [Online]. Available: https://bitbucket.org/kganes2/text-mining-resources/downloads. [Accessed: 30-Dec-2017].

[10] “NVIDIA CUDA,” *Developer NVIDIA*. [Online]. Available: http://www.nvidia.com/object/cuda_home_new.html.

[11] Chris Paciorek, “An Introduction to Using GPUs for Computation,” *University of Berkley*. [Online]. Available: http://www.stat.berkeley.edu/scf/paciorek-gpuWorkshop.html. [Accessed: 30-Dec-2017].

[12] Miguel Lazaro-Gredilla, “An introduction to CUDA using Python.” Universidad Carlos III. de Madrid, May-2013.

[13] Talomnies, “How do I choose grid and block dimensions for CUDA kernels?,” *Stackoverflow*. [Online]. Available: http://stackoverflow.com/questions/9985912/how-do-i-choose-grid-and-block-dimensions-for-cuda-kernels. [Accessed: 30-Dec-2017].

[14] W. A. Little, “Analytic study of the memory storage capacity of a neural network,” *Math.Biosci.*, 1978.

[15] J. J. Hopfield, “Neural networks and physical systems with emergent collective computational abilities,” vol. Proceedings of the National Academy of Sciences of the USA, 79 no. 8, Apr. 1982.

[16] Raul Rojas, *Neural Networks A Systematic Introduction*. Springer-Verlag, 1996.

[17] Bart Kosko, “Bidirectional Associative Memories,” *Transactions on Systems, Man, and Cybernetics*, Jan. 1988.
[18] S. Brin and L. Page, *The anatomy of a Large-Scale Hypertecxtual Web Search Engine*. Stanford University, 1998.
[19] Larry Page and S. Brin, *The PageRank Citation Ranking Bringing Order to the Web*. Stanford University, 1998.
[20] Bing Liu, *Web data mining, Exploring hyperlinks, Contents and usage Data*. Springer-Verlag, 2011.
[21] Ashkon Farhangi, “A Python implementation of Larry’s famous PageRank algorithm.,” *Github*. [Online]. Available: https://github.com/ashkonf/PageRank. [Accessed: 30-Dec-2017].
[22] Jon M. Kleinberg, *Authoritative Sources in a Hyperlinked Environment*. ACM, 1999.
[23] Matt Garret, “HITS,” *Github*. [Online]. Available: https://github.com/mattgarrett/hits/blob/master/rank.py. [Accessed: 30-Dec-2017].